EP4 - 9359365

Nicholas Funari Voltani

14 de novembro de 2019

Os *scripts* foram feitos em Python; os gráficos foram feitos com o pacote Pyplot. Os códigos dos exercícios $II-1c$ e $II-2$ estão no apêndice.

1 Parte I: Método de Euler e Runge-Kutta de Ordem 4

Temos a equação diferencial

$$\ddot{y} = \dot{y} - y - 3t^2 + 6t$$

equivalente ao sistema

$$\begin{cases} \dot{y} &= z \\ \dot{z} &= z - y - 3t^2 + 6t \end{cases}$$

Tem-se que a solução analítica é $y(t) = t^3$. Os resultados obtidos pelo método de Euler, Runge-Kutta ordem 4 e pela solução analítica são:

```
Rotina por Método de Euler (passo 0.01)
y(6.0)=215.75499450830407
v(6.0)=108.3605108964238

Rotina por Método de Runge Kutta de ordem 4 (passo 0.01)
y(6.0)=215.99386015585367
v(6.0)=107.99974341958658

Resposta analítica:
y(6) = 6^3 = 216
v(t) = 3*6^2 = 108
```

2 Parte II: Poço Duplo de Potencial (Equação de Duffing)

2.1 Exercício 1: Espaços de Fase

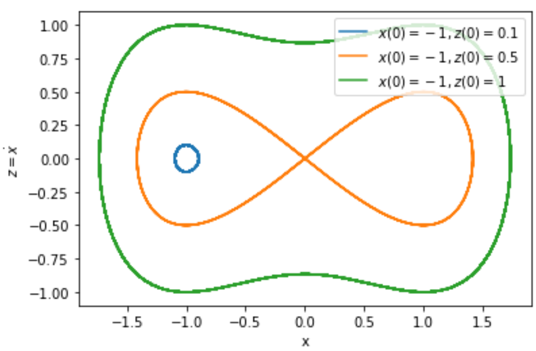

Figura 1: Espaço de fase para $\ddot{x} - \frac{1}{2}x(1-x^2) = 0$.

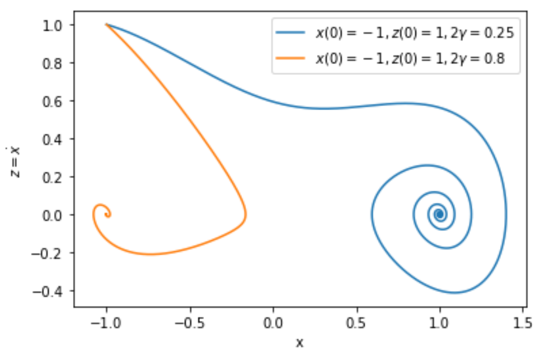

Figura 2: Espaço de fase para $\ddot{x} + 2\gamma\dot{x} - \frac{1}{2}x(1-x^2) = 0$.

Para tratar do caso forçado $\ddot{x} + 2\gamma\dot{x} - \frac{1}{2}x(1 - x^2) = F\cos(\omega t)$, foram feitos vários gráficos por fins de legibilidade; começou-se a contar o tempo a partir de 3 segundos a fim de eliminar o transiente do gráfico ($2\gamma = 0.25$).

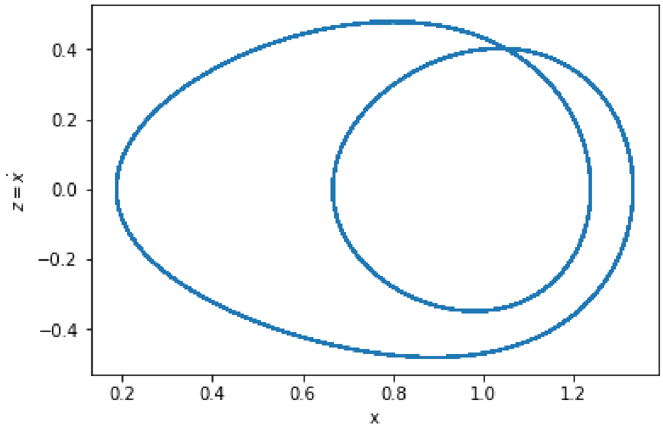

Figura 3: $F = 0.22$; $200 \leq t \leq 1000$.

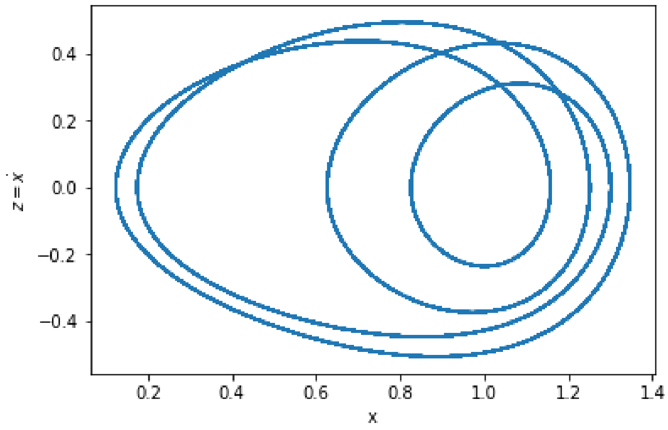

Figura 4: $F = 0.23$; $200 \leq t \leq 1000$.

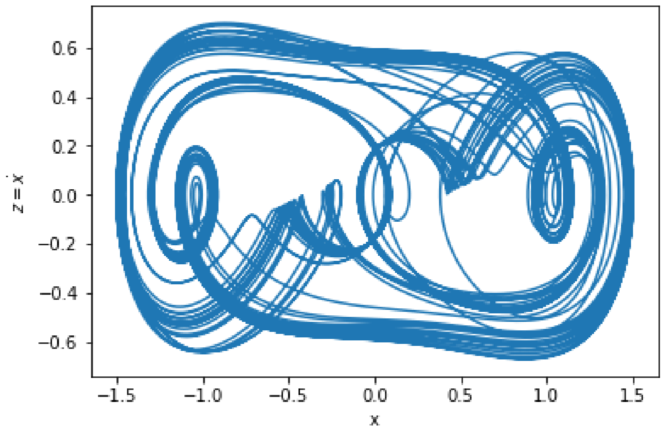

Figura 5: $F = 0.28$; $200 \leq t \leq 1000$.

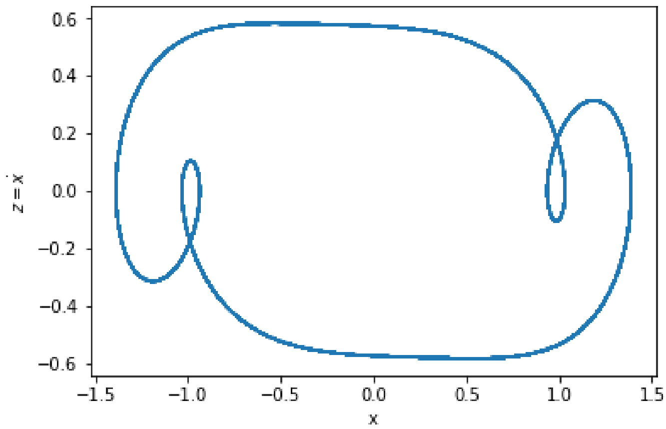

Figura 6: $F = 0.35$; $200 \leq t \leq 1000$.

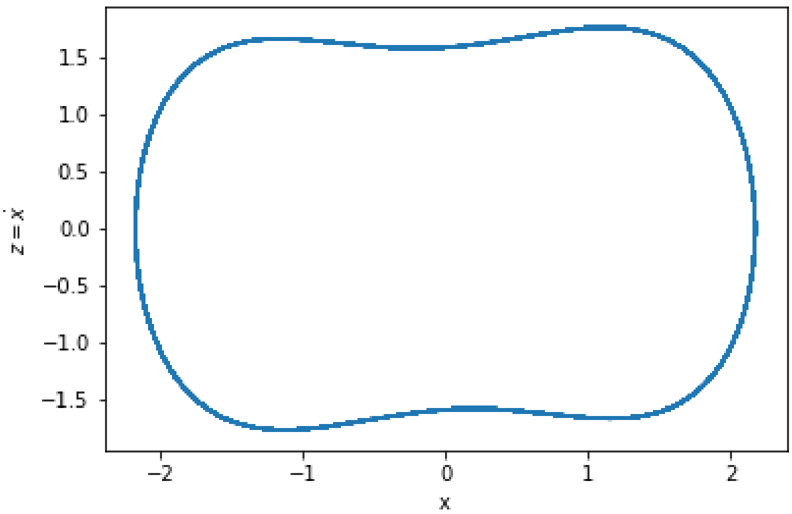

Figura 7: $F = 0.6$; $200 \leq t \leq 1000$.

Nota-se que os atratores do item a (referentes ao gráfico 1), assim como os do item c (gráficos 3 a 7) são as próprias órbitas mostradas; o movimento está restrito a essas trajetórias.

No item b (gráfico 2), com pouca atenuação, $x = 1$ é atrator; para maiores atenuações, a órbita espirala para $x = -1$.

2.2 Exercício 2: Diagrama de Bifurcação

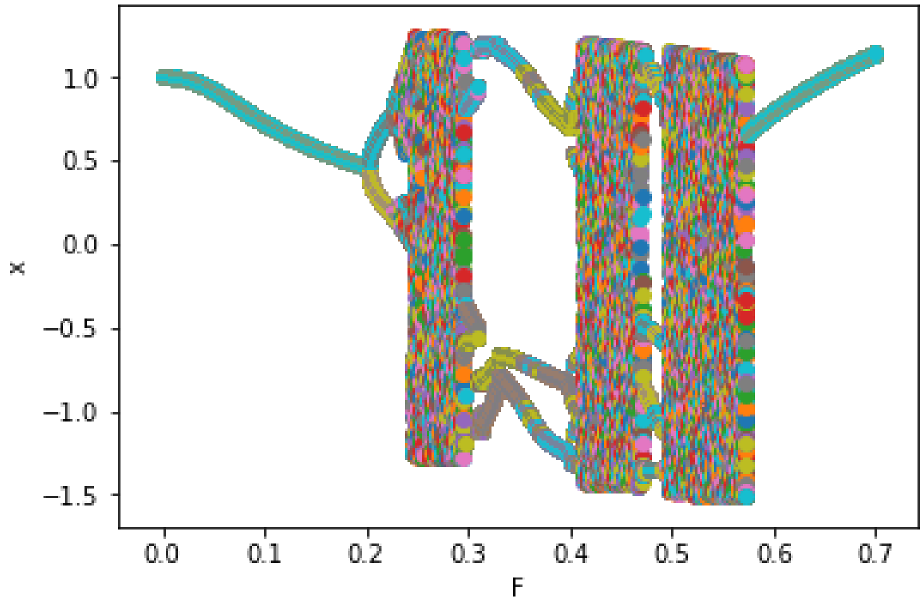

Figura 8: Diagrama de bifurcação de $\ddot{x} + 0.25\,\dot{x} - \frac{1}{2}x(1 - x^2) = 0.28\cos(t)$.

2.3 Exercício 3: Mapa de Poincaré

Para se obter o mapa de Poincaré da trajetória forçada acima, foram feitos 20000 períodos da órbita (com o transiente retirado após 200000 passos).

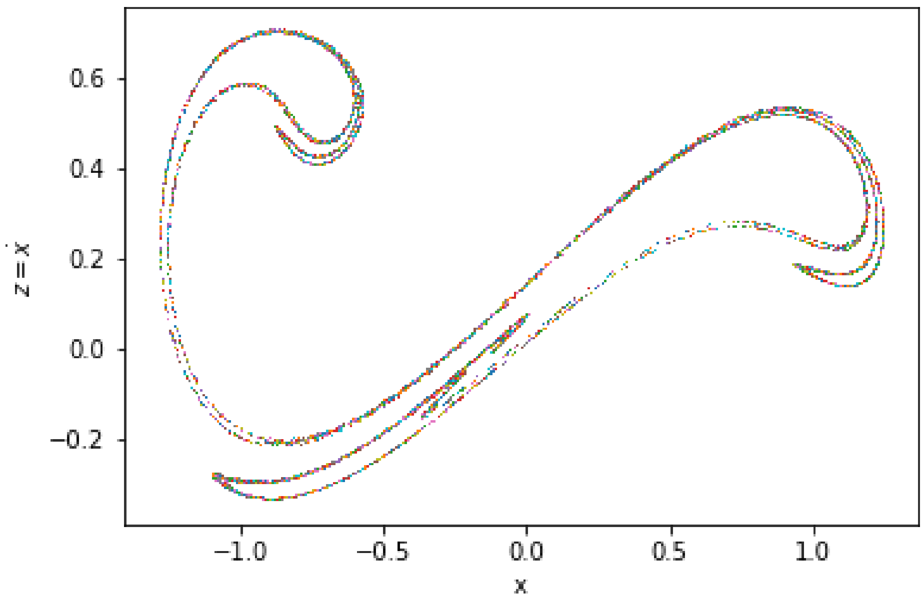

Figura 9: Mapa de Poincaré para $\ddot{x} + 0.25\,\dot{x} - \frac{1}{2}x(1 - x^2) = 0.28\cos(t)$.

A Código do exercício $II - 1c$

Code 1: Código do exercício II-1c; algumas linhas estão comentadas

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

def g(t,x,z,F):
    return x*(1-x**2)/2 - 0.25*z + F*np.cos(t)

## \dot(x)=z
## \ddot(x)=f(t,x,z)


def RotinaRK4(y,z,h,t,F):
    k1_y=h*z
    k1_z=h*g(t,y,z,F)

    k2_y=h*(z+k1_z/2)
    k2_z=h*g(t+h/2,y+k1_y/2,z+k1_z/2,F)

    k3_y=h*(z+k2_z/2)
    k3_z=h*g(t+h/2,y+k2_y/2,z+k2_z/2,F)

    k4_y=h*(z+k3_z)
    k4_z=h*g(t+h,y+k3_y,z+k3_z,F)

    y+=(1/6)*(k1_y + 2*(k2_y+k3_y)+k4_y)
    z+=(1/6)*(k1_z+2*(k2_z+k3_z)+k4_z)
    t+=h
    return y,z,t

Farray=[0.22,0.23,0.28,0.35,0.6]
h=0.001
for k in range(len(Farray)):
    x=-1
    z=1
    t=0
    xarray=[]
    zarray=[]
    for i in range(1,int(1000/h + 1)):
        x,z,t = RotinaRK4(x,z,h,t,Farray[k])
        if t>200:
            xarray.append(x)
            zarray.append(z)
    plt.figure()
    plt.plot(xarray,zarray)
    plt.xlabel("x")
    plt.ylabel("$z=\\dot{x}$")
    plt.show()
```

B Código do exercício $II - 2$

Code 2: Código do exercício II-2.

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

def g(t,x,z,F):
    return x*(1-x**2)/2 - 0.25*z + F*np.cos(t)

## \dot(x)=z
## \ddot(x)=f(t,x,z)


def RotinaRK4(y,z,h,t,F): ##Condicoes iniciais x,v
    k1_y=h*z
    k1_z=h*g(t,y,z,F)

    k2_y=h*(z+k1_z/2)
    k2_z=h*g(t+h/2,y+k1_y/2,z+k1_z/2,F)

    k3_y=h*(z+k2_z/2)
    k3_z=h*g(t+h/2,y+k2_y/2,z+k2_z/2,F)

    k4_y=h*(z+k3_z)
    k4_z=h*g(t+h,y+k3_y,z+k3_z,F)

    y+=(1/6)*(k1_y + 2*(k2_y+k3_y)+k4_y)
    z+=(1/6)*(k1_z+2*(k2_z+k3_z)+k4_z)
    t+=h
    return y,z,t

for F in np.arange(0,0.7,0.0005):
    ## Condicoes iniciais
    x=-1
    z=1
    T=2*np.pi ## Periodo da forca, omega=1
    h=0.01*T
    t=0 ## tempo inicial
    ## Retirando transiente
    for i in range(200000):
        x,z,t=RotinaRK4(x,z,h,t,F)
    print("Transiente de F={} tirado".format(F))
    ##
    h=0.001*T
    for i in range(100): ## qtd de periodos
        for j in range(1000):
            x,z,t=RotinaRK4(x,z,h,t,F) ## evolui um periodo
            plt.plot(F,x,',')
        print("F={},i={}".format(F,i))
    print("F={} feito".format(F))
plt.xlabel("F")
plt.ylabel("x")
```