Análise do Experimento das Oscilações Amortecidas

Universidade Federal de Campina Grande
Disciplina: Instrumentação Científica
Professor Adriano de A. Batista

Sony Gonzaga de Melo Neto

9 de abril de 2022

1 INTRODUÇÃO

Neste relatório será apresentada uma análise dos dados obtidos pelas oscilações de um Ressonador Mecânico. O objetivo é a determinação da frequência de oscilação do mesmo e seus fatores de decaimento e de qualidade, visto que seu comportamento é de um oscilador amortecido.

O Ressonador Mecânico é um dispositivo eletrônico capaz de gerar sinais elétricos oscilatórios. O mesmo, ao ser posto numa corrente elétrica, retorna-a oscilando com uma certa frequência. Podem ser construídos com um piezoelétrico que vibra ao receber a tensão, como também por um volume de ar confinado que ressoa enquanto está sobre a tensão, nesse caso, é chamado de Ressonador de Helmholtz. Suas aplicações, no geral, são em circuitos que requerem sinais periódicos.

2 DESCRIÇÃO DO EXPERIMENTO

O experimento foi realizado simplesmente aplicando uma tensão no Ressonador e o deixando oscilar por cerca de 10 segundos, enquanto sua amplitude diminuía. Os sinais retornados pelo equipamento foram registrados no computador, num arquivo .csv (Comma-separated values), e plotados num gráfico como a posição do elemento oscilatório no interior do Ressonador em função do tempo. O gráfico possui característica períodico de decaimento exponencial, podendo ser verificada sua semelhança com o oscilador subamortecido.

3 O OSCILADOR AMORTECIDO

O sistema de oscilação amortecida acontece quando há uma força dissipativa, proporcional à velocidade do objeto, que retira energia do sistema oscilatório a medida que o tempo passa. Seja pensado um sistema massa-mola, forças dissipativas possíveis seriam a resistência com o ar e o atrito de seus componentes mecânicos.

A equação de movimento do oscilador amortecido para a posição é geralmente escrita como:

$$m\ddot{x} + \beta\dot{x} + kx = 0. \tag{1}$$

Dividindo todos os termos por m, sabendo que $\omega_0^2 = \frac{k}{m}$, com ω_0 a frequência ângular natural do sistema, e chamando $\gamma = \frac{\beta}{m}$ de fator de decaimento (pois está relacionado com a força dissipativa), a equação se torna:

$$\ddot{x} + \gamma\dot{x} + \omega_0^2 x = 0 \tag{2}$$

e sua solução é:

$$x(t) = Ae^{(-\gamma/2)t}\cos(\omega t + \phi), \tag{3}$$

com A sendo o envelope de $x(t)$ e ϕ uma constante que depende das condições iniciais do sistema.

A frequência angular ω difere da frequência angular natural pela seguinte equação:

$$\omega = \sqrt{\omega_0^2 - (\frac{\gamma}{2})^2}. \tag{4}$$

4 ANÁLISE DOS DADOS

A série temporal dos dados foi plotada com um programa em Python, com a ajuda da biblioteca MatPlotLib. O script também executa a Transformada de Fourier e mostra seu gráfico para a análise das frequências que compõem a figura de oscilação.

Abaixo, as referidas imagens para os dados brutos:

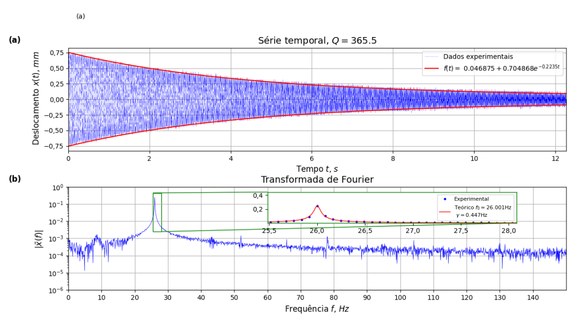

Figura 1: Dados Brutos

O programa também retorna uma equação para o envelope exponencial $f(t)$ da oscilação amortecida, o fator de qualidade Q, a frequência natural de oscilação f_0 e o fator de decaimento γ, extraídos da Transformada de Fourier. Esses dados são obtidos ao se fazer uma aproximação pelos dois lados para γ:

Sabendo que a T.F., nesse caso, é equacionada como $f(\omega) = \frac{1}{\sqrt{(\omega^2-\omega_0^2)+\gamma^2\omega^2}}$, toma-se γ_1 e γ_2, tal que $\gamma_1 < \gamma < \gamma_2$ e aproxima-se γ_1 de γ_2, calculando as raízes da equação $g(\gamma) = f(\gamma) - f(\gamma_1) + f(\gamma_2)$.

Entretanto, apesar da T.F. expressar os parâmetros procurados, a partir dos dados brutos, o envelope não reflete com precisão a real amplitude em função do tempo das oscilações.

Para melhorá-la, faz-se o uso de um Filtro de Média Móvel, utilizando um programa escrito com a biblioteca Pandas, que atenua os ruídos e promove um série temporal mais limpa. O Filtro funciona seguindo a seguinte equação, para um certo sinal t e uma janela n:

$$y[t] = \frac{1}{n}\sum_{k=0}^{n-1} x[t-k]. \tag{5}$$

Para um n muito pequeno, os ruídos pouco mudarão. Para n muito grande, o gráfico perde sua característica oscilatória, portanto, cria-se um script com a biblioteca Scipy para encontrar o melhor n, determinando qual o menor valor de γ, pois o envelope é uma função do tipo $f(t) = C_1 + C_2 e^{-(\gamma t)/2}$ e o fator de qualidade $Q = \frac{\omega_0}{\gamma}$ será o maior possível.

Verifica-se que a função $\gamma(n)$ cresce para n pequeno e para $n > 512$, portanto, o script encontra o menor valor de maneira decrescente a partir de $n = 512$, até achar um pico de mínimo em $n = 414$, com $\gamma = 0,446897314$. O gráfico da sequência temporal ao passar pelo Filtro de Média Móvel, fica, então:

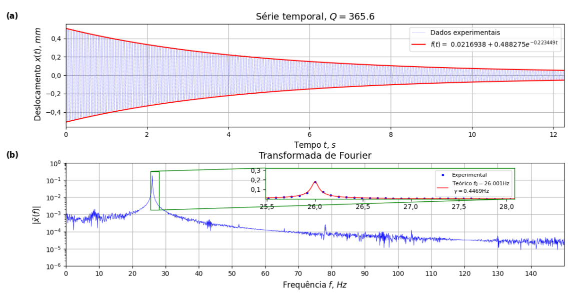

Figura 2: Dados filtrados com Filtro de Média Móvel

Por outro lado, como o Ressonador não desenvolve um movimento oscilatório ideal, este é aproximado e sua frequência encontrada pela T.F. não é satisfatória. Vê-se necessário truncar os dados para obter um intervalo da série temporal que melhor se aproxima do oscilador amortecido ideal.

Para a obtenção de um bom valor aproximado da frequência natural, cria-se um script parecido com o anterior, mas que compare a frequência correspondentes a intervalos do final da série para passos decrescentes, ou seja, para truncamentos cada vez maiores. Obteve-se o melhor intervalo $9,32s < t < 10,0s$ para uma frequência natural $f_0 = 26,4721Hz$

5 CONCLUSÃO

O valor da frequência natural extraído da T.F. não ofereceu nenhuma variação considerável com a aplicação do Filtro.

Substituindo os parâmetros obtidos da análise dos gráficos construídos com os dados da série temporal, $\gamma = 0,446897314$ e $f_0 = 26,4721Hz$ na equação 4, com $\omega = 2\pi f$, a frequência das oscilações é $f = \frac{\omega}{2\pi} = 26,4721Hz$. A semelhança com f_0 se deve pelo fator de qualidade ser $Q = \frac{\omega_0}{\gamma} = 372.19 >> 1$.

A equação do gráfico de oscilações é, portanto, aplicando os parâmetros em 3 e o envelope $A = f(t)$,

$$x(t) = (0,0216938 + 0,488275e^{-0,223449t})cos(166,3291t + \phi). \tag{6}$$

O gráfico abaixo mostra como a equação acima se comporta ao ser posta junto com os dados experimentais, tomando $\phi = 1.84rad$:

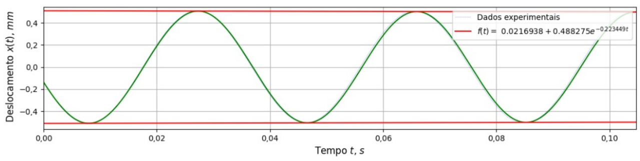

Figura 3: Ajuste da Série Temporal

Também é justo confirmar que, como $\omega_0 > \frac{\gamma}{2}$, o oscilador se encaixa no caso de subamortecido.

Referências

Carlos E. Rufino da Silva. Relatório do experimento do pêndulo físico física não-linear (tef 2018.1), 2018.

Niels Fontes Lima. Experimentos com o ressonador de Helmholtz. http://www.ifba.edu.br/fisica/nfl/fge2/RessonadorDeHelmholtz/ExperimentoRessonadorDeHelmholtz.html.

MatPlotLib. Matplotlib documentation. https://matplotlib.org/3.5.0/index.html.

Carl R. Nave. Damped harmonic oscillator. http://hyperphysics.phy-astr.gsu.edu/hbase/oscda.html.

Numpy. Numpy documentation. https://numpy.org/doc/stable/index.html.

Pandas. Pandas documentation. https://pandas.pydata.org/docs/.

Scipy. Scipy documentation. https://docs.scipy.org/doc/scipy/.

6 APÊNDICE

Programas utilizados:

```
###CÓDIGO PARA FILTRO DE MÉDIA MÓVEL, PLOTAGEM DE GRÁFICO E TRANSFORMADA DE FOURIER###
# -*- coding: utf-8 -*-
#Locale settings
import locale, sys
# Set to pt_BR to get comma decimal separater
locale.setlocale(locale.LC_NUMERIC, 'pt_BR.utf8')
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from mpl_toolkits.axes_grid1.inset_locator import zoomed_inset_axes
from mpl_toolkits.axes_grid1.inset_locator import mark_inset
from mpl_toolkits.axes_grid1.inset_locator import InsetPosition
from scipy.optimize import least_squares
import scipy.fftpack

##Função para o Algoritmo do Melhor Valor de n da Janela do Filtro Móvel
def freqop(s):
    df = pd.read_csv(sys.argv[1], sep =',', names=['t', 'theta'], header=2)#Leitura dos Dados para DataFrame
    df['Média móvel'] = df['theta'].rolling(s).mean()#Filtro Móvel
    dft = df['t']#DataFrame dos Valores de Tempos
    dfm = pd.merge(dft, df['Média móvel'].dropna(), right_index=True, left_index=True)#DataFrame da Média Móvel com o Tempo
    arq = sys.argv[1].replace('.csv','') + 'Filtrado.csv'#Definindo Nome do Arquivo
    dfm.to_csv(arq, index=False)#Salvando Arquivo
   #print(dfm)
    plt.rcParams['axes.formatter.use_locale'] = True
#     ##################################################
# Leitura de dados
    def lerDados(arquivo):
        dados = arquivo
        dataset = np.loadtxt(dados, delimiter=',', skiprows=2)
        signal = dataset [:, 1]
        time = dataset[:,0]
        time = time - time[0]
        return[time, signal]

    nPer = 1704 #1491
    nPoints = 180*nPer
    time, signal = lerDados(str(sys.argv[1].replace('.csv','') +'Filtrado.csv'))
    time = time[-nPoints:]
    time = time-time[0]
    signal = signal[-nPoints:]
    avg = np.average(signal[4*int(len(signal)/5):])
# Plota grafico
# plot data
    fig=plt.figure(figsize=(8, 10))
    plt.subplots_adjust(hspace=0.35)
    ax1=fig.add_subplot(211)
    ax1.plot(time, signal-avg, 'b-.', linewidth=0.2, markersize=0.2, label= "Dados experimentais")
#legenda = u"$f(t)=$ $%g+%g e^{-%g t}$" % (C,A,G)
    t_max = time[-1]
    y_max = np.amax(signal-avg)
#ax1.axhline(y=y_max, ls='--')
    ax1.set_xlim(0, t_max)
    ax1.set_ylim(-1.1*y_max, 1.1*y_max)
    ax1.set_xlabel("Tempo $t$, $s$", fontsize=12)
    ax1.set_ylabel(" Deslocamento $x(t)$, $mm$", fontsize=12)
    ax1.text(0.2, 1.3, '(a)')

    ax1.grid()

# second subplot: the Fourier transform
#     ##################################################
# Transformada de Fourier
    def FFT(signal, time):
        N_fft = len(signal)
        fft = 2*np.abs(scipy.fftpack.fft(signal))/N_fft
        dt = time[1]-time[0]
        freqs = scipy.fftpack.fftfreq(signal.size, dt)
        return [freqs, fft]
#     ##################################################
# taxa de amostragem para fft
    sampRate = 1
    time = time[::sampRate]
    signal = signal[::sampRate]
# plotar serie temporal
    ax2=fig.add_subplot(212)
# Transformada de Fourier
    freqs, FFT = FFT(signal, time)
    N_fft = len(freqs)
    fft_max = np.amax(FFT[:N_fft//2])
    n_max = np.argmax(FFT[:N_fft//2])
    f_max = n_max*(freqs[1]-freqs[0])
    #print ("f_max = ", f_max, freqs[1]-freqs[0])
    ax2.set_yscale('log')
```

```python
    ax2.set_xticks(np.arange(0., 150., 10.))
    ax2.set_title(u'Transformada de Fourier', fontsize=14)
    ax2.set_xlim(0., 150.)
    ax2.set_ylim([10**-6, 10**0])
    ax2.plot(freqs[:N_fft//2], FFT[:N_fft//2], 'b-', linewidth = 0.5)
    ax2.grid()
#plt.legend(numpoints = '1', loc = "upper left")
    ax2.set_xlabel(u"Frequência $f$, $Hz$", fontsize=12)
    ax2.set_ylabel(r"$|\tilde x(f)|$ ", fontsize=12)
    ax1.text(-0.12, 1.05, '(a)', transform=ax1.transAxes, size=12, weight='bold')
    ax2.text(-0.12, 1.05, '(b)', transform=ax2.transAxes, size=12, weight='bold')
# inset of the peak
# previsão teórica
    f_0 = f_max
#f_0 = 10.5832
    om_0 = 2*np.pi*f_0
    freqInt = np.linspace(0.90*f_0, 1.1*f_0, 1024)
    om = 2*np.pi*freqInt
# sub region of the original image
    x1, x2, y1, y2 = 0.98*f_0, 1.08*f_0, 0.01*fft_max, 1.8*fft_max
#x1, x2, y1, y2 = 10.45, 10.69, 0.01, 0.33
    axins = zoomed_inset_axes(ax2, zoom=4, loc=4) # zoom = 6
    ip = InsetPosition(ax2, [.4, .65, .5, .3])
    axins.set_axes_locator(ip)
    axins.set_xlim(x1, x2)
    axins.set_ylim(y1, y2)
    for axis in ['top','bottom','left','right']:
        axins.spines[axis].set_linewidth(1)
        axins.spines[axis].set_color('g')
    mark_inset(ax2, axins, loc1=2, loc2=4, fc="none", lw=1, ec="g")
    xdata = freqs[n_max-10:n_max+10]
    ydata = FFT[n_max-10:n_max+10]
#    ############################################

    def yMod(gamma, om):
      ganho = 1.0/np.sqrt((om**2-om_0**2)**2+(gamma*om)**2)
      ganho_max = np.amax(ganho)
      coef = fft_max/ganho_max
      ganho *= coef
      return ganho
#    ############################################

    def dyMod(gamma, om):
      dy = -gamma*om*om/np.cbrt((om**2-om_0**2)**2+(gamma*om)**2)
      return dy
#    ############################################

    def residual(gamma):
        yM = yMod(gamma, 2*np.pi*xdata)
        dif = np.sum((yM-ydata)**2)
        return dif
#    ############################################

    def dres(gamma):
        yM = yMod(gamma, 2*np.pi*xdata)
        dy = dyMod(gamma, 2*np.pi*xdata)
        dif = np.sum((yM-ydata)*dy)
        return dif
#    ############################################

#axins.plot(xdata, yMod(xdata, *gammaFit), 'r-', label = u"Teórico", linewidth = 1)
    axins.plot(freqs[1:N_fft//2], FFT[1:N_fft//2], 'bo', label = "Experimental",\
            linewidth = 1.5, ms = 2.5)
    ga1 = 0.01
    ga2 = 10.0
    f1 = dres(ga1)
    f2 = dres(ga2)
    if f1*f2 < 0:
      rt=scipy.optimize.bisect (dres, ga1, ga2)
      ga_fit = rt
      #print('f_0', f_0, 'ga_fit', ga_fit)
    ganho = yMod(ga_fit, om)
    legenda = 'Teórico $f_0=%.5g$Hz\n $\gamma=%.4g$Hz' % (f_0, ga_fit)
    axins.plot(freqInt, ganho, 'r-', label = legenda, linewidth = 1)
    axins.legend(fontsize=8, loc = "upper right")
    X = time
    dt= time[1]-time[0]
    T = 1.0/f_0
    nPer = int(T/dt)
    #print('nPer', nPer)
    C = 0.3*np.amax(signal[-nPer:])
#ax1.axvline(time[-nPer])
#C = np.amax(signal[15*int(len(signal)/16):])
    x = np.linspace(0,12,306720)
    y = (0.0216938+0.488275)*np.exp(-0.223449*x)*np.cos(2*np.pi*26.472*x+1.84)#Função da Posição pelo Tempo para o Ajuste
    ax1.plot(X, y, c='green')
    envelope = C+(y_max-C)*np.exp(-ga_fit*(X-X[0])/2)
    legenda = u"$f(t)=$ $%g+%g e^{-%g t}$" % (C, y_max-C, ga_fit/2)
    ax1.plot(X, envelope, 'r-', label = legenda)
    ax1.plot(X, -envelope,'r-')
    ax1.legend(loc="upper right")
    Q = 2*np.pi*f_0/ga_fit
    #print('Q', Q)
    print(ga_fit)
    print(s)
    ax1.set_title(u'Série temporal, $Q=%.4g$'%(Q),fontsize=14)
    figura = sys.argv[1].replace('.csv', '')
    figura = figura+'f_0%.5gga%.4g.pdf' %(f_0, ga_fit)
    #print(figura)
    plt.savefig(figura)
    return ga_fit
freqop(414)
##Achando o Melhor Valor da Janela do Filtro de Média Móvel
#m=512
#while (freqop(m)-freqop(m-1))>=0:
#       m=m-1
#print(freqop(m))
#a = np.array([999])
#for s in range(300,400):
#    a = np.append(a, freqop(s))
#print(np.min(a))
#print(np.where(a == np.min(a)))
plt.show()
```

```python
###CÓDIGO PARA O MELHOR VALOR DE TRUNCAMENTO PARA A MAIOR FREQUÊNCIA NATURAL DE OSCILAÇÃO###
import sys
import matplotlib.pyplot as plt
import numpy as np
```

```python
import pandas as pd
import scipy.fftpack
plt.rcParams['axes.formatter.use_locale'] = True

##Função para o Algoritmo do Melhor Valor bf de Truncamento
def freqop(s, bf):
    df = pd.read_csv(sys.argv[1], sep =',', names=['t', 'x'], header=2)#Leitura dos Dados para DataFrame
    df = df.truncate(before=bf, after=500000)#Truncamento
    df['Média móvel'] = df['x'].rolling(s).mean()#Filtro Móvel
    dft = df['t']#DataFrame dos Valores de Tempos
    dfm = pd.merge(dft, df['Média móvel'].dropna(), right_index=True, left_index=True)#DataFrame da Média Móvel com o Tempo
    #arq = sys.argv[1].replace('.csv','') + 'Filtrado.csv'#Definindo Nome do Arquivo
    #dfm.to_csv(arq, index=False)#Salvando Arquivo
   # print(dfm)
#pd.display(df)

#print(df)
#figura = sys.argv[1].replace('.csv', '.pdf')
#plt.savefig(figura)
#print(figura)
# Transformada de Fourier
    def FFT(signal, time):
        N_fft = len(signal)
        fft = 2*np.abs(scipy.fftpack.fft(signal))/N_fft
        dt = time[1]-time[0]
        freqs = scipy.fftpack.fftfreq(signal.size, dt)
        return [freqs, fft]
# #####################################################################

# taxa de amostragem para fft
    sampRate = 1
    time = np.array(df['t'])
    signal = np.array(df['x'])
    time = time[::sampRate]
    signal = signal[::sampRate]
# plotar serie temporal
    fig, ax = plt.subplots(1, figsize=(8, 8))
# Transformada de Fourier
    freqs, FFT = FFT(signal, time)
    N_fft = len(freqs)
    fft_max = np.amax(FFT[:N_fft//2])
    n_max = np.argmax(FFT[:N_fft//2])
    print(FFT[:N_fft//2][18])
    f_max = n_max*(freqs[1]-freqs[0])
    print ("f_max = ", f_max, freqs[1]-freqs[0])
    ax.set_yscale('log')
    ax.set_xticks(np.arange(0., 10., 1.))
#ax.set_title('Transformada de Fourier', fontsize=14)
    ax.set_xlim(0., 30.)
    ax.set_ylim([10**-6, 10**1])
    ax.plot(freqs[:N_fft//2], FFT[:N_fft//2], 'b-', linewidth = 0.5)
    ax.grid()
    x = np.linspace(-5.15,-4.7,1000)
    y = (0.0216938+0.488275)*np.exp(-0.223449*(x))*np.cos(2*np.pi*26*(x)+0.4)
    plt.figure()
    plt.plot(dfm['t'], dfm['Média móvel'], mec='black', ms=1, label = 'Dados não filtrados')
    plt.plot(x,y)
    plt.xlabel('$t (s)$')
    plt.ylabel(r'$V(t)$')
    plt.legend()
    plt.grid()
    return f_max
    print(bf)
##Achando o Melhor Valor de bf para o Truncamento e Maior Frequência Natural de Oscilação
freqop(414, 483000)
#a = np.array([0])
#for bf in range(495000,480000,-1000):
#     a = np.append(a, freqop(1,bf))
#print(np.max(a))
#print(np.where(a == np.max(a)))
plt.show()
```