

Métodos Computacionais 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 arquivos *.py* e dar o seguinte comando `python3 nome_arquivo.py`. Se voce estiver num ambiente de programação é so abrir o arquivo *.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

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

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

$$\begin{cases} w = \Omega_d \\ y = \theta \\ z = \dot{\theta} \end{cases} \quad (2)$$

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-mail¹. Bom proveito!

Comentario

A energia do oscilador utilizamos a expressão

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

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

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

¹joao.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

```
1 import matplotlib.pyplot as plt
2 from matplotlib import rc
3 import numpy as np
4 import math
5
6 rc('font',**{'family':'serif','serif':['Computer Modern Roman']}) #fonte para os graficos
7 rc('text', usetex=True)
8
9 def g(t,y,z,F,q,w):
10     return -z/q -math.sin(y) + F*math.sin(w*t) #funcao do problema
11
12 def rungekutta(y0,z0,F,q,w):
13     y, z, k1y, k1z, k2y, k2z, k3y, k3z, k4y, k4z = ([0 for i in range(N+1)] for i in range
14         (10))
15     t = np.arange(0,N+1,h).tolist() #lista tempo com intervalo h
16     y[0] = y0 #condicoes iniciais
17     z[0] = z0
18     for i in range(N):
19         k1y = h*z[i]
20         k1z = h*g(t[i],y[i],z[i],F,q,w) #calculando os k
21         k2y = h*(z[i] + k1z/2)
22         k2z = h*g(t[i] + h/2, y[i] + k1y/2, z[i] + k1z/2,F,q,w)
23         k3y = h*(z[i] + k2z/2)
24         k3z = h*g(t[i] + h/2, y[i] + k2y/2, z[i] + k2z/2,F,q,w)
25         k4y = h*(z[i] + k3z)
26         k4z = h*g(t[i] + h, y[i] + k3y, z[i] + k3z,F,q,w)
27         y[i+1] = y[i] + (k1y + 2*k2y + 2*k3y + k4y)/6 #rungekutta para y
28         z[i+1] = z[i] + (k1z + 2*k2z + 2*k3z + k4z)/6 #rungekutta para z
29         if y[i+1] > math.pi:
30             y[i+1] = y[i+1] - 2*math.pi #condicao para manter o valor de theta entre -pi e
31             #pi
32         if y[i+1] < -math.pi:
33             y[i+1] = y[i+1] + 2*math.pi
34     return t,y,z
35
36 for F in [0.2, 0.9, 1.07, 1.20, 1.35, 1.45, 1.47, 1.5]: #looping para cada forca
37     h, N = 0.01, 200000
38     t,y,z=rungekutta(0,0,F,2,2/3) #runge-kutta
39     plt.plot(y[100000:],z[100000:], 'darkblue', label='$F=%.2f$' %F,linewidth =0.7)#plot
40     plt.grid(linestyle='dotted',color='black')
41     plt.xlabel(r'$\theta$', fontsize=18)
42     plt.xticks(fontsize=14)
43     plt.yticks(fontsize=14)
44     plt.ylabel(r'$\dot{\theta}$', fontsize=18,rotation=0)
45     plt.title('Espaço de Fase', fontsize=20)
46     plt.legend(fontsize=14)
47     plt.savefig('espaco_fase_sem_transiente%.2f.png'%F)
48     plt.show()
```

Code 2: Mapa de Poincaré

```
1 import matplotlib.pyplot as plt
2 from matplotlib import rc
3 import numpy as np
4 import math
5
6 rc('font',**{'family':'serif','serif':['Computer Modern Roman']})
7 rc('text', usetex=True)
8
9 def g(t,y,z,F,q,w):
```

```

10     return -z/q -math.sin(y) + F*math.sin(w*t)
11
12 def rungekutta(y0,z0,F,q,w):
13     y, z, k1y, k1z, k2y, k2z, k3y, k3z, k4y, k4z = ([0 for i in range(N+1)] for i in range
14         (10))
15     t = np.arange(0,N+1,h).tolist()
16     y[0] = y0
17     z[0] = z0
18     for i in range(N):
19         k1y = h*z[i]
20         k1z = h*g(t[i],y[i],z[i],F,q,w)
21         k2y = h*(z[i] + k1z/2)
22         k2z = h*g(t[i] + h/2, y[i] + k1y/2, z[i] + k1z/2,F,q,w)
23         k3y = h*(z[i] + k2z/2)
24         k3z = h*g(t[i] + h/2, y[i] + k2y/2, z[i] + k2z/2,F,q,w)
25         k4y = h*(z[i] + k3z)
26         k4z = h*g(t[i] + h, y[i] + k3y, z[i] + k3z,F,q,w)
27         y[i+1] = y[i] + (k1y + 2*k2y + 2*k3y + k4y)/6 # rungekutta para y
28         z[i+1] = z[i] + (k1z + 2*k2z + 2*k3z + k4z)/6 # rungekutta para z
29         if y[i+1] > math.pi:
30             y[i+1] = y[i+1] - 2*math.pi
31         if y[i+1] < -math.pi:
32             y[i+1] = y[i+1] + 2*math.pi
33     return t,y,z
34
35
36 for F in [0.2, 0.9, 1.07, 1.20, 1.35, 1.45, 1.47, 1.5]:#looping forca
37     q, w, h, N = 2, 2/3, 0.01,100000
38     t,y,z=rungekutta(0,0,F,q,w) # evolui o sistema para um estado sem transiente
39     a = [] #lista muda
40     b = [] #lista muda
41     for i in range(10000):
42         h = 0.01*2*math.pi/w
43         N = 100 # anda 1 periodo
44         t,y,z=rungekutta(y[-1],z[-1],F,q,w)
45         a.append(y[-1]) #adicionamos o ultimo elementos na lista a
46         b.append(z[-1]) #adicionamos o ultimo elementos na lista b
47
48     plt.scatter(a,b,label='$F=%.2f$' %F,color='purple',s=0.3)#plot
49     plt.grid(linestyle='dotted',color='black')
50     plt.xlabel(r'$\theta$', fontsize=18)
51     plt.xticks(fontsize=14)
52     plt.yticks(fontsize=14)
53     plt.ylabel(r'$\dot{\theta}$', fontsize=18,rotation=0)
54     plt.title("Mapa de Poincare", fontsize=20)
55     plt.legend(fontsize=14)
56     plt.savefig('mapa_de_poincare%.2f.pdf'%F,dpi=300)
57     plt.show()

```

Code 3: Diagrama de Bifurcação

```

1 import matplotlib.pyplot as plt
2 from matplotlib import rc
3 import numpy as np
4 import math
5
6 rc('font',**{'family':'serif','serif':['Computer Modern Roman']})
7 rc('text', usetex=True)
8
9 def g(t,y,z,F,q,w):
10     return -z/q -math.sin(y) + F*math.sin(w*t)
11
12 def rungekutta(y0,z0,F,q,w):
13     y, z, k1y, k1z, k2y, k2z, k3y, k3z, k4y, k4z = ([0 for i in range(N+1)] for i in range
14         (10))
15     t = np.arange(0,N+1,h).tolist()

```

```

15     y[0] = y0
16     z[0] = z0
17     for i in range(N):
18         k1y = h*z[i]
19         k1z = h*g(t[i], y[i], z[i], F, q, w)
20         k2y = h*(z[i] + k1z/2)
21         k2z = h*g(t[i] + h/2, y[i] + k1y/2, z[i] + k1z/2, F, q, w)
22         k3y = h*(z[i] + k2z/2)
23         k3z = h*g(t[i] + h/2, y[i] + k2y/2, z[i] + k2z/2, F, q, w)
24         k4y = h*(z[i] + k3z)
25         k4z = h*g(t[i] + h, y[i] + k3y, z[i] + k3z, F, q, w)
26         y[i+1] = y[i] + (k1y + 2*k2y + 2*k3y + k4y)/6 # rungekutta para y
27         z[i+1] = z[i] + (k1z + 2*k2z + 2*k3z + k4z)/6 # rungekutta para z
28         if y[i+1] > math.pi:
29             y[i+1] = y[i+1] - 2*math.pi
30         if y[i+1] < -math.pi:
31             y[i+1] = y[i+1] + 2*math.pi
32     return t, y, z
33
34 Forca = np.arange(1.35, 1.5, 0.0001).tolist()
35 for F in Forca:
36     f=[]
37     a=[]
38     q, w, h, N = 2, 2/3, 0.01, 100000
39     t, y, z = rungekutta(0, 0, F, q, w) # evolui o sistema para um estado sem transiente
40     for i in range(1000):
41         h = 0.01*2*math.pi/w
42         N = 100 # anda 1 periodo
43         t, y, z = rungekutta(y[-1], z[-1], F, q, w)
44         a.append(y[-1])
45         f.append(F)
46
47     plt.scatter(f, a, color='purple', s=1)
48
49 plt.grid(linestyle='dotted', color='black')
50 plt.xlabel(r'$F$', fontsize=18)
51 plt.xticks(fontsize=14)
52 plt.yticks(fontsize=14)
53 plt.ylabel(r'$\theta$', fontsize=18, rotation=0)
54 plt.title("Diagrama de Bifurca\c{c}\~{a}o", fontsize=20)
55 plt.savefig('diagrama_bifurcacao.png')
56 plt.show()

```

Code 4: Exponente de Lyapunov

```

1 import matplotlib.pyplot as plt
2 from matplotlib import rc
3 import numpy as np
4 import math
5 from scipy.optimize import curve_fit # modulo scipy
6
7 rc('font', **{'family': 'serif', 'serif': ['Computer Modern Roman']})
8 rc('text', usetex=True)
9
10 def g(t, y, z, F, q, w):
11     return -z/q - math.sin(y) + F*math.sin(w*t)
12
13 def rungekutta(y0, z0, F, q, w):
14     y, z, k1y, k1z, k2y, k2z, k3y, k3z, k4y, k4z = ([0 for i in range(N+1)] for i in range
15     (10))
16     t = np.arange(0, N+1, h).tolist()
17     y[0] = y0
18     z[0] = z0
19     for i in range(N):
20         k1y = h*z[i]
21         k1z = h*g(t[i], y[i], z[i], F, q, w)
22         k2y = h*(z[i] + k1z/2)

```

```

22     k2z = h*g(t[i] + h/2, y[i] + k1y/2, z[i] + k1z/2,F,q,w)
23     k3y = h*(z[i] + k2z/2)
24     k3z = h*g(t[i] + h/2, y[i] + k2y/2, z[i] + k2z/2,F,q,w)
25     k4y = h*(z[i] + k3z)
26     k4z = h*g(t[i] + h, y[i] + k3y, z[i] + k3z,F,q,w)
27     y[i+1] = y[i] + (k1y + 2*k2y + 2*k3y + k4y)/6      # rungekutta para y
28     z[i+1] = z[i] + (k1z + 2*k2z + 2*k3z + k4z)/6      # rungekutta para z
29
30     return t,y,z      #nao temos mais a restricao de -pi a pi !!!!
31
32
33 q, w, h, N= 2, 2/3, 0.01, 200000
34 t,y,z=rungekutta(0,2,1.2,q,w)      #condicao inicial 0 e 2
35 theta1=y
36 t,y,z=rungekutta(0.001,2.001,1.2,q,w) #condicao inicial 0.0001 e 2.0001
37 theta2 = y
38 theta=abs(np.array(theta2)-np.array(theta1)) #grandeza theta
39 t= np.array(t)
40 T = 105000      #tempo
41 def func(x, a, b, c):
42     return a * np.exp(-b * x) + c      #funcao do ajuste
43
44 popt, pcov = curve_fit(func,t[0:T],theta[0:T]) #ajuste
45
46 #plot
47 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) )
48
49 t=t.tolist()      #t vira uma lista
50 theta=theta.tolist()      #theta vira uma lista
51
52 plt.plot(t[0:T],theta[0:T],color='teal',label='F=1.2',linewidth=2) #segundo plot
53 plt.grid(linestyle='dotted',color='black')
54 plt.xlabel(r'$t(s)$',fontsize=18)
55 plt.xticks(fontsize=14)
56 plt.yticks(fontsize=14)
57 plt.ylabel(r'$\Delta\theta$',fontsize=18,rotation=0)
58 plt.yscale('log')
59 plt.title("Expoente de Lyapunov",fontsize=20)
60 plt.legend(fontsize=14)
61 plt.savefig('expoente.pdf',dpi=300)
62 plt.show()

```

Code 5: Energia do Oscilador

```

1  import matplotlib.pyplot as plt
2  from matplotlib import rc
3  import numpy as np
4  import math
5  from scipy.optimize import curve_fit
6
7
8  rc('font',**{'family':'serif','serif':['Computer Modern Roman']})
9  rc('text', usetex=True)
10
11 def g(t,y,z,F,q,w):
12     return -z/q -math.sin(y) + F*math.sin(w*t)
13
14 def rungekutta(y0,z0,F,q,w):
15     y, z, k1y, k1z, k2y, k2z, k3y, k3z, k4y, k4z = ([0 for i in range(N+1)] for i in range
16     (10))
17     t = np.arange(0,N+1,h).tolist()
18     y[0] = y0
19     z[0] = z0
20     for i in range(N):
21         k1y = h*z[i]
22         k1z = h*g(t[i],y[i],z[i],F,q,w)

```

```

22     k2y = h*(z[i] + k1z/2)
23     k2z = h*(t[i] + h/2, y[i] + k1y/2, z[i] + k1z/2,F,q,w)
24     k3y = h*(z[i] + k2z/2)
25     k3z = h*(t[i] + h/2, y[i] + k2y/2, z[i] + k2z/2,F,q,w)
26     k4y = h*(z[i] + k3z)
27     k4z = h*(t[i] + h, y[i] + k3y, z[i] + k3z,F,q,w)
28     y[i+1] = y[i] + (k1y + 2*k2y + 2*k3y + k4y)/6           # rungekutta para y
29     z[i+1] = z[i] + (k1z + 2*k2z + 2*k3z + k4z)/6           # rungekutta para z
30     if y[i+1] > math.pi:
31         y[i+1] = y[i+1] - 2*math.pi
32     if y[i+1] < -math.pi:
33         y[i+1] = y[i+1] + 2*math.pi
34     return t,y,z
35
36 for F in [0,0.1,0.2,0.5, 1.2,1.45,1.47, 1.5]: #looping forca
37     q, w, h, N = 10, 2/3, 0.01,100000
38     t,y,z=rungekutta(-0.5,1,F,q,w)           #runge-kutta ara condicao inicial -0.5 e 1
39     z=(np.array(z)**2)/2                     #energia cinetica
40     y = 1-np.cos(np.array(y))               #energia potencial
41     E = z + y                               #energia total
42     t= np.array(t)
43     if F==0:
44         def func(x, a, b, c):               #ajusta apenas para F = 0
45             return a * np.exp(-b * x) + c
46         popt, pcov = curve_fit(func,t[0:10000],E[0:10000])
47         plt.plot(t[0:10000],func(t[0:10000],*popt),'b--',label='ajuste: $a=%5.3f$, $\gamma$
48             =%5.3f$, $c=%5.3f$' % tuple(popt) )
49     pass
50     t = t.tolist()
51     E = E.tolist()
52
53     #segundo plot
54     plt.plot(t[0:10000],E[0:10000],label='$F=%.2f$' %F,color='indianred',linewidth =2)
55     plt.grid(linestyle='dotted',color='black')
56     plt.xlabel(r'$t$(s)',fontsize=18)
57     plt.xticks(fontsize=14)
58     plt.yticks(fontsize=14)
59     plt.ylabel(r'$E$(dimens~{a}o apropriada)",fontsize=18)
60     plt.title("Energia do Oscilador",fontsize=20)
61     plt.legend(fontsize=14)
62     plt.savefig('energia_oscilador%.2f.pdf'%F,dpi=300)
63     plt.show()

```
