MRR_viga

August 1, 2021

[1]:
```
import numpy as np
from scipy.linalg import eig
import matplotlib.pyplot as plt
```

1 Propriedades da viga

[2]:
```
L=0.5
b=0.02
h=0.005
Ar=b*h
I=b*h**3/12
pho=2700
E=7.1e10
```

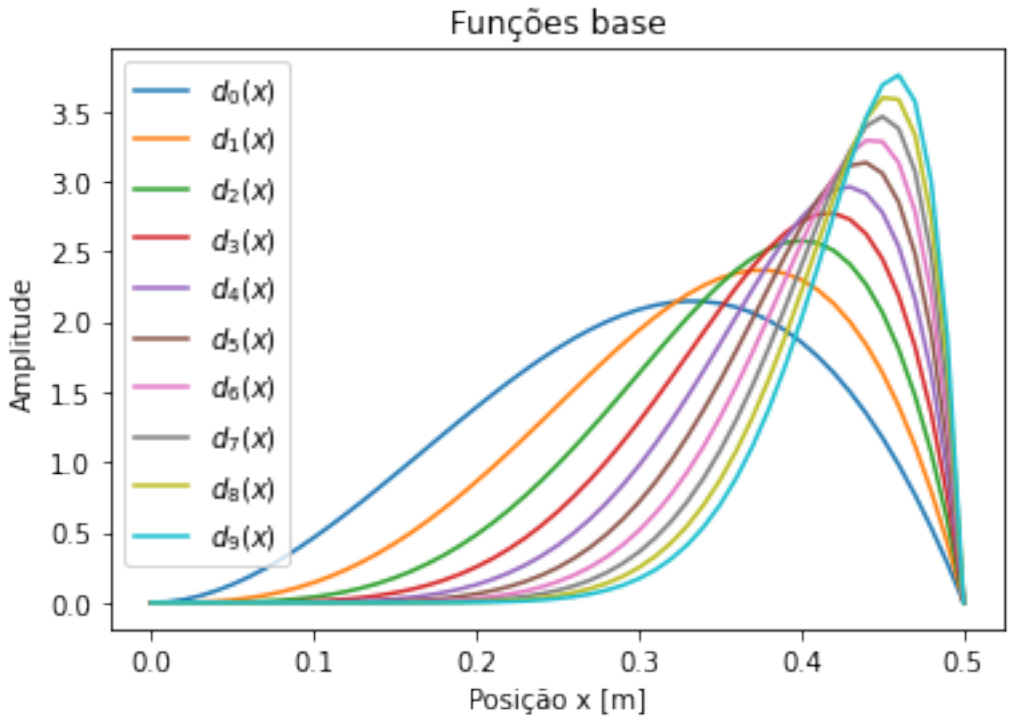
2 Análise

[3]:
```
n=[4, 6, 10] # Numero de funcoes base em cada solucao
# Obs: Um numero de funcoes base maior que 10 apresenta problemas, o que
# pode ser devido a ma qualidade das funcoes base (elas passam a ter pouca
# diferenca entre elas para valores de i grandes).

resultados = dict()
x = np.linspace(0, L, int(L/0.01 + 1)) # Pontos onde a resposta sera calculada
for N in n:
    M = np.zeros((N,N))
    K = np.zeros((N,N))

    for j in range(1, N+1):
        for k in range(1, N+1):
            # As constantes abaixo sao a norma das funcoes e sao usadas
            # para normatizar as funcoes (o que nem sempre eh necessario).
            C1 = L**(j+5/2)*(1/(2*j+3)-2/(2*j+4)+1/(2*j+5))**0.5
            C2 = L**(k+5/2)*(1/(2*k+3)-2/(2*k+4)+1/(2*k+5))**0.5
            # C1 = 1
            # C2 = 1
            # Matriz de massa e rigidez
```

```
            M[j-1,k-1] = pho*Ar*(L**(j+k+5)/(j+k+3)-2*L**(j+k+5)/
↪(j+k+4)+L**(j+k+5)/(j+k+5))/(C1*C2)

            K[j-1,k-1] = E*I*(j*k*(j+1)*(k+1)*L**(j+k+1)/
↪(j+k-1)-k*(j+2)*(j+1)*(k+1)*L**(j+k+1)/(j+k)-j*(j+1)*(k+2)*(k+1)*L**(j+k+1)/
↪(j+k)+(j+2)*(j+1)*(k+2)*(k+1)*L**(j+k+1)/(j+k+1))/(C1*C2)

    # Problema de autovalor generalizado. W é um vetor e A uma matrix com as␣
↪colunas normalizadas
    W, A = eig(K, M)

    # Ordenando os autovalores e a matriz de autovetores
    idx = W.argsort()
    W = W[idx]
    A = A[:,idx]

    # Normalizando os autovetores pela matriz de massa, de forma que A'@M@A = I␣
↪(@ = matrix multiplication)
    m_r = np.diagonal(A.T @ M @ A)
    m_r = np.reciprocal(np.sqrt(m_r))
    for a in range(A.shape[1]):
        A[:,a] *= m_r[a] # multiplica cada coluna pelo fator de escala

    # Calculando as funções base nos pontos x
    d = np.zeros((N, x.size))
    for j in range(1, N+1):
        C1 = L**(j+5/2)*(1/(2*j+3)-2/(2*j+4)+1/(2*j+5))**0.5
        d[j-1,:] = x**(j+1)*(L-x)/C1

    # Faz as formas modais terem a mesma orientação, analisando o sinal da␣
↪covariância
    phi = d[:N, :].T @ A
    try:
        for k in range(N): # k-ésimo modo
            cov = np.cov(resultados[n[0]]['V'][:,k], phi[:,k])[0][1]
            cov = cov/np.abs(cov) # -1 ou 1
            phi[:,k] *= cov
    except:
        pass

    # armazenando os resultados em um dicionário
    resultados[N] = dict()
    resultados[N]['V'] = phi
    resultados[N]['fn'] = np.real(W**0.5/(2*np.pi))
    resultados[N]['d'] = d
```

2.1 Funcoes base utilizadas na análise

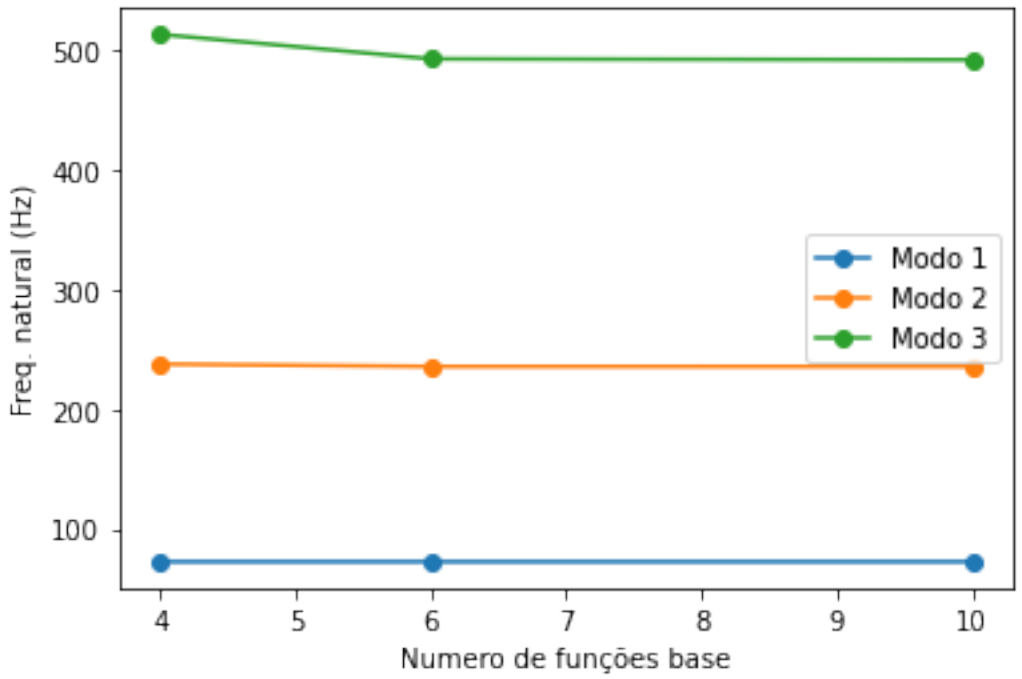
[4]:
```
n_b = n[2]
for i in range(n_b):
    plt.plot(x, resultados[n_b]['d'][i], label=f'$d_{i}(x)$')

plt.title('Funções base')
plt.xlabel('Posição x [m]')
plt.ylabel('Amplitude')
plt.legend()
plt.show()
```



2.2 Comparação das frequências naturais

[5]:
```
fn_N=np.zeros((4, len(n)))
for j in range(0,3):
    for k in range(len(n)):
        fn_N[j, k] = resultados[n[k]]['fn'][j]

for k in range(fn_N[1,:].size):
    plt.plot(n, fn_N[k,:], marker='o')
plt.xlabel('Numero de funções base')
plt.ylabel('Freq. natural (Hz)')
plt.legend(['Modo 1', 'Modo 2','Modo 3'])
```

```
plt.show()
```



2.3 Comparação das formas modais

[6]:
```
fig, axs = plt.subplots(2,2)

for n_b in n:
    axs[0,0].set_title('Primeiro modo')
    axs[0,0].plot(x, resultados[n_b]['V'][:,0], label=f"{n_b} funções")
    axs[0,0].set_xlabel('x [m]')
    axs[0,0].set_ylabel('Forma modal')

    axs[0,1].set_title('Segundo modo')
    axs[0,1].plot(x, resultados[n_b]['V'][:,1])
    axs[0,1].set_xlabel('x [m]')
    axs[0,1].set_ylabel('Forma modal')

    axs[1,0].set_title('Terceiro modo')
    axs[1,0].plot(x, resultados[n_b]['V'][:,2])
    axs[1,0].set_xlabel('x [m]')
    axs[1,0].set_ylabel('Forma modal')

    axs[1,1].set_title('Quarto modo modo')
    axs[1,1].plot(x, resultados[n_b]['V'][:,3])
```

```
    axs[1,1].set_xlabel('x [m]')
    axs[1,1].set_ylabel('Forma modal')


fig.legend(loc='upper left', bbox_to_anchor=(1, 0.9))
fig.tight_layout()
plt.show()
```