Latihan Osilasi Harmonik
TF2103

Kerjakan secara individu atau berkelompok (dua orang).

Anda dapat menggunakan program komputer seperti Excel, MATLAB, atau Python untuk membuat plot atau visualisasi.

Untuk penurunan rumus, Anda dapat menggunakan tulisan tangan yang di-scan atau diketik atau menggunakan SymPy / MATLAB.

Laporan dikumpulkan dalam bentuk ipynb (jika menggunakan Jupyter Lab) dan/atau PDF berisi penjelasan dan penurunan rumus.

Soal 1. Energi potensial dari dua atom pada suat molekul dapat diaproksimasi menggunakan fungsi Morse:

$$U(r) = A\left[\left(e^{(R-r)/S} - 1\right)^2 - 1\right] \tag{1}$$

dengan r adalah jarak antara dua atom dan A, R dan S adalah konstanta positif dan $S \ll R$. Tentukan jarak r_0 yang menyebabkan $U(r)$ bernilai minimum. Tulis $r = r_0 + x$ di mana x adalah pergeseran dari ekuilibrium dan tunjukkan bahwa untuk x kecil, U dapat diaproksimasi menjadi

$$U_{\mathrm{approx}}(r) = \text{konstanta} + \frac{1}{2}kx^2 \tag{2}$$

Berikan ekspresi untuk k. Buat plot untuk $U(r)$ dan $U_{\mathrm{approx}}(r)$ dengan nilai A, R dan S yang Anda pilih.

Soal 2. Tinjau solusi umum dari osilasi harmonik terpaksa dengan gaya pemaksa berbentuk cosinus:

$$x(t) = A\cos(\omega t - \delta) + C_1 e^{r_1 t} + C_2 e^{r_2 t} \tag{3}$$

atau, untuk kasus *critically damped*:

$$x(t) = A\cos(\omega t - \delta) + C_1 e^{r_1 t} + C_2 t e^{r_1 t} \tag{4}$$

Berikan ekspresi untuk C_1 dan C_2 untuk kasus *underdamped*, *overdamped*, dan *critically-damped* untuk kondisi awal $x(t = 0) = x_0$ dan $x'(t = 0) = v_0$.

Soal 3. Tinjau kasus osilasi tanpa gaya paksa (atau $f_0 = 0$). Dengan menggunakan kondisi awal $x(t = 0) = 1.0$ dan $x'(t = 0) = 0$, dan nilai ω_0 yang Anda pilih, buat plot $x(t)$ untuk nilai-nilai β berikut (dalam satu plot) $\omega_0/10, \omega_0/2, \omega_0, 2\omega_0$, dan $10\omega_0$.

Soal 4. Tinjau Contoh 5.3 pada Taylor. Modifikasi parameter-parameter yang diberikan, misalnya *driving frequency* ω dan parameter lain. Buat visualisasi untuk parameter-parameter yang Anda pilih (minimal 3 set parameter) dan berikan komentar atau penjelasan dari hasil yang Anda peroleh.

Anda dapat melengkapi kode berikut jika diperlukan, tidak harus mengikuti program yang seperti ini, Anda dapat menulis program yang lain.

```python
import numpy as np
import matplotlib.pyplot as plt
import cmath

def calc_auxiliary_roots(ω0, β):
    r1 = .... # lengkapi
    r2 = .... # lengkapi
    return r1, r2

def simulate_harmonic_oscillation(params, tfinal=5.0, NptsPlot=1000):
    ω = params["ω"]
    f0 = params["f0"]
    ω0 = params["ω0"]
    β = params["β"]
    x0 = params["x0"]
    v0 = params["v0"]

    r1, r2 = calc_auxiliary_roots(ω0, β)
    # Print out r1 and r2 if needed

    # Handle special case of critically damped
    # we check ω0 and β instead of r1 and r2 as it is easier to do
    SMALL = 1e-10 # a tolerance under which a value is considered to be zero
    CRITICALLY_DAMPED = False
    if np.abs(ω0 - β) <= SMALL:
        print("Oscillation is critically damped")
        CRITICALLY_DAMPED = True

    FORCED_OSCILLATION = False
    if np.abs(f0) > SMALL:
        FORCED_OSCILLATION = True

    if ω0 > β:
        print("Oscillation is underdamped")

    if ω0 < β:
        print("Oscillation is overdamped")

    if β == 0:
        print("Oscillation is not damped")

    # For driven oscillation
    A2 = f0**2/( (ω0**2 - ω**2)**2 + 4*β**2 * ω**2 )
    A = np.sqrt(A2)
    δ = np.arctan2( 2*β*ω, ω0**2 - ω**2 )

    if not FORCED_OSCILLATION:
        cond = (np.abs(x0) < SMALL) and (np.abs(v0) < SMALL)
        if cond:
            print("WARNING: x0 and v0 are both close to zeros")

```

```
    # From Sympy
    if CRITICALLY_DAMPED:
        C1 = .... # lengkapi
        C2 = .... # lengkapi
    else:
        C1 = .... # lengkapi
        C2 = .... # lengkapi

    t = np.linspace(0.0, tfinal, NptsPlot)
    f = f0*np.cos(ω*t)

    if CRITICALLY_DAMPED:
        x = A*np.cos(ω*t - δ) + C1*np.exp(r1*t) + C2*t*np.exp(r1*t)
        v = A*ω*np.sin(δ - ω*t) + C1*r1*np.exp(r1*t) + C2*r1*t*np.exp(r1*t) + C2*np.exp(r1*t)
    else:
        x = A*np.cos(ω*t - δ) + C1*np.exp(r1*t) + C2*np.exp(r2*t)
        v = A*ω*np.sin(δ - ω*t) + C1*np.exp(r1*t)*r1 + C2*np.exp(r2*t)*r2

    # Return various things that might be visualized or processed later
    return t, x, v, f

# Contoh penggunaan
params = {
    "ω" : 2*np.pi,
    "f0" : 1000.0,
    "ω0" : 10*np.pi,
    "β" : 0.5*np.pi,
    "x0" : 0.0,
    "v0" : 0.0
}

plt.clf()
t, x, v, _ = simulate_harmonic_oscillation(params, tfinal=5.0, NptsPlot=1000)
plt.plot(t, np.real(x), label="x(t)")
plt.legend()
plt.grid(True)
```