Universidade Federal de Alagoas - UFAL
Centro de Tecnologia - CTEC
Programa de Pós-Graduação em Engenharia Civil - PPGEC

Dinâmica de estruturas

Lista de exercícios 3

Weverton Marques da Silva
28 de janeiro de 2019

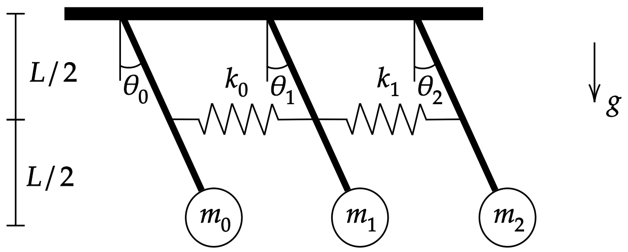
Questão 1: Equação de movimento

Definir um exemplo com mais de dois graus de liberdade e obter a equação de movimento utilizando os dois métodos abaixo

- Método direto
- Princípio de Hamilton

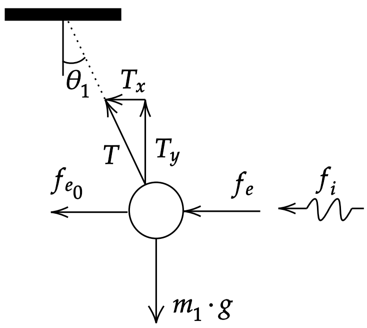
Solução

O exemplo escolhido é baseado no exercício 8.9 livro do Craig [1] (https://www.amazon.com/Fundamentals-Structural-Dynamics-Roy-Craig/dp/0471430447):



Método direto

Vamos montrar a equação de equilíbrio para o grau de θ_1 já que os outros podem ser considerados casos particulares dele. Para esta massa temos o seguinte diagrama de corpo livre:



As manipulações algébricas a seguir serão realizadas usando a biblioteca de Matemática simbólica SymPy (https://www.sympy.org/en/index.html).

In [1]:
```
from sympy import *
interactive.printing.init_printing(use_unicode=True)
m = symbols("m0, m1, m2") # Massas
k = symbols("k0, k1") # Constantes elásticas
L, g = symbols("L, g", positive=True) # Constantes do problema
fe0, fe1, Tx, fi, t = symbols("fe_0, fe_1, T_x, f_i, t") # Símbolos auxiliares
theta = symbols("theta0, theta1, theta2", cls=Function) # Função dos d.o.f. no tempo
eq = {}
```

Considerando o sentido positivo dos θ's como anti-horário, vamos definir valor das forças:

In [2]:
```
i = 1
fe0 = k[0] * (L/2) * sin(theta[i](t))
fe1 = k[1] * (L/2) * sin(theta[i](t))
Tx = m[i] * g * sin(theta[i](t))
fi = m[i] * (theta[i](t) * L).diff(t, t)
```

Assim, temos como equação de equilíbrio na direção horizontal:

In [3]:
```
eq[i] = Eq(fi + Tx + fe0 + fe1, 0); display(eq[i])
```

$$\frac{Lk_0 \sin\left(\theta_1(t)\right)}{2} + \frac{Lk_1 \sin\left(\theta_1(t)\right)}{2} + Lm_1 \frac{d^2}{dt^2}\theta_1(t) + gm_1 \sin\left(\theta_1(t)\right) = 0$$

Colocando todos os membros do mesmo lado da equação e fazendo a aproximação $\sin(\theta) \approx \theta$, então, podemos simplificar a equação:

In [4]:
```
eq[i] = eq[i].subs(sin(theta[i](t)), theta[i](t)); display(eq[i])
```

$$\frac{Lk_0\theta_1(t)}{2} + \frac{Lk_1\theta_1(t)}{2} + Lm_1 \frac{d^2}{dt^2}\theta_1(t) + gm_1\theta_1(t) = 0$$

De forma análoga para as masssas m_0 e m_2:

In [5]:
```
i = 0
fe0 = k[0] * L/2 * theta[i](t)
fe1 = 0
Tx = m[i] * g * theta[i](t)
fi = m[i] * (theta[i](t) * L).diff(t, t)
eq[i] = Eq(fi + Tx + fe0 + fe1, 0)
```

In [6]:
```
display(eq[i])
```

$$\frac{Lk_0\theta_0(t)}{2} + Lm_0 \frac{d^2}{dt^2}\theta_0(t) + gm_0\theta_0(t) = 0$$

In [7]:
```
i = 2
fe0 = 0
fe1 = k[1] * L/2 * theta[i](t)
Tx = m[i] * g * theta[i](t)
fi = m[i] * (theta[i](t) * L).diff(t, t)
eq[i] = Eq(fi + Tx+ fe0 + fe1, 0)
```

In [8]:
```
display(eq[2])
```

$$\frac{Lk_1\theta_2(t)}{2} + Lm_2\frac{d^2}{dt^2}\theta_2(t) + gm_2\theta_2(t) = 0$$

Princípio de Hamilton

$$\frac{d}{dt}\frac{\partial T}{\partial \dot{q}} - \frac{\partial T}{\partial q} + \frac{\partial U}{\partial \dot{q}} - Q = 0$$

- Parcela $\frac{d}{dt}\frac{\partial T_i}{\partial v_i}$:

Sendo $T_i = \frac{m_i v_i^2}{2} \Rightarrow \frac{\partial T_i}{\partial v_i} = m_i v_i$. Assumindo θ_i pequeno, $v_i = L\frac{d}{dt}\theta_i(t)$, assim temos $\frac{\partial T_i}{\partial v_i} = m_i L\frac{d}{dt}\theta_i(t)$:

In [9]:
```
dT_dv = [(m[i] * L * (theta[i](t).diff(t))) for i in range(len(theta))]; dT_dv
```

Out[9]:

$$\left[Lm_0\frac{d}{dt}\theta_0(t), \quad Lm_1\frac{d}{dt}\theta_1(t), \quad Lm_2\frac{d}{dt}\theta_2(t)\right]$$

E consequentemente $\frac{d}{dt}\frac{\partial T_i}{\partial v_i}$:

In [10]:
```
ddt_dT_dv = [dT_dv[i].diff(t) for i in range(len(theta))]; ddt_dT_dv
```

Out[10]:

$$\left[Lm_0\frac{d^2}{dt^2}\theta_0(t), \quad Lm_1\frac{d^2}{dt^2}\theta_1(t), \quad Lm_2\frac{d^2}{dt^2}\theta_2(t)\right]$$

- Parcela $\frac{\partial T_i}{\partial \theta_i} = 0$

In [11]:
```
dT_dq = [0] * len(theta)
dT_dq
```

Out[11]: $[0, \quad 0, \quad 0]$

- Energia potencial (elásica): $U_i = \dfrac{K_i \cdot x_i^2}{2} \Rightarrow \dfrac{\partial U_i}{\partial x_i} = K_i \cdot x_i$, e assumindo $x_i = (L/2) \cdot \theta_i(t)$, e K definido como:

In [12]:
```
K = [k[0], k[0]+k[1], k[1]]
```

In [13]:
```
dU_dq = []
for i in range(len(theta)):
    q = theta[i](t)
    dU_dq.append((K[i] * L/2 * q))
dU_dq
```

Out[13]: $$\left[\frac{Lk_0\theta_0(t)}{2}, \quad \frac{L\left(k_0 + k_1\right)\theta_1(t)}{2}, \quad \frac{Lk_1\theta_2(t)}{2} \right]$$

Trabalho das formas não conservativas: $\delta Wc_i = (-m_i \cdot g \cdot \theta_i)\delta u \Rightarrow Q_i = -m_i \cdot g \cdot \theta_i$:

In [14]:
```
Q = [-(m[i] * g * theta[i](t)) for i in range(len(theta))]
Q
```

Out[14]: $$\left[-gm_0\theta_0(t), \quad -gm_1\theta_1(t), \quad -gm_2\theta_2(t) \right]$$

Armando as equações para cada grau de liberdade, temos:

In [15]:
```
for i in range(len(theta)):
    eq[i] = Eq(dT_dv[i].diff(t) - dT_dq[i] + dU_dq[i] - Q[i], 0)
    display(eq[i])
```

$$\frac{Lk_0\theta_0(t)}{2} + Lm_0\frac{d^2}{dt^2}\theta_0(t) + gm_0\theta_0(t) = 0$$

$$Lm_1\frac{d^2}{dt^2}\theta_1(t) + \frac{L\left(k_0 + k_1\right)\theta_1(t)}{2} + gm_1\theta_1(t) = 0$$

$$\frac{Lk_1\theta_2(t)}{2} + Lm_2\frac{d^2}{dt^2}\theta_2(t) + gm_2\theta_2(t) = 0$$

Questão 2: Resposta no tempo

Para as equações de movimento obtidas no exemplo anterior obter a solução no tempo utilizando os dois métodos abaixo. Neste exemplo devem ser definidos três tipos de excitação: *impulsivas, quase estáticas e harmônicas*. Fazer uma análise comparativa de desempenho dos algoritmos para cada tipo de excitação.

- Superposição modal
- Integração temporal: Algoritmo da diferença central e de Newmark

Solução

Com base nas equações encontrada na questão 1, podemos definir as montar as matrizes de massa, rigidez, e amortecimento do problema:

In [16]:
```
M_sym = zeros(3,3)
for i in range(3):
    for j in range(3):
        M_sym[i, j] = eq[i].lhs.coeff((theta[j](t)).diff(t,t))

K_sym = Matrix([[0]*3]*3)
for i in range(3):
    for j in range(3):
        K_sym[i, j] = eq[i].lhs.coeff(theta[j](t))

C_sym = Matrix([[0]*3]*3)
for i in range(3):
    for j in range(3):
        C_sym[i, j] = eq[i].lhs.coeff(theta[j](t).diff(t))
```

In [17]:
```
display(M_sym)
display(K_sym)
```

$$\left[\begin{matrix}Lm_0 & 0 & 0\\0 & Lm_1 & 0\\0 & 0 & Lm_2\end{matrix}\right]$$

$$\left[\begin{matrix}\frac{Lk_0}{2} + gm_0 & 0 & 0\\0 & \frac{L(k_0+k_1)}{2} + gm_1 & 0\\0 & 0 & \frac{Lk_1}{2} + gm_2\end{matrix}\right]$$

Atribuindo valores, temos:

In [18]:
```
substituitions = [(m[0], 1), (m[1], 1), (m[2], 1), (k[0], 10), (k[1], 10), (L, 1),(g, 9.816)]
```

In [19]:
```
M_sym = M_sym.subs(substituitions)
K_sym = K_sym.subs(substituitions)
C_sym = C_sym.subs(substituitions)
```

In [20]:
```
display(M_sym)
display(K_sym)
```

$$\left[\begin{matrix}1 & 0 & 0\\0 & 1 & 0\\0 & 0 & 1\end{matrix}\right]$$

$$\left[\begin{matrix}14.816 & 0 & 0\\0 & 19.816 & 0\\0 & 0 & 14.816\end{matrix}\right]$$

```
In [21]: import numpy as np
         import scipy.linalg as la
         # Convertendo as matrizes simbólicas do sympy em numpy.array
         M = np.array(M_sym).astype(np.float64)
         K = np.array(K_sym).astype(np.float64)
         C = np.array(C_sym).astype(np.float64)
         # Frequência máxima e dt crítico
         freq = np.sqrt(la.eigh(K, M)[0])
         fmax = max(freq)
         fmin = min(freq)
         def summary():
             print(f"Frequência natural máxima: {fmax:.4g} Hz")
             print(f"Frequência natural mínima: {fmin:.4g} Hz")
             print(f"Maior Período: {2*np.pi / fmin:.4g} s")
             print(f"Menor Período: {2*np.pi / fmax:.4g} s")
```

```
In [22]: summary()
```

```
Frequência natural máxima: 4.452 Hz
Frequência natural mínima: 3.849 Hz
Maior Período: 1.632 s
Menor Período: 1.411 s
```

Carga harmônica

Uma vez que a frequência natural da estrutura é 4.452 Hz, então vamos adotar uma carga com frequência 5 hz, isto é, `omega = 5` . Vamos também definir as condições iniciais do problema:

```
In [23]: from python.load_cases import *
         from python.numeric_solutions import diferencacentral, newmark_linear
         # Condições iniciais do problema
         u0 = np.array([0, 0, 0])
         v0 = np.array([0, 0, 0])
         fo = np.array([10, 15, 20])
         # Tempo de simulacao-(s)
         sim_time = 20
         # Número de incrementos de tempo
         num_steps = 500
         # Incremento de tempo crítico
         dtcr = 2 * np.pi / fmax
         dt = (sim_time * dtcr / num_steps) / 2 # dtcr / 10
         omega = 5  # Frquência da carga
         carga_harmonica = Harmonic(fo, omega)
         resposta = carga_harmonica.response
         load = carga_harmonica.load
         print(f"Relação dt/dt crítico: {dt/dtcr:.1%}")
```

```
Relação dt/dt crítico: 2.0%
```

Vamos agora, medir o tempo gasto por cada algorítmo:

- superposição modal:

```
In [24]: %timeit carga_harmonica.modal_superposition(M, C, K, u0, v0, fo, sim_time, num_
         steps)
```

```
588 µs ± 35.1 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
```

- Diferença central:

In [25]:
```
%timeit diferencacentral(M, C, K, load, u0, v0, sim_time, dt)
```

```
59.6 ms ± 983 µs per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

- Newmar (linear):

In [26]:
```
%timeit newmark_linear(M, C, K, load, u0, v0, sim_time, dt, 1/2, 1/4)
```

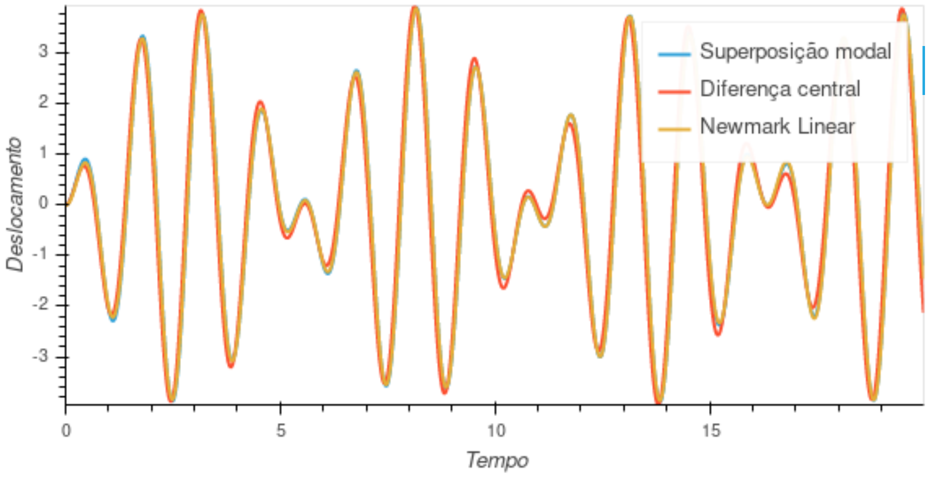
```
75.7 ms ± 1.79 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

Agora, vamos visualizar as resposta de cada algorítmo para o grau de liberdade 1, por exemplo:

In [27]:
```
import holoviews as hv
from IPython.display import HTML, display
hv.extension("bokeh")

def grafico(resposta_modal, resposta_df, resposta_nl, dof):
    import holoviews as hv
    hv.extension("bokeh")
    dof = 2
    sm_plot = hv.Curve((resposta_modal[0], resposta_modal[1][dof]), "Tempo", "D
eslocamento", label=f"Superposição modal")
    dc_plot = hv.Curve((resposta_df[0], resposta_df[1][dof]), "Tempo", "Desloca
mento", label=f"Diferença central")
    nm_plot = hv.Curve((resposta_nl[0], resposta_nl[1][dof]), "Tempo", "Desloca
mento", label=f"Newmark Linear")
    layout = (sm_plot * dc_plot * nm_plot).opts(height=300, width=600)
    hv.save(layout, filename="carga_rampa.html")
    return HTML(open("carga_rampa.html", "r").read())
```

In [28]:
```
resposta_modal = carga_harmonica.modal_superposition(M, C, K, u0, v0, fo, sim_t
ime, num_steps)
resposta_df = diferencacentral(M, C, K, load, u0, v0, sim_time, dt)
resposta_nl = newmark_linear(M, C, K, load, u0, v0, sim_time, dt, 1/2, 1/4)
display(grafico(resposta_modal, resposta_df, resposta_nl, dof=1))
```



(htt

Carga impulsiva (pulso retangular não amortecido)

Uma vez que maior período da estrutura está entorno 1,6 s, então vamos adotar uma carga com tempo de duração em torno de 1,1 vezes o crítico (`td = 1.1*dtcr`). E vamos também definir as condições iniciais do problema:

In [29]:
```
from python.load_cases import Impulsive
# Condições iniciais do problema
u0 = np.array([0, 0, 0])
v0 = np.array([0, 0, 0])
fo = np.array([10, 15, 20])
# Tempo de simulacao (s)
sim_time = 10
# Número de incrementos de tempo
num_steps = 500
# Incremento de tempo crítico
dtcr = 2 * np.pi / fmax
# dt = (sim_time * dtcr / num_steps) / 3
dt = dtcr / 100
td = dtcr * 1.1  # Tempo de duração do pulso.
carga_impulsiva = Impulsive(fo, td)
load = carga_impulsiva.load
print(f"Relação dt/dt crítico: {dt/dtcr*100:.5g}%")
```

```
Relação dt/dt crítico: 1%
```

Vamos agora, medir o tempo gasto por cada algorítmo:

- superposição modal:

In [30]:
```
%timeit carga_impulsiva.modal_superposition(M, K, u0, fo, sim_time, num_steps)
```

```
762 µs ± 18.6 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
```

- Diferença central:

In [31]:
```
%timeit diferencacentral(M, C, K, load, u0, v0, sim_time, dt)
```

```
58.6 ms ± 483 µs per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

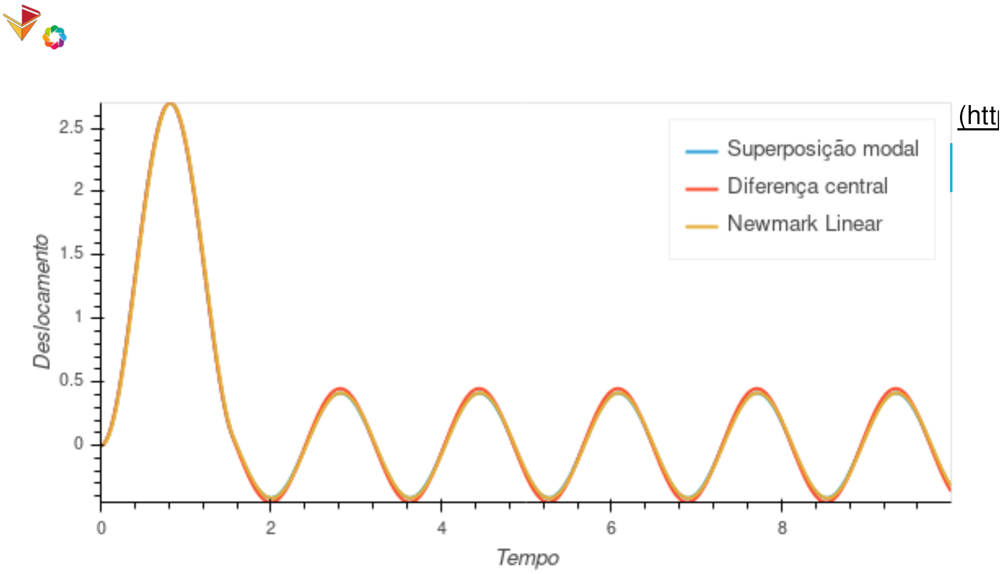
- Newmar (linear):

In [32]:
```
%timeit newmark_linear(M, C, K, load, u0, v0, sim_time, dt, 1/2, 1/4)
```

```
75.3 ms ± 475 µs per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

Agora, vamos visualizar as resposta de cada algorítmo para o grau de liberdade 1, por exemplo:

In [33]:
```
resposta_modal = carga_impulsiva.modal_superposition(M, K, u0, fo, sim_time, nu
m_steps)
resposta_df = diferencacentral(M, C, K, load, u0, v0, sim_time, dt)
resposta_nl = newmark_linear(M, C, K, load, u0, v0, sim_time, dt, 1/2, 1/4)
display(grafico(resposta_modal, resposta_df, resposta_nl, dof=1))
```



Carga rampa (quase-estática)

Uma vez que maior período da estrutura está entorno 1,6 s, então vamos adotar uma carga com tempo de rampa de 10 vezes o maior período (`tr = dtcr * 10`). Vamos também definir as condições iniciais do problema:

In [34]:
```
from python.load_cases import Ramp
# Condições iniciais do problema
u0 = np.array([0, 0, 0])
v0 = np.array([0, 0, 0])
fo = np.array([10, 15, 20])
# Tempo de simulacao-(s)
sim_time = 30
# Número de incrementos de tempo
num_steps = 5000
# Incremento de tempo crítico
dtcr = 2 * np.pi / fmax
dt = sim_time * dtcr / num_steps # dtcr / 10
tr = dtcr * 10  # Tempo de duração do pulso.
carga_rampa = Ramp(fo, tr)
load = carga_rampa.load
print(f"Relação dt/dt crítico: {dt/dtcr*100:.5g}%")
```

```
Relação dt/dt crítico: 0.6%
```

Vamos agora, medir o tempo gasto por cada algorítmo:

- superposição modal:

In [35]:
```
%timeit carga_rampa.modal_superposition(M, K, fo, sim_time, num_steps)
```

```
4.36 ms ± 141 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

- Diferença central:

In [36]:
```
%timeit diferencacentral(M, C, K, load, u0, v0, sim_time, dt)
```

```
315 ms ± 20.1 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

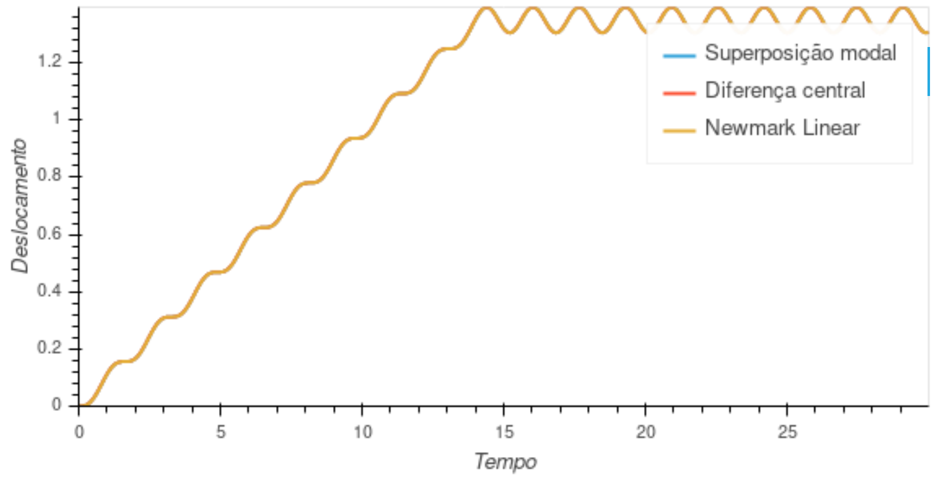
- Newmar (linear):

In [37]:
```
%timeit newmark_linear(M, C, K, load, u0, v0, sim_time, dt, 1/2, 1/4)
```

```
390 ms ± 7.91 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

Agora, vamos visualizar as resposta de cada algorítmo para o grau de liberdade 2, por exemplo:

In [38]:
```
resposta_modal = carga_rampa.modal_superposition(M, K, fo, sim_time, num_steps)
resposta_df = diferencacentral(M, C, K, load, u0, v0, sim_time, dt)
resposta_nl = newmark_linear(M, C, K, load, u0, v0, sim_time, dt, 1/2, 1/4)
display(grafico(resposta_modal, resposta_df, resposta_nl, dof=1))
```



(htt

Conclusão

- Equações de movimento:
  - Método direto é mais intuitiva, porém menos generalizável que o Princípio de Hamilton
- Comparação dos algorítmos:
  - Desde que sejam usando parâmetros adequados, todos os métodos conseguem fornecer respostas precisas;
  - Em todos os casos, em ordem crescente, o algorítmo mais performático foi:
    1. Superposição modal
    2. Diferença central
    3. Newmark linear