Métodos Computationais em Física 2020
Pêndulo Amortecido Forçado

João Henrique de Sant'Ana

20 de abril de 2020

Introdução

Nesse manual, tenho o intuito de instruir o usuário a usar os script's que resolvem o problema do pêndulo amortecido forçado. Primeiro de tudo você precisa ter python3 instalado em sua máquina e também ter os módulos *numpy*, *matplotlib* e *scipy*. Se você estiver numa distribuição linux ou MacOS, você precisa apenas abrir o terminal e ir no diretório onde estão os aquivos *.py* e dar o seguindo comando python3 *nome_arquivo.py*. Se voce estiver num ambiente de programação é so abrir o aquivo *.py* e rodar. Não tem erro. Os arquivos são

- $espaco\_fase.py$
- $mapa\_poincare.py$
- $diagrama\_bifurcacao.py$
- $expoente\_lyapunov.py$
- $energia\_oscilador.py$

Equações

$$\begin{cases} \dot{\omega} = -\frac{1}{q}\dot{\theta} - sin(\theta) + Fsin(\Omega_d t) \\ \dot{\theta} = \omega \end{cases} \tag{1}$$

Para resolver numericamente utilizamos o algoritmo de runge-kutta de 4 ordem. Em todos os aquivos utilizamos a seguinte notação:

$$\begin{cases} w = \Omega_d \\ y = \theta \\ z = \dot{\theta} \end{cases} \tag{2}$$

Em todos os programas utilizamos a mesma função rungekutta($y0, z0, F, q, w$), onde $y0 = \theta(0)$, $z0 = \dot{\theta(0)}$ que retorna a lista tempo, y e z, além de que as variáveis h e N, intervalo do tempo e número total de elementos na lista do tempo, são variáveis globais. Em toda análise da dinâmica usamos $0 < F \leq 1.5$, $q = 2$ ou $q = 4$ e $w = 2/3$. Como o algoritmo de runge-kutta é relativamente simples, resolvi deixar a mesma função em todos os arquivos para você rodar diretamente, sem erro. Qualquer dúvida é so me enviar um e-mail[1]. Bom proveito!

Comentario

A energia do oscilador utilizamos a expressão

$$E = ml^2\dot{\theta}^2 + mgl(1 - cos(\theta)) \tag{3}$$

Para o plot da energia do oscilador, divimos tudo por ml^2 e ajustamos $\frac{g}{l} = 1$, daí

$$E' = \dot{\theta}^2 + (1 - cos(\theta)) \tag{4}$$

[1] joao.henrique.santana@usp.br

Código

Para você ler código e os comentários de uma forma mais confortável, resolvi colocar nesse manual. Para não torna repetitivo só comentei a rungekutta no primeiro código.

Code 1: Espaço de Fase

```
import matplotlib.pyplot as plt
from matplotlib import rc
import numpy as np
import math

rc('font',**{'family':'serif','serif':['Computer Modern Roman']}) #fonte para os graficos
rc('text', usetex=True)

def g(t,y,z,F,q,w):
    return -z/q -math.sin(y) + F*math.sin(w*t)                      #funcao do problema

def rungekutta(y0,z0,F,q,w):
    y, z, k1y, k1z, k2y, k2z, k3y, k3z, k4y, k4z = ([0 for i in range(N+1)] for i in range(10))
    t = np.arange(0,N+1,h).tolist()        #lista tempo com intervalo h
    y[0] = y0                              #condicoes inicais
    z[0] = z0
    for i in range(N):
        k1y = h*z[i]
        k1z = h*g(t[i],y[i],z[i],F,q,w)                  #calculando os k
        k2y = h*(z[i] + k1z/2)
        k2z = h*g(t[i] + h/2, y[i] + k1y/2, z[i] + k1z/2,F,q,w)
        k3y = h*(z[i] + k2z/2)
        k3z = h*g(t[i] + h/2, y[i] + k2y/2, z[i] + k2z/2,F,q,w)
        k4y = h*(z[i] + k3z)
        k4z = h*g(t[i] + h, y[i] + k3y, z[i] + k3z,F,q,w)
        y[i+1] = y[i] + (k1y + 2*k2y + 2*k3y + k4y)/6        #rungekutta para y
        z[i+1] = z[i] + (k1z + 2*k2z + 2*k3z + k4z)/6        #rungekutta para z
        if y[i+1] > math.pi:
            y[i+1] = y[i+1] - 2*math.pi #condicao para manter o valor de theta entre -pi e pi
        if y[i+1] < -math.pi:
            y[i+1] = y[i+1] + 2*math.pi
    return t,y,z


for F in [0.2, 0.9, 1.07, 1.20, 1.35, 1.45, 1.47, 1.5]:          #looping para cada forca
    h, N =0.01, 200000
    t,y,z=rungekutta(0,0,F,2,2/3)                                  #runge-kutta
    plt.plot(y[100000:],z[100000:],'darkblue',label='$F=%.2f$' %F,linewidth =0.7)#plot
    plt.grid(linestyle='dotted',color='black')
    plt.xlabel(r'$\theta$',fontsize=18)
    plt.xticks(fontsize=14)
    plt.yticks(fontsize=14)
    plt.ylabel(r'$\dot{\theta}$',fontsize=18,rotation=0)
    plt.title('Espa\c{c}o de Fase',fontsize=20)
    plt.legend(fontsize=14)
    plt.savefig('espaco_fase_sem_transiente%.2f.png'%F)
    plt.show()
```

Code 2: Mapa de Poincaré

```
import matplotlib.pyplot as plt
from matplotlib import rc
import numpy as np
import math

rc('font',**{'family':'serif','serif':['Computer Modern Roman']})
rc('text', usetex=True)

def g(t,y,z,F,q,w):
```

```
    return -z/q -math.sin(y) + F*math.sin(w*t)

def rungekutta(y0,z0,F,q,w):
    y, z, k1y, k1z, k2y, k2z, k3y, k3z, k4y, k4z = ([0 for i in range(N+1)] for i in range(10))
    t = np.arange(0,N+1,h).tolist()
    y[0] = y0
    z[0] = z0
    for i in range(N):
        k1y = h*z[i]
        k1z = h*g(t[i],y[i],z[i],F,q,w)
        k2y = h*(z[i] + k1z/2)
        k2z = h*g(t[i] + h/2, y[i] + k1y/2, z[i] + k1z/2,F,q,w)
        k3y = h*(z[i] + k2z/2)
        k3z = h*g(t[i] + h/2, y[i] + k2y/2, z[i] + k2z/2,F,q,w)
        k4y = h*(z[i] + k3z)
        k4z = h*g(t[i] + h, y[i] + k3y, z[i] + k3z,F,q,w)
        y[i+1] = y[i] + (k1y + 2*k2y + 2*k3y + k4y)/6          # rungekutta para y
        z[i+1] = z[i] + (k1z + 2*k2z + 2*k3z + k4z)/6          # rungekutta para z
        if y[i+1] > math.pi:
            y[i+1] = y[i+1] - 2*math.pi
        if y[i+1] < -math.pi:
            y[i+1] = y[i+1] + 2*math.pi
    return t,y,z



for F in [0.2, 0.9, 1.07, 1.20, 1.35, 1.45, 1.47, 1.5]:#looping forca
    q, w, h, N = 2, 2/3, 0.01,100000
    t,y,z=rungekutta(0,0,F,q,w) # evolui o sistema para um estado sem transiente
    a = []                           #lista muda
    b = []                           #lista muda
    for i in range(10000):
        h = 0.01*2*math.pi/w
        N = 100                              # anda 1 periodo
        t,y,z=rungekutta(y[-1],z[-1],F,q,w)
        a.append(y[-1])                          #adicionamos o ultimo elementos na lista a
        b.append(z[-1])                          #adicionamos o ultimo elementos na lista b

    plt.scatter(a,b,label='$F=%.2f$' %F,color='purple',s=0.3)#plot
    plt.grid(linestyle='dotted',color='black')
    plt.xlabel(r'$\theta$',fontsize=18)
    plt.xticks(fontsize=14)
    plt.yticks(fontsize=14)
    plt.ylabel(r'$\dot{\theta}$',fontsize=18,rotation=0)
    plt.title("Mapa de Poincare",fontsize=20)
    plt.legend(fontsize=14)
    plt.savefig('mapa_de_poincare%.2f.pdf'%F,dpi=300)
    plt.show()
```

Code 3: Diagrama de Bifurcação

```
import matplotlib.pyplot as plt
from matplotlib import rc
import numpy as np
import math

rc('font',**{'family':'serif','serif':['Computer Modern Roman']})
rc('text', usetex=True)

def g(t,y,z,F,q,w):
    return -z/q -math.sin(y) + F*math.sin(w*t)

def rungekutta(y0,z0,F,q,w):
    y, z, k1y, k1z, k2y, k2z, k3y, k3z, k4y, k4z = ([0 for i in range(N+1)] for i in range(10))
    t = np.arange(0,N+1,h).tolist()
```

```python
    y[0] = y0
    z[0] = z0
    for i in range(N):
        k1y = h*z[i]
        k1z = h*g(t[i],y[i],z[i],F,q,w)
        k2y = h*(z[i] + k1z/2)
        k2z = h*g(t[i] + h/2, y[i] + k1y/2, z[i] + k1z/2,F,q,w)
        k3y = h*(z[i] + k2z/2)
        k3z = h*g(t[i] + h/2, y[i] + k2y/2, z[i] + k2z/2,F,q,w)
        k4y = h*(z[i] + k3z)
        k4z = h*g(t[i] + h, y[i] + k3y, z[i] + k3z,F,q,w)
        y[i+1] = y[i] + (k1y + 2*k2y + 2*k3y + k4y)/6          # rungekutta para y
        z[i+1] = z[i] + (k1z + 2*k2z + 2*k3z + k4z)/6          # rungekutta para y'
        if y[i+1] > math.pi:
            y[i+1] = y[i+1] - 2*math.pi
        if y[i+1] < -math.pi:
            y[i+1] = y[i+1] + 2*math.pi
    return t,y,z

Forca = np.arange(1.35,1.5,0.0001).tolist()
for F in Forca:
    f=[]
    a=[]
    q, w, h, N = 2, 2/3, 0.01,100000
    t,y,z=rungekutta(0,0,F,q,w) # evolui o sistem para um estado sem transiente
    for i in range(1000):
        h = 0.01*2*math.pi/w
        N = 100                          # anda 1 periodo
        t,y,z=rungekutta(y[-1],z[-1],F,q,w)
        a.append(y[-1])
        f.append(F)

    plt.scatter(f,a,color='purple',s=1)

plt.grid(linestyle='dotted',color='black')
plt.xlabel(r'$F$',fontsize=18)
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.ylabel(r'$\theta$',fontsize=18,rotation=0)
plt.title("Diagrama de Bifurca\c{c}\~{a}o",fontsize=20)
plt.savefig('diagrama_bifurcacao.png')
plt.show()
```

Code 4: Exponente de Lyapunov

```python
import matplotlib.pyplot as plt
from matplotlib import rc
import numpy as np
import math
from scipy.optimize import curve_fit  #modulo scipy

rc('font',**{'family':'serif','serif':['Computer Modern Roman']})
rc('text', usetex=True)

def g(t,y,z,F,q,w):
    return -z/q -math.sin(y) + F*math.sin(w*t)

def rungekutta(y0,z0,F,q,w):
    y, z, k1y, k1z, k2y, k2z, k3y, k3z, k4y, k4z = ([0 for i in range(N+1)] for i in range(10))
    t = np.arange(0,N+1,h).tolist()
    y[0] = y0
    z[0] = z0
    for i in range(N):
        k1y = h*z[i]
        k1z = h*g(t[i],y[i],z[i],F,q,w)
        k2y = h*(z[i] + k1z/2)
```

```
        k2z = h*g(t[i] + h/2, y[i] + k1y/2, z[i] + k1z/2,F,q,w)
        k3y = h*(z[i] + k2z/2)
        k3z = h*g(t[i] + h/2, y[i] + k2y/2, z[i] + k2z/2,F,q,w)
        k4y = h*(z[i] + k3z)
        k4z = h*g(t[i] + h, y[i] + k3y, z[i] + k3z,F,q,w)
        y[i+1] = y[i] + (k1y + 2*k2y + 2*k3y + k4y)/6          # rungekutta para y
        z[i+1] = z[i] + (k1z + 2*k2z + 2*k3z + k4z)/6          # rungekutta para \

    return t,y,z        #nao temos mais a restricao de -pi a pi !!!!


q, w, h, N= 2, 2/3, 0.01, 200000
t,y,z=rungekutta(0,2,1.2,q,w)      #condicao inicial 0 e 2
theta1=y
t,y,z=rungekutta(0.001,2.001,1.2,q,w) #condicao inicial 0.0001 e 2.0001
theta2 = y
theta=abs(np.array(theta2)-np.array(theta1)) #grandeza theta
t= np.array(t)
T = 105000               #tempo
def func(x, a, b, c):
    return a * np.exp(-b * x) + c    #funcao do ajuste

popt, pcov = curve_fit(func,t[0:T],theta[0:T])  #ajuste

#plot
plt.plot(t[0:T],func(t[0:T],*popt),'r--',label='ajuste: $a=%5.3f$, $|\lambda_{1}|=%5.3f$, $c
    =%5.3f$' % tuple(popt) )

t=t.tolist()                  #t vira uma lista
theta=theta.tolist()          #theta vira uma lista

plt.plot(t[0:T],theta[0:T],color='teal',label='F=1.2',linewidth=2) #segundo plot
plt.grid(linestyle='dotted',color='black')
plt.xlabel(r'$t(s)$',fontsize=18)
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.ylabel(r'$\Delta\theta$',fontsize=18,rotation=0)
plt.yscale('log')
plt.title("Expoente de Lyapunov",fontsize=20)
plt.legend(fontsize=14)
plt.savefig('expoente.pdf',dpi=300)
plt.show()
```

Code 5: Energia do Oscilador

```
import matplotlib.pyplot as plt
from matplotlib import rc
import numpy as np
import math
from scipy.optimize import curve_fit


rc('font',**{'family':'serif','serif':['Computer Modern Roman']})
rc('text', usetex=True)

def g(t,y,z,F,q,w):
    return -z/q -math.sin(y) + F*math.sin(w*t)

def rungekutta(y0,z0,F,q,w):
    y, z, k1y, k1z, k2y, k2z, k3y, k3z, k4y, k4z = ([0 for i in range(N+1)] for i in range
        (10))
    t = np.arange(0,N+1,h).tolist()
    y[0] = y0
    z[0] = z0
    for i in range(N):
        k1y = h*z[i]
        k1z = h*g(t[i],y[i],z[i],F,q,w)
```

```python
        k2y = h*(z[i] + k1z/2)
        k2z = h*g(t[i] + h/2, y[i] + k1y/2, z[i] + k1z/2,F,q,w)
        k3y = h*(z[i] + k2z/2)
        k3z = h*g(t[i] + h/2, y[i] + k2y/2, z[i] + k2z/2,F,q,w)
        k4y = h*(z[i] + k3z)
        k4z = h*g(t[i] + h, y[i] + k3y, z[i] + k3z,F,q,w)
        y[i+1] = y[i] + (k1y + 2*k2y + 2*k3y + k4y)/6          # rungekutta para y
        z[i+1] = z[i] + (k1z + 2*k2z + 2*k3z + k4z)/6          # rungekutta para z
        if y[i+1] > math.pi:
            y[i+1] = y[i+1] - 2*math.pi
        if y[i+1] < -math.pi:
            y[i+1] = y[i+1] + 2*math.pi
    return t,y,z

for F in [0,0.1,0.2,0.5, 1.2,1.45,1.47, 1.5]: #looping forca
    q, w, h, N = 10, 2/3, 0.01,100000
    t,y,z=rungekutta(-0.5,1,F,q,w)      #runge-kutta ara condicao inicial -0.5 e 1
    z=(np.array(z)**2)/2                #energia cinetica
    y = 1-np.cos(np.array(y))           #energia potencial
    E = z + y                           #energia total
    t= np.array(t)
    if F==0:
        def func(x, a, b, c):       #ajusta apenas para F = 0
            return a * np.exp(-b * x) + c
        popt, pcov = curve_fit(func,t[0:10000],E[0:10000])
        plt.plot(t[0:10000],func(t[0:10000],*popt),'b--',label='ajuste: $a=%5.3f$, $\gamma=%5.3f$, $c=%5.3f$' % tuple(popt) )
        pass

    t = t.tolist()
    E = E.tolist()
                                                          #segundo plot
    plt.plot(t[0:10000],E[0:10000],label='$F=%.2f$' %F,color='indianred',linewidth =2)
    plt.grid(linestyle='dotted',color='black')
    plt.xlabel(r'$t$(s)',fontsize=18)
    plt.xticks(fontsize=14)
    plt.yticks(fontsize=14)
    plt.ylabel(r"$E'$ (dimens\~{a}o apropriada)",fontsize=18)
    plt.title("Energia do Oscilador",fontsize=20)
    plt.legend(fontsize=14)
    plt.savefig('energia_oscilador%.2f.pdf'%F,dpi=300)
    plt.show()
```