Coefficient of Restitution Lab (Curling)

Jaivir Parmar

February 2024

1 Discussion

1. From the curling experiment you conducted in lab, you have time and position data for a two-dimensional collision between two curling stones.

a) Describe how to approximate the velocities of the two stones over time using the given data and then write a script to do this.

To calculate the velocity of each slider at any given moment, it can be understood by evaluating the displacement of the slider's position with respect to the time elapsed during such displacement. To find the displacement, we can use the formula:

$$D = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

We can then find the velocity by taking the displacement over time:

$$v = \frac{D}{\Delta t}$$

A screenshot below illustrates how this can be accomplished with the data and variables provided.

Program 1: Velocity Calculator

```python
import numpy as np

def load_data(filename):
    data = []
    with open(filename, 'r') as file:
        for line in file:
            parts = line.strip().split(',')
            if len(parts) == 6:
                data.append([float(part) for part in parts])
    return data

def calculate_velocity(data, start_frame, end_frame):
    """
    Calculate the average velocity of each slider between the specified frames.
    """
    # Filter data for the specified frame range
    filtered_data = [row for row in data if start_frame <= row[0] <= end_frame]

    if len(filtered_data) < 2:
        return None  # Not enough data to calculate velocity

    # Initial positions and times
    _, time1, x1_s1, y1_s1, x1_s2, y1_s2 = filtered_data[0]
    # Final positions and times
    _, time2, x2_s1, y2_s1, x2_s2, y2_s2 = filtered_data[-1]

    delta_t = time2 - time1

    # Calculate displacements
    displacement_s1 = np.sqrt((x2_s1 - x1_s1)**2 + (y2_s1 - y1_s1)**2)
    displacement_s2 = np.sqrt((x2_s2 - x1_s2)**2 + (y2_s2 - y1_s2)**2)

    # Calculate velocities
    velocity_s1 = displacement_s1 / delta_t
    velocity_s2 = displacement_s2 / delta_t

    return velocity_s1, velocity_s2

# Example usage:
filename = 'data.txt'
data = load_data(filename)

# Enter the frames you want to assess the velocity between
start_frame = int(input("Enter the start frame: "))
end_frame = int(input("Enter the end frame: "))

velocity = calculate_velocity(data, start_frame, end_frame)

if velocity:
    print(f"Velocity of Slider 1 between frames {start_frame} and {end_frame}: {velocity[0]} EGR units/s")
    print(f"Velocity of Slider 2 between frames {start_frame} and {end_frame}: {velocity[1]} EGR units/s")
else:
    print("Not enough data to calculate velocity between the specified frames.")
```

This code loads data in from the text file, identifies the data variables in each line, and allows you to enter the two frames for which you'd like to find the velocity between. An example velocity calculation from frame 30 to 120 is shown below:

Output 1: Experimental Velocity Sample

```
Enter the start frame: 30
Enter the end frame: 120
Velocity of Slider 1 between frames 30 and 120: 2.962045911252895 EGR units/s
Velocity of Slider 2 between frames 30 and 120: 0.07713104822147632 EGR units/s
```

b) Consider how to choose time points just before and just after the collision and then write a script to do this.

Given that the second stone will have a velocity and position of zero before the collision and a non-zero velocity and position after the collision, we can identify the relevant point of collision by finding this transitional moment. The script below identifies the time points of just before and after the collision using the aforementioned strategy:

Program 2: Time Before and After Collision

```
def load_data(filename):
    data = []
    with open(filename, 'r') as file:
        for line in file:
            parts = line.strip().split(',')
            if len(parts) == 6:  # Ensure the line has the correct number of elements
                data.append([float(part) for part in parts])
    return data

def find_collision(data):
    for i in range(1, len(data)):
        frame_prev, time_prev, _, _, x2_prev, y2_prev = data[i-1]
        frame_curr, time_curr, _, _, x2_curr, y2_curr = data[i]

        # Check if Slider 2 has moved from its initial position
        if (x2_prev, y2_prev) == (0.0, 0.0) and (x2_curr, y2_curr) != (0.0, 0.0):
            return (frame_prev, time_prev), (frame_curr, time_curr)
    return (None, None), (None, None)

filename = 'data.txt'
data = load_data(filename)

collision_before, collision_during = find_collision(data)

if collision_before[0] is not None and collision_during[0] is not None:
    print(f"Prior to collision detected at frame {collision_before[0]} and time {collision_before[1]} seconds.")
    print(f"Collision detected at frame {collision_during[0]} and time {collision_during[1]} seconds.")
else:
    print("No collision detected.")
```

Output 2: Program 2 Output on Experimental Data

```
================== RESTART: /Users/jaivir/Downloads/impact.py ==================
Prior to collision detected at frame 73.0 and time 2.431874 seconds.
Collision detected at frame 74.0 and time 2.465188 seconds.
```

c) Present how to find the coefficient of restitution from the data and then write a script to do this. Clearly present your resulting coefficient of restitution value.

The coefficient of restitution can be calculated by using the velocities of the objects just before and just after the collision. The formula for the coefficient of restitution is given by:

$$e = \frac{v_2 after - v_1 after}{v_1 before}$$

We can simply combine the two codes from before to find the velocities at the point of impact, and then plug in those velocities into the equation above. The code for this and the result for our experiment is shown:

Program 3: COR Calculator

```python
import numpy as np

def load_data(filename):
    data = []
    with open(filename, 'r') as file:
        for line in file:
            parts = line.strip().split(',')
            if len(parts) == 6:
                data.append([float(part) for part in parts])
    return data

def calculate_velocity(data, frame):
    i = frame
    _, time1, x1_s1, y1_s1, x1_s2, y1_s2 = data[i-1]
    _, time2, x2_s1, y2_s1, x2_s2, y2_s2 = data[i]

    delta_t = time2 - time1
    velocity_s1 = np.sqrt((x2_s1 - x1_s1)**2 + (y2_s1 - y1_s1)**2) / delta_t
    velocity_s2 = np.sqrt((x2_s2 - x1_s2)**2 + (y2_s2 - y1_s2)**2) / delta_t

    return velocity_s1, velocity_s2


def find_collision(data):
    for i in range(1, len(data)):
        _, _, _, _, x2_prev, y2_prev = data[i-1]
        frame, _, _, _, x2_curr, y2_curr = data[i]

        if (x2_prev, y2_prev) == (0.0, 0.0) and (x2_curr, y2_curr) != (0.0, 0.0):
            return i
    return None

def calculate_coefficient_of_restitution(data):
    collision_frame = find_collision(data)
    if collision_frame is not None:
        v1_before, _ = calculate_velocity(data, collision_frame-1)
        v1_after, v2_after = calculate_velocity(data, collision_frame)

        e = (v2_after - v1_after) / v1_before
        return e
    else:
        return None

filename = 'data.txt'
data = load_data(filename)

e = calculate_coefficient_of_restitution(data)

if e is not None:
    print(f"The coefficient of restitution is: {e}")
else:
    print("Collision not detected or not enough data.")
```

Output 3: COR Experimental Result

```
================== RESTART: /Users/jaivir/Downloads/impact.py ==================
The coefficient of restitution is: 0.4132142822253344
```