Universidade de São Paulo
Instituto de Física

EXERCÍCIO PROGRAMA 4

MAP0214 – Cálculo Numérico com Aplicações em Física

Ícaro Vaz Freire
11224779

São Paulo, 2020.

I.

Resolvemos a equação diferencial ordinária de segunda ordem abaixo utilizando os métodos numéricos de Euler e Runge-Kutta:

$$\ddot{y} = 12t^2 + 4^3 - t^4 + y - \dot{y}$$

Método de Euler

Achamos a solução para y(t = 5) e dy/dt(t = 5):

y(5) = 627.6572294470509
dy/dt(5) = 503.8821211260556

Método de Runge-Kutta

Resolvemos a mesma equação acima utilizando o método de Runge-Kutta de 4ª ordem:

y(5) = 630.0150199892586
dy/dt(5) = 503.006003973503

II.

1. Espaço de Fase

Item A

Substituísmo a equação diferencial da questão anterior por uma equação de um oscilador harmônico simples, sem amortecimento ou forçamento, por enquanto:

$$\ddot{x} - \frac{1}{2}x(1 - x^2) = 0$$

Resolvemos a equação acima passando como condições iniciais do problema x(0) = 1 e dx/dt(0) = -0.1, -0.5 e -1.0 e fazemos os diagramas de espaço de fase para cada condição:

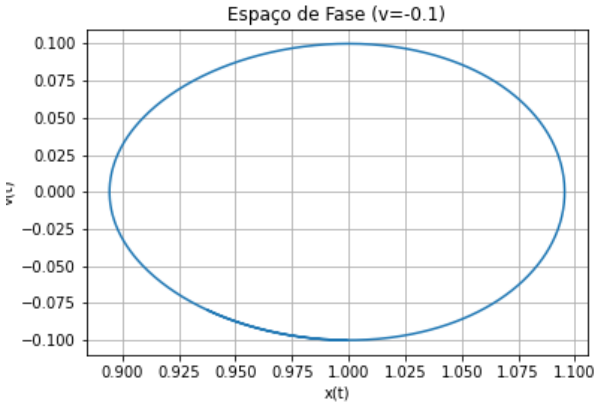


Figura 1. Diagrama do espaço de fase para v(0) = -0.1.

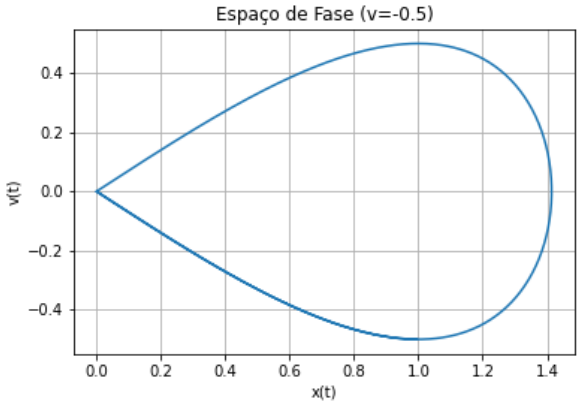


Figura 2. Diagrama do espaço de fase para v(0) = -0.5.

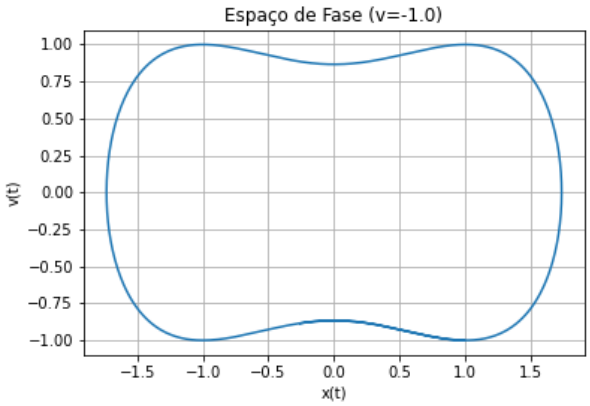


Figura 3. Diagrama do espaço de fase para v(0) = -1.0.

Item B

Aplicamos um termo de amortecimento na expressão do item anteriror e resolvermos para condições inicicias similares: x(0) = 1, dx/dt(0) = -1:

$$\ddot{x}+2\gamma\dot{x}-\frac{1}{2}x\left(1-x^2\right)=0$$

Resolvemos para $2\gamma = 0.25$ e $2\gamma = 0.8$:

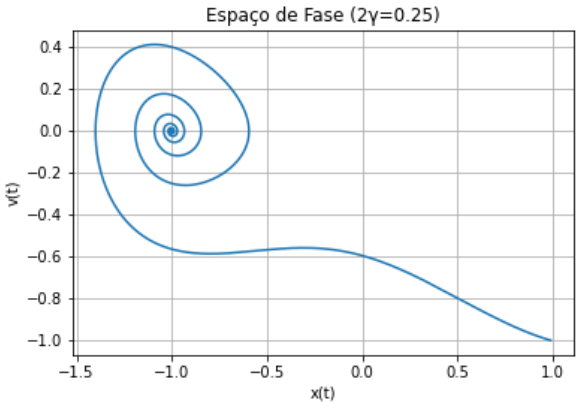


Figura 4. Diagrama do espaço de fase para fator de amortecimento 2γ=0.25.

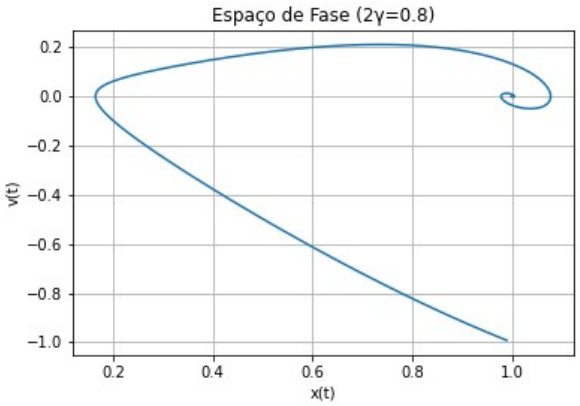


Figura 5. Diagrama do espaço de fase para fator de amortecimento 2γ=0.8.

Item C

Agora, aplicamos também o termo de forçamento na equação do oscilador, portanto, se tornando um oscilador harmônico amortecido forçado.

$$\ddot{x}+2\gamma\dot{x}-\frac{1}{2}x\left(1-x^2\right)=F\cos\left(\omega t\right)$$

onde, F é o fator de escala da força e ω a frequência de oscilação da força. Resolvemos a equação com as condições iniciais de: x(0) = 1, v(0) = -1, ω = 1 e F = 0.22, 0.23, 0.28, 0.35 e 0.6.

Antes de começar a recolher os pontos para montar o espaço de fase, eliminamos o transiente evoluindo o sistema 200000 passos.

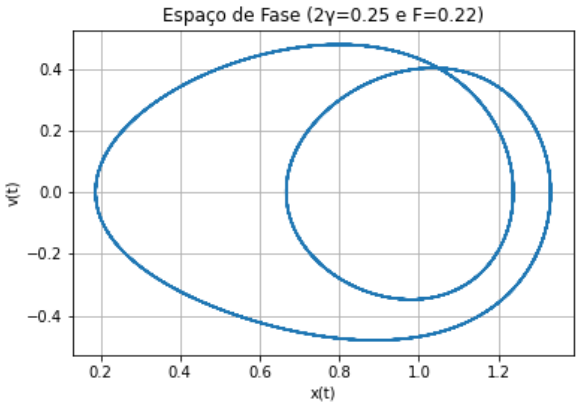


Figura 6. Diagrama do espaço de fase para fator de amortecimento 2γ=0.25 e fator de força F=0.22.

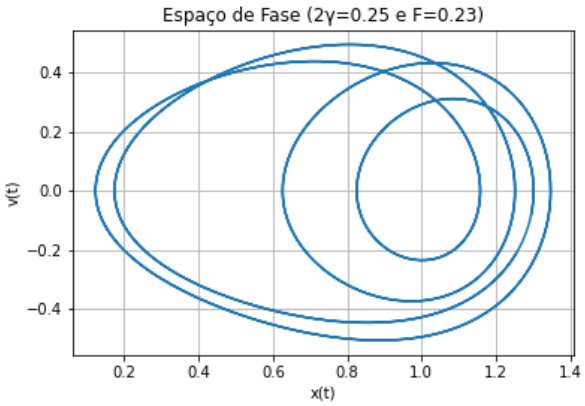


Figura 7. Diagrama do espaço de fase para fator de amortecimento 2γ=0.25 e fator de força F=0.23.

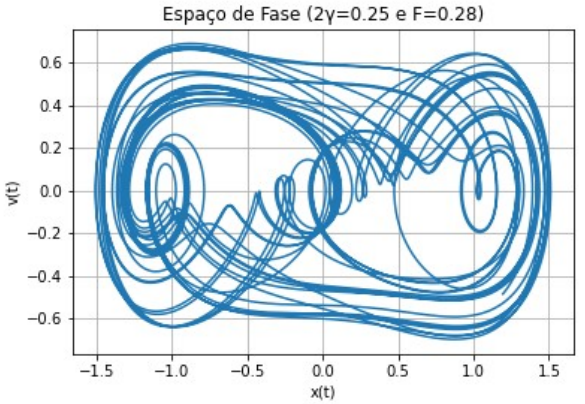


Figura 8. Diagrama do espaço de fase para fator de amortecimento 2γ=0.25 e fator de força F=0.28.

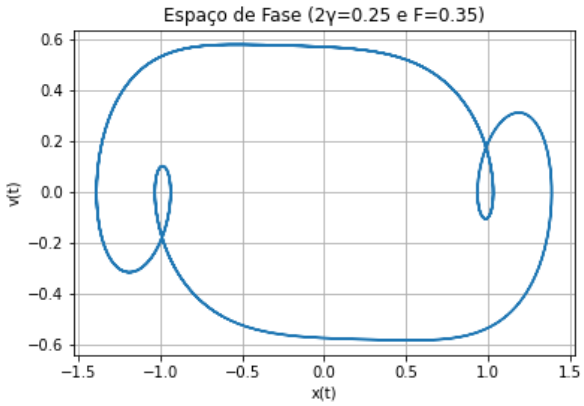


Figura 9. Diagrama do espaço de fase para fator de amortecimento 2γ=0.25 e fator de força F=0.35.

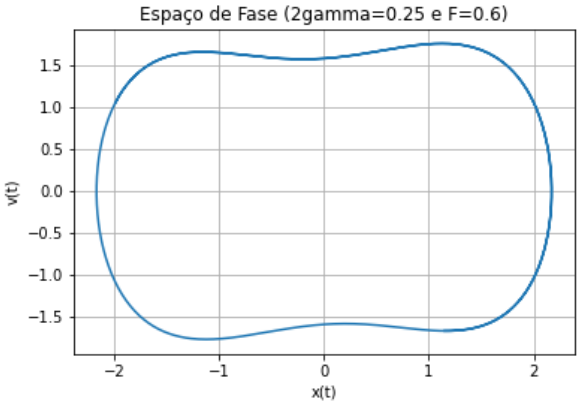


Figura 10. Diagrama do espaço de fase para fator de amortecimento 2γ=0.25 e fator de força F=0.6.

Item D

Observando como os pontos se distribuem no espaço de fase dos sistemas, observamos a existência de dois pontos atratores globais:

$x_1 = -1$ e $v_1 = 0$
$x_2 = 1$ e $v_2 = 0$

2. Diagrama de Bifurcação

Variamos a força sobre o sistema de 0 a 0.7 com passos 0.0005 e capturamos os pontos finais para evolução dos perídodos dos sistema:

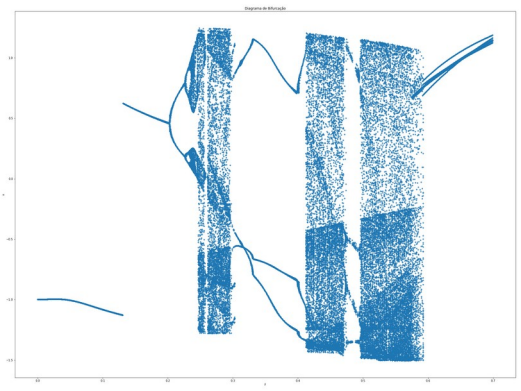


Figura 11. Diagrama de bifurcação.

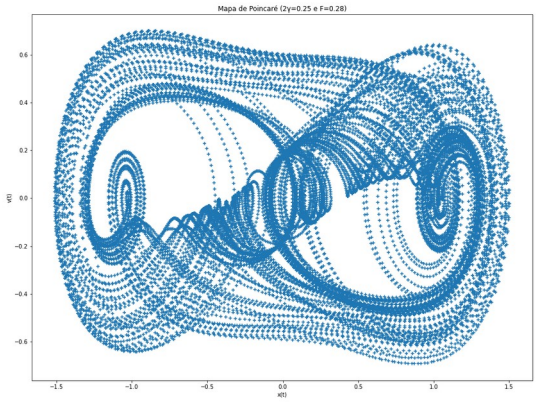
A partir da Figura 11, vemos como os pontos reagem a mudanças pequenas nas condições iniciais do sistema. Há intervalos onde a evolução do sistema é ordenada enquanto há outros caóticos.

A partir do diagrama de bifurcação, podemos fazer uma estimativa da constante de Feigenbaum, uma constante que determina a proporção entre duas bifurcações consecutivas do diagrama:

$$\delta = 4.66$$

3. Mapa de Poincaré

O mapa de Poincaré é obtido coletando os pontos confirme o sistema evolui, porém com condições iniciais fixas. Fazemos o mapa utilizando as mesmas condições iniciais da questão anterior, porém com F = 0 .28:



Figuura 12. Mapa de Poincaré para F = 0.28.

PROGRAMAS

I.

Método de Euler

```
import numpy as np
import matplotlib.pyplot as plt

# Definimos a função a ser resolvida
def g(t, y, z): return 12 * t**2 + 4 * t**3 - t**4 + y - z

# Definimos as condições iniciais do problema
h = 0.01
y = 0
z = 0
t = 0
N = int(6 / h)

# Matriz que contém os valores de t, x e v
axis = np.zeros((3, N))

# Método de Euler
for i in range(N):
    y += h * z
    z += h * g(t, y, z)

    axis[0][i] = t
    axis[1][i] = y
    axis[2][i] = z

    t += h
```

Método de Runge-Kutta

```
import numpy as np
import matplotlib.pyplot as plt

# Definimos a função que será resolvida
def g(t, y, z): return 12 * t**2 + 4 * t**3 - t**4 + y - z

# Determinamos as condições iniciais
h = 0.01
y = 0
z = 0
t = 0
N = int(6 / h)

# Estrutura de dados
axis = np.zeros((3, N))

# Método de Runge-Kutta
for i in range(N):
    k1y = h * z
    k1z = h * g(t, y, z)
    k2y = h * (z + k1z / 2)
    k2z = h * g(t + h / 2, y + k1y / 2, z + k1z / 2)
    k3y = h * (z + k2z / 2)
    k3z = h * g(t + h / 2, y + k2y / 2, z + k2z / 2)
    k4y = h * (z + k3z)
    k4z = h * g(t + h, y + k3y, z + k3z)

    y += (k1y + 2 * k2y + 2 * k3y + k4y) / 6
    z += (k1z + 2 * k2z + 2 * k3z + k4z) / 6

    axis[0][i] = t
    axis[1][i] = y
    axis[2][i] = z

    t += h
    t += h
```

II-1C.

```
import numpy as np
import matplotlib.pyplot as plt

# Função que faz evolução do sistema
def calculate(t, x, v, h, N, n=0):
    if n == 0: n = N

    axis = np.zeros((3, n))
    for i in range(N):
        k1x = h * v
        k1v = h * g(t, x, v)
        k2x = h * (v + k1v / 2)
        k2v = h * g(t + h / 2, x + k1x / 2, v + k1v / 2)
        k3x = h * (v + k2v / 2)
        k3v = h * g(t + h / 2, x + k2x / 2, v + k2v / 2)
        k4x = h * (v + k3v)
        k4v = h * g(t + h, x + k3x, v + k3v)

        x += (k1x + 2 * k2x + 2 * k3x + k4x) / 6
        v += (k1v + 2 * k2v + 2 * k3v + k4v) / 6

        if i >= N - n:
            axis[0][i - (N - n)] = t
            axis[1][i - (N - n)] = x
            axis[2][i - (N - n)] = v

        t += h
    return axis

# Intervalos do fator de escala da força
Fs = [0.22, 0.23, 0.28, 0.35, 0.6]
w = 1

for F in Fs:
    # Definimos função a ser resolvida
    def g(t, x, v): return (1/2) * (x - x**3) - 0.25 * v + F * np.cos(w * t)

    # Faz o cálculo ignorando o transiente
    axis = calculate(0, 1, -1, 0.01, 200000, 50000)

    # Faz o plot dos dados do sistema
    plt.plot(axis[1], axis[2])
    plt.title(f'Espaço de Fase (2γ={F})')
    plt.xlabel('x(t)')
    plt.ylabel('v(t)')
    plt.grid()
    plt.savefig(f'ep4_II_1b_{F}.png')
    plt.show()
```

II-2.

```
import numpy as np
import matplotlib.pyplot as plt

# Frequência da força
w = 1

# Função a ser resolvida
def g(t, x, v): return (1/2) * (x - x**3) - 0.25 * v + F * np.cos(w * t)

# Estrutura para plotar dados
data = [[], []]

# Loop para variar a força
for F in np.arange(0, 0.7, 0.0005):

    # Condições iniciais
    t = 0
    x = 1
    v = -1
    h = 0.1 * 2 * np.pi / w

    # Evolui o transiente
    for i in range(200000):
        k1x = h * v
        k1v = h * g(t, x, v)
        k2x = h * (v + k1v / 2)
        k2v = h * g(t + h / 2, x + k1x / 2, v + k1v / 2)
        k3x = h * (v + k2v / 2)
        k3v = h * g(t + h / 2, x + k2x / 2, v + k2v / 2)
        k4x = h * (v + k3v)
        k4v = h * g(t + h, x + k3x, v + k3v)

        x += (k1x + 2 * k2x + 2 * k3x + k4x) / 6
        v += (k1v + 2 * k2v + 2 * k3v + k4v) / 6
        t += h

    h = 0.001 * 2 * np.pi / w

    # Evolui os períodos
    for i in range(100):
        for j in range(1000):
            k1x = h * v
            k1v = h * g(t, x, v)
            k2x = h * (v + k1v / 2)
            k2v = h * g(t + h / 2, x + k1x / 2, v + k1v / 2)
            k3x = h * (v + k2v / 2)
            k3v = h * g(t + h / 2, x + k2x / 2, v + k2v / 2)
            k4x = h * (v + k3v)
            k4v = h * g(t + h, x + k3x, v + k3v)

            x += (k1x + 2 * k2x + 2 * k3x + k4x) / 6
            v += (k1v + 2 * k2v + 2 * k3v + k4v) / 6
            t += h

        data[0].append(F)
        data[1].append(x)

# Faz o plot dos dados
plt.scatter(data[0], data[1], marker='+')
plt.title('Diagrama de Bifurcação')
plt.xlabel('F')
plt.ylabel('x')
plt.savefig('ep4_bifurcation_diagram')
plt.show()
```