Harmonic Oscillator with Damping

Introduction

This activity project solves a damped harmonic oscillator system using an adaptive Runge-Kutta 4th/5th order method (RK45). The system consists of two differential equations for displacement ***x*** and velocity ***v***, with the frequency ω and damping factor β. RK45 computes two approximations at each step and adjusts the step size based on the error estimate to maintain accuracy and efficiency. The solution is computed over a time span, and displacement and velocity are plotted using **matplotlib**, with the adaptive step size optimizing both accuracy and computation time.

Model

The **Harmonic Oscillator with Damping** is a mathematical model that describes the motion of an oscillating system experiencing a restoring force and a damping force. It applies to physical systems such as springs, pendulums in a viscous medium, or RLC electrical circuits. The interplay between these forces governs the system's behavior over time.

$$\frac{dv}{dt} = -\omega^2 x - \beta v \qquad \frac{dx}{dt} = v$$

Variables and Parameters:

1. **v(t) -** Velocity of the oscillator at time t.
2. **x(t) -** Displacement of the oscillator at time t.
3. **ω -** Angular frequency of the undamped system ($\omega = \sqrt{k/m}$).
4. **β -** Damping coefficient (proportional to friction or resistance).

Numerical Solution

Adaptive step size control is a key feature in numerical methods, such as the Runge-Kutta 4th/5th (RK45) method, used to solve ordinary differential equations (ODEs). It dynamically adjusts the step size during the solution process based on error estimates to maintain accuracy and efficiency.

- **Error Estimation**: Two approximations (one using a 4th-order method and another using a 5th-order method) are computed. The difference between these provides the error estimate. A large error indicates the step size is too large.
- **Step Size Adjustment**:

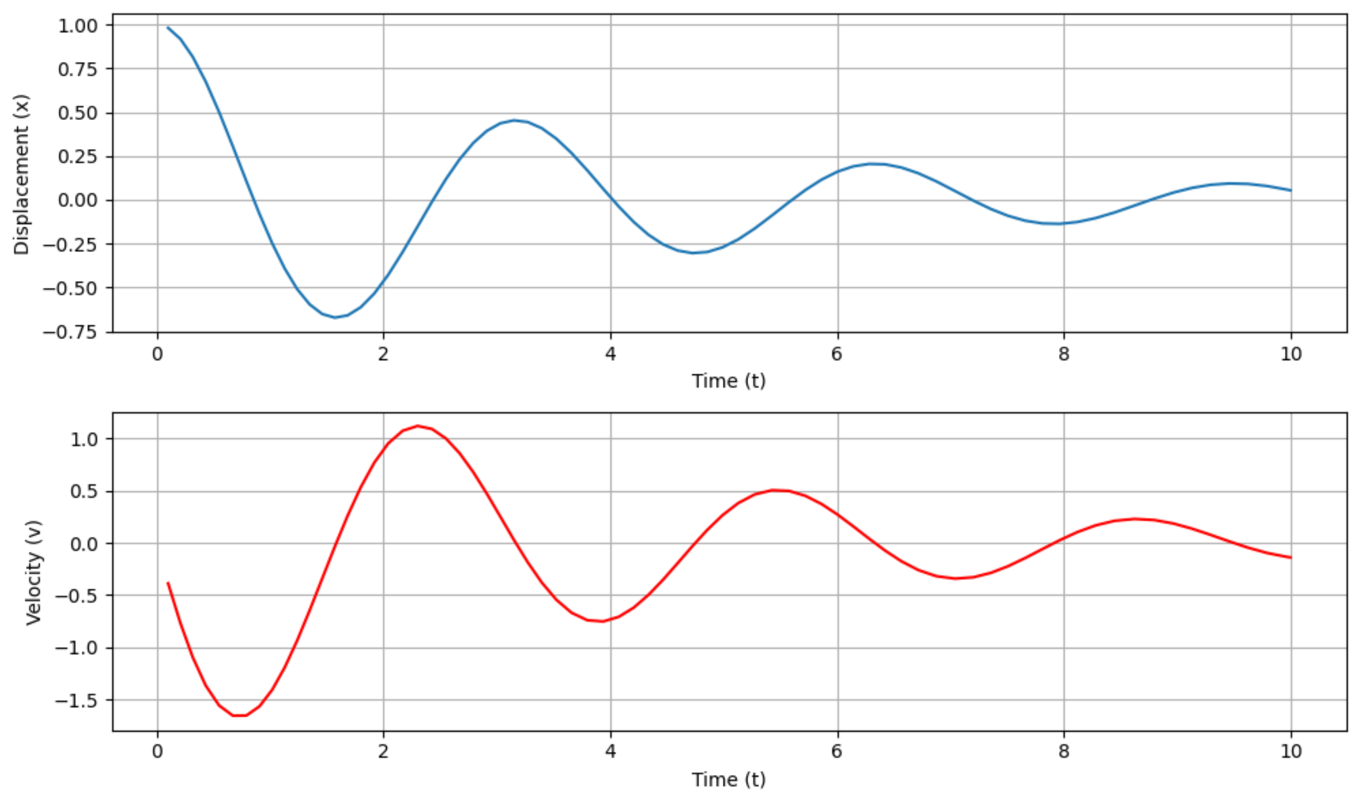
    - **Decrease step size** if the error exceeds tolerance, to improve accuracy.
    - **Increase step size** if the error is small, to improve efficiency by reducing the number of steps.
- **Tolerance and Efficiency**: The method balances accuracy and efficiency, using larger steps where the solution is smooth and smaller steps when precision is needed (e.g., for steep gradients).
- **Final Step Size Limitation**: The step size is limited to ensure the solution doesn't overshoot the final time.
- **Overall Process**: The algorithm computes solutions at each step, estimates errors, adjusts step size, and continues until the entire time span is covered.

Implementation

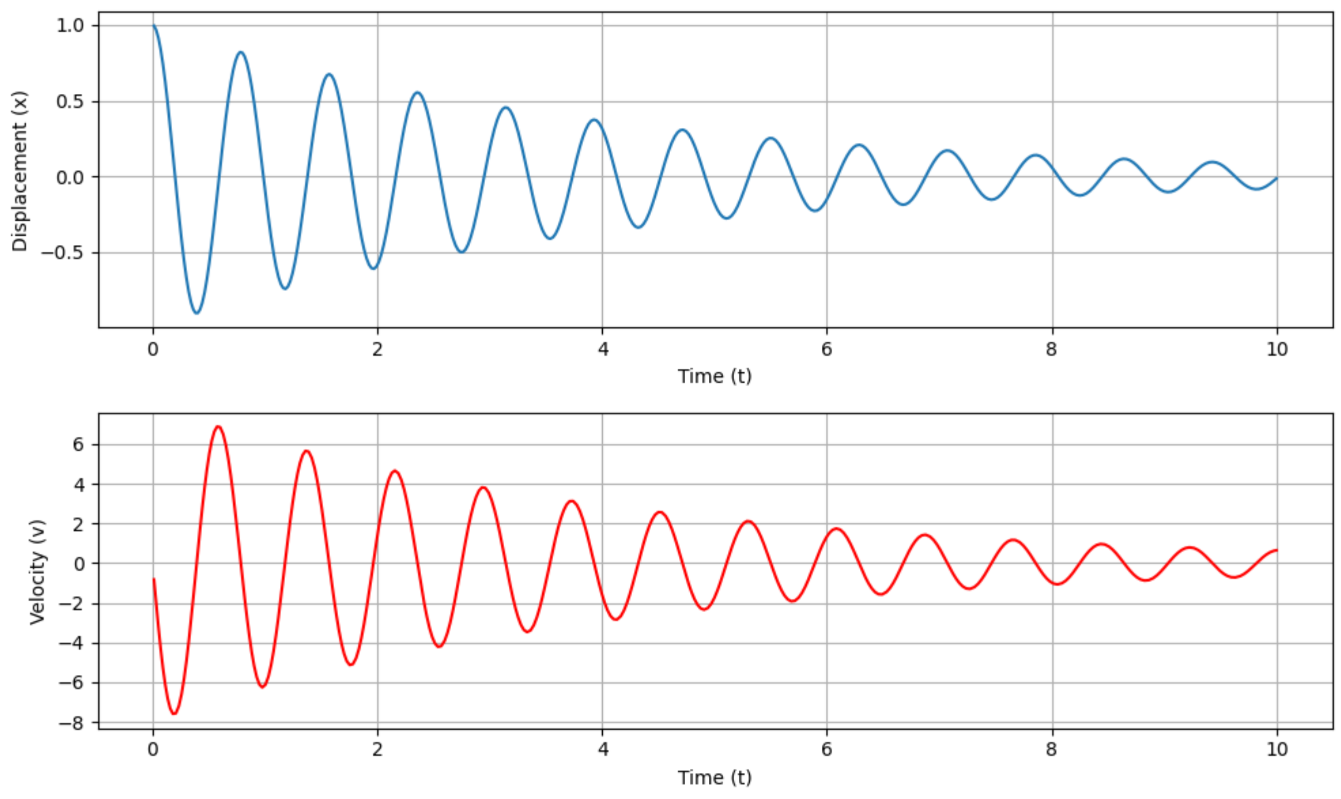
```
# Runge-Kutta 4th and 5th order method
def rk45_step(f, t, y, h, omega, beta):
    # Compute the Runge-Kutta intermediate steps
    k1 = h * f(t, y, omega, beta)
    k2 = h * f(t + h / 4, y + k1 / 4, omega, beta)
    k3 = h * f(t + 3 * h / 8, y + 3 * k1 / 32 + 9 * k2 / 32, omega, beta)
    k4 = h * f(t + 12 * h / 13, y + 1932 * k1 / 2197 - 7200 * k2 / 2197 + 7296 * k3 / 2197, omega, beta)
    k5 = h * f(t + h, y + 439 * k1 / 216 - 8 * k2 + 3680 * k3 / 513 - 845 * k4 / 4104, omega, beta)
    k6 = h * f(t + h / 2, y - 8 * k1 / 27 + 2 * k2 - 3544 * k3 / 2565 + 1859 * k4 / 4104 - 11 * k5 / 40, omega, beta)

    # Calculate the RK4 and RK5 results
    y4 = y + 25 * k1 / 216 + 1408 * k3 / 2565 + 2197 * k4 / 4104 - k5 / 5
    y5 = y + 16 * k1 / 135 + 6656 * k3 / 12825 + 28561 * k4 / 56430 - 9 * k5 / 50 + 2 * k6 / 55

    # Estimate error
    error = np.linalg.norm(y5 - y4)

    return y5, error


# Adaptive step size control
def solve_adaptive_rk45(f, t_span, y0, omega, beta, h_init=0.1, tol=1e-6):
    t0, tf = t_span
    y = np.array(y0)
    t = t0
    h = h_init
    solution = []

    while t < tf:
        # Take a step
        y_next, error = rk45_step(f, t, y, h, omega, beta)

        # Estimate the error and adjust step size
        if error < tol:
            # Accept the step
            t += h
            y = y_next
            solution.append([t, *y])  # Store time and values of x, v

        # Compute the next step size
        h_new = h * (tol / error) ** 0.25
        h = min(h_new, tf - t)  # Prevent overshooting the final time
    return np.array(solution)
```

Visualization

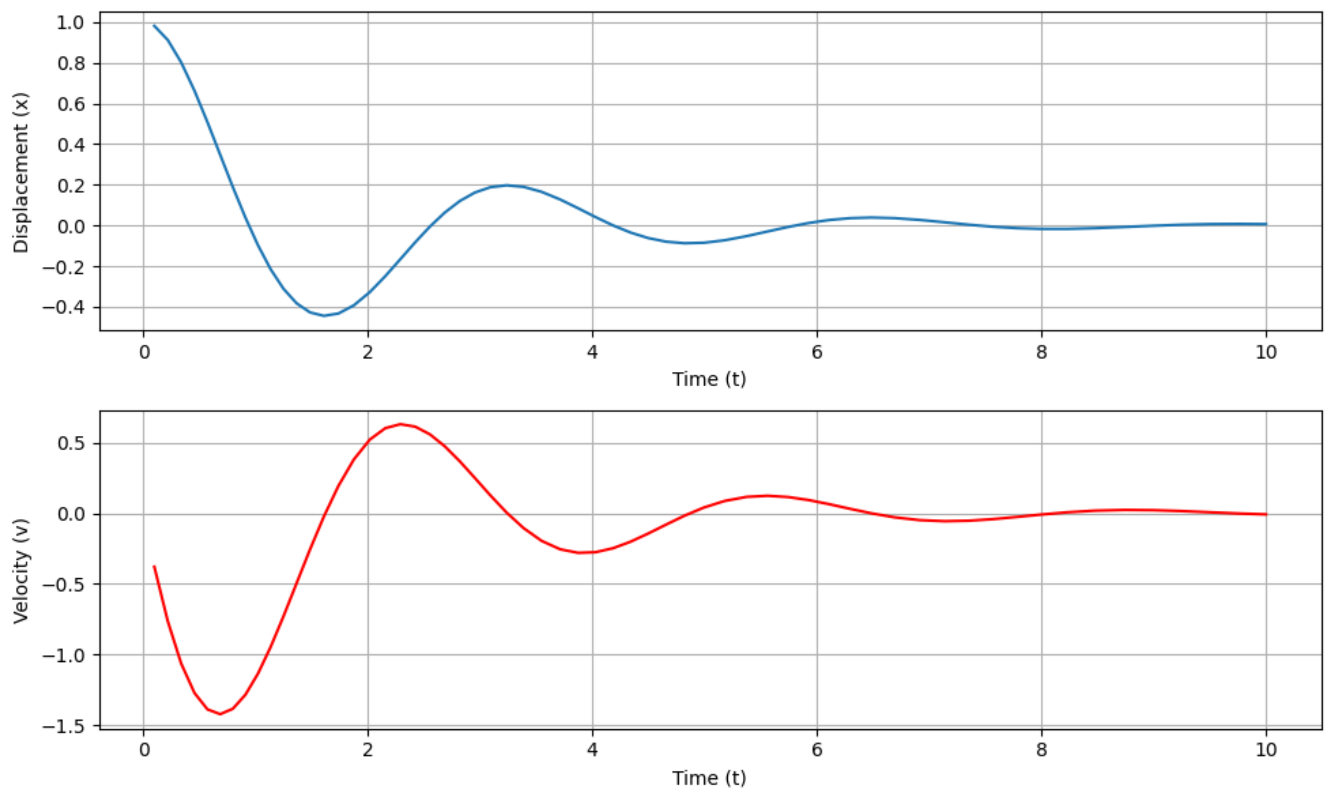
- `omega = 2.0`, `beta = 0.5`, `x0 = 1.0`, `v0 = 0.0`



- `omega = 8.0`, `beta = 0.5`, `x0 = 1.0`, `v0 = 0.0`



- `omega = 2.0`, `beta = 1.0`, `x0 = 1.0`, `v0 = 0.0`



Conclusion

In conclusion, adaptive step size control in the Runge-Kutta 4th/5th order method offers an efficient and accurate solution for complex ODEs like the damped harmonic oscillator. By adjusting the step size based on error estimates, it maintains accuracy while minimizing computations. The numerical results demonstrate its effectiveness in capturing the system's behavior, making it a balanced approach for improving both accuracy and efficiency in simulations.

Changing static constant parameters like frequency (ω) and damping factor (β) in a damped harmonic oscillator affects the system in the following ways:

1. Frequency (ω):

- **Increase:** Faster oscillations, shorter periods, and higher initial velocity.
- **Decrease:** Slower oscillations, longer periods, and lower initial velocity.

2. Damping Factor (β):

- **Increase:** Faster damping, reduced amplitude, and quicker stabilization.
- **Decrease:** Slower damping, larger amplitude, and more sustained oscillations.

In summary, changes in ω and β affect oscillation speed, amplitude, and how quickly the system stabilizes.