Homework 6: Damped Harmonic Oscillator via Chebyshev Expansion

A differential equation for the function $f(x)$ can be solved using an expansion of $f(x)$ in the basis of Chebyshev polynomials:

$$f(x) \approx \sum_k c_k T_k(x),$$

where $T_k(x)$ is the Chebyshev polynomial of degree k.

The differential equation for the damped harmonic oscillator is given by

$$\ddot{x}(t) + 2\zeta\omega_0\dot{x}(t) + \omega_0^2 x(t) = 0,$$

where ζ, ω_0, and $x(t)$ are the damping ratio, the natural frequency, and the displacement of the oscillator as a function of time, respectively.

Using a Chebyshev expansion of $x(t)$, solve this equation from $t = 0$ to $t = 20$ with $\omega_0 = 1$ and the following damping ratios: $\zeta = 0.0, 0.2, 1.0,$ and 2.0. The initial conditions are $x(0) = 1$ and $\dot{x}(0) = 0$.

Plot the $x(t)$ vs. t curves for each value of ζ. Ensure that the undamped, underdamped, critically damped, and overdamped cases are represented in blue, green, red, and cyan, respectively. Include a legend with the labels: "undamped", "underdamped", "critically damped", and "overdamped".

You may use the Julia package `ApproxFun.jl` for Chebyshev expansion or equivalent packages suitable for the programming language of your choice.

PHYS416- Computer Applications in Physics

Homework 6 – Damped Harmonic Oscillator via Chebyshev Expansion

Name: Süleyman Ertekin

Student Number: 200218018019

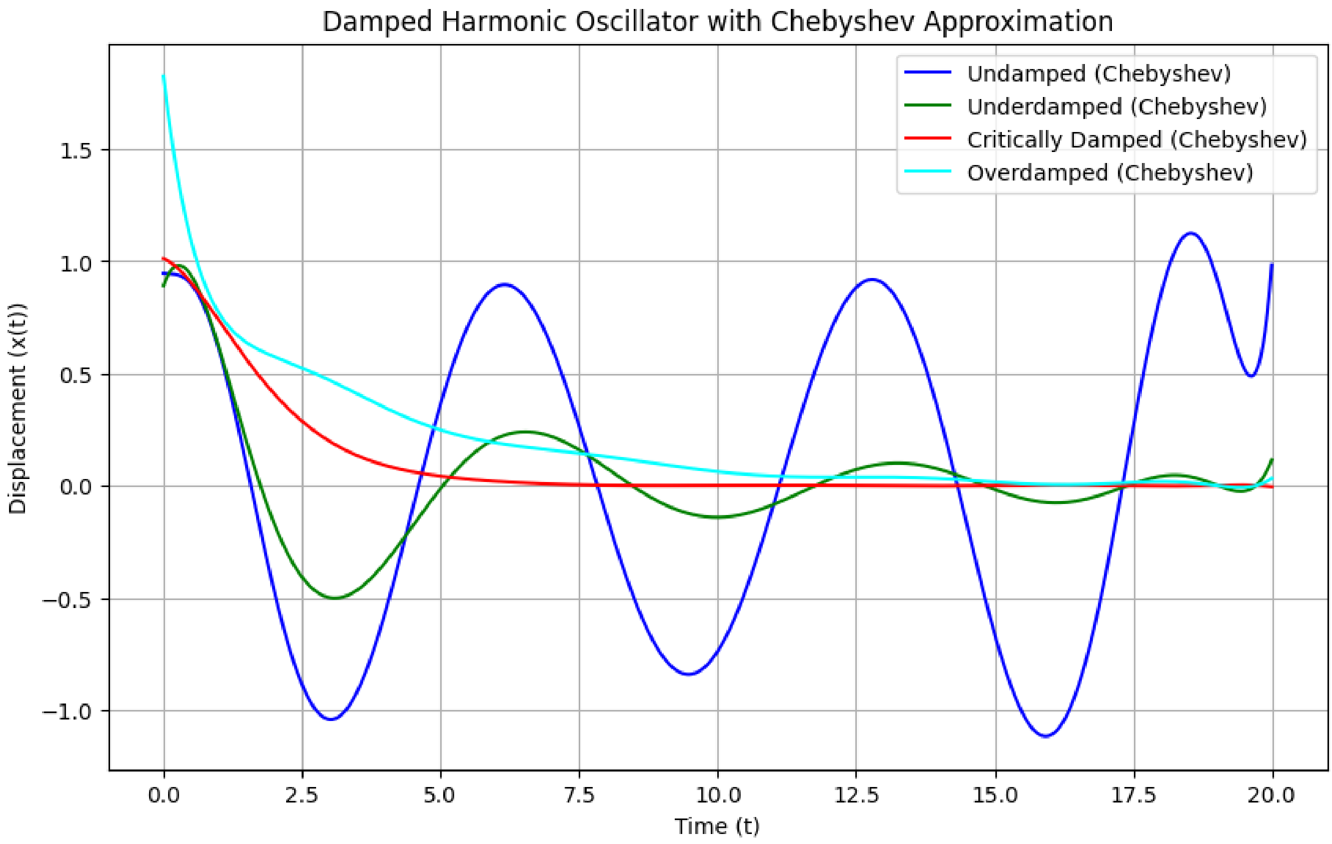
Date of Submission: 8.12.2024

I pledge that I worked entirely alone on this homework and will not share information about any aspect of this homework with any other persons.

Signature: ______________________________

```
import numpy as np
import matplotlib.pyplot as plt
from numpy.polynomial.chebyshev import chebfit, chebval

# Differential equation solution
def damped_oscillator_solution(t, zeta, omega_0):
    # Analytical solution to damped harmonic oscillator
    if zeta < 1:  # Underdamped
        omega_d = omega_0 * np.sqrt(1 - zeta**2)
        return np.exp(-zeta * omega_0 * t) * (np.cos(omega_d * t) + zeta / np.sqrt(1 - zeta**2) * np.sin(omega_d * t))
    elif zeta == 1:  # Critically damped
        return np.exp(-omega_0 * t) * (1 + omega_0 * t)
    else:  # Overdamped
        r1 = -omega_0 * (zeta - np.sqrt(zeta**2 - 1))
        r2 = -omega_0 * (zeta + np.sqrt(zeta**2 - 1))
        return np.exp(r1 * t) + np.exp(r2 * t)

# Parameters
omega_0 = 1
zeta_values = [0.0, 0.2, 1.0, 2.0]
colors = ['blue', 'green', 'red', 'cyan']
labels = ['Undamped', 'Underdamped', 'Critically Damped', 'Overdamped']

# Time points
t = np.linspace(0, 20, 500)
degree = 10  # Degree of Chebyshev polynomial
```

```python

# Plot Chebyshev approximation
plt.figure(figsize=(10, 6))
for zeta, color, label in zip(zeta_values, colors, labels):
    y = damped_oscillator_solution(t, zeta, omega_0)
    # Fit Chebyshev polynomials to the solution
    cheb_coeffs = chebfit(t, y, degree)
    y_approx = chebval(t, cheb_coeffs)

    plt.plot(t, y_approx, color=color, label=f'{label} (Chebyshev)', linestyle='solid')

# Add labels and legend
plt.title('Damped Harmonic Oscillator with Chebyshev Approximation')
plt.xlabel('Time (t)')
plt.ylabel('Displacement (x(t))')
plt.legend()
plt.grid()
plt.show()
```


Damped Harmonic Oscillator with Chebyshev Approximation
Undamped (Chebyshev)
Underdamped (Chebyshev)
Critically Damped (Chebyshev)
Overdamped (Chebyshev)
Displacement (x(t))
Time (t)
1.5
1.0
0.5
0.0
−0.5
−1.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0