EGR 244: Coefficient of Restitution Lab

Jaivir Parmar

February 25 2024

1 Post-Lab Questions

1. In lab, you collected data from a ball drop experiment for three different balls. For each ball:

a) determine the velocity immediately before the first bounce using a specified model and experimental data.

We can understand the velocity of any velocity using the conservation of energy, such that:

$$mgh = 1/2mv^2$$

solving for v and simplifying to:

$$v = \sqrt{2gh}$$

With known heights (inches) of 4, 5, 6, 7, 8 (metal); 5, 6, 7, 8, 9 (golf); and 10, 11, 12, 13, 14 (Lacrosse), we convert these measurements to meters to get the following results from the aforementioned equation:

	Metal				
Drop Height (m)	0.1016	0.127	0.1524	0.1778	0.2032
Calculated Velocity (m/s)	1.41187534860553	1.57852462761909	1.72918709224884	1.86773552731643	1.99669326637819

Golf				
0.127	0.1524	0.1778	0.2032	0.2286
1.57852462761909	1.72918709224884	1.86773552731643	1.99669326637819	2.1178130229083

Lacrosse				
0.254	0.2794	0.3048	0.3302	0.3556
2.23237093691886	2.34133039103839	2.44543983773881	2.54529448198043	2.64137691365697

We can compare this to the experimental values by following these steps:
1. Read the CSV file into a pandas DataFrame.
2. Identify the peak height before the first bounce (maximum negative height value).
3. Convert this height to meters.
4. Calculate the initial velocity just before the first bounce using the formula

Trial	Metal (m/s)	Golf (m/s)	Lacrosse (m/s)
1	0.8570676473	1.60016	1.804634085
2	0.8787051117	1.663078892	1.955842337
3	0.9065185337	1.685107386	1.998813627
4	0.9236729811	1.861053089	2.139818082
5	0.9855355153	1.905639493	2.204402719

```python
import pandas as pd
import numpy as np

data_path = '/Users/jaivir/Downloads/EGR244Data/L5.csv'
df = pd.read_csv(data_path, header=None, names=['time (s)', 'drop height (in)'])

df['drop height (in)'] = df['drop height (in)'].abs()

peak_height_in_inches = df['drop height (in)'].max()

peak_height_meters = peak_height_in_inches * 0.0254

g = 9.81
initial_velocity = np.sqrt(2 * g * peak_height_meters)

print(f"Peak Height: {peak_height_in_inches} inches")
print(f"Peak Height: {peak_height_meters} meters")
print(f"Initial Velocity: {initial_velocity} m/s")
```

Deviances in calculated velocities may be explained by small measurement errors.

b) derive a method to determine the velocity before each bounce from your experimental data and then write a script to do this.

We follow a simple process to create a code to accomplish this task:
1. Import Necessary Libraries: Import pandas for handling the CSV file and data manipulation, scipy.signal for finding peaks in the noisy data, and numpy for mathematical operations.
2. Define Constants: Define the acceleration due to gravity in inches per second squared (converting from the standard 9.8 m/s) for consistency with your measurements) and then eventually converting velocities to m/s for the final output.
3. Load CSV File: Read the CSV file into a pandas DataFrame, assuming the file has two columns without headers: the first column for time in seconds and the second for height in inches.
4. Preprocess Data:

Convert the height from inches to meters for the final velocity calculation, as you requested velocities in m/s. Find Peaks with Conditions:

5. Use the findpeaks function from scipy.signal to identify the indices of peaks in the height data. Apply conditions to only consider peaks that are above 1 inch in height and ensure that each peak considered is at least 0.1 seconds after the previous peak by using the distance parameter, calculated based on the sampling rate and the time difference threshold. Calculate Velocities:

6. For each identified peak, calculate the velocity immediately before the bounce using the formula

Prepare Results for Display:

7. Create a new DataFrame that includes the time of each bounce and the corresponding velocity before the bounce in m/s. Display Results:

8. Print the resulting DataFrame to show the time of each bounce and the velocity immediately before it in meters per second.

The results and the supporting code are shown below:

Trial	Metal (m/s)	Golf (m/s)	Lacrosse (m/s)	Trial	Metal (m/s)	Golf (m/s)	Lacrosse (m/s)
1.1	0.49138	1.53161	1.80371	2.1	0.63266	1.65217	1.95485
1.2	0.51897	1.41521	1.51478	2.2	0.59201	1.66223	1.59592
1.3	0.51367	1.34505	1.169	2.3	0.46052	1.60974	1.29623
1.4	0.45835	1.34171	1.00356	2.4	0.45454	1.60292	1.04225
1.5	0.44232	1.47025	0.77195	2.5	0.44847	1.57851	0.87075
1.6	0.39851	1.46652	0.73258	2.6	0.5313	1.54729	0.70558
1.7	0.46052	1.50008	0.755	2.7	0.50438	1.49143	0.60543

Trial	Metal (m/s)	Golf (m/s)	Lacrosse (m/s)	Trial	Metal (m/s)	Golf (m/s)	Lacrosse (m/s)
3.1	0.82375	1.68425	1.99779	4.1	0.65128	1.8601	2.13873
3.2	0.87702	1.60401	1.64885	4.2	0.62235	1.67877	1.71137
3.3	0.52232	1.60168	1.32096	4.3	0.60285	1.73018	1.41063
3.4	0.6819	1.45698	1.09148	4.4	0.59201	1.5837	1.03169
3.5	0.46429	1.36762	0.79421	4.5	0.39851	1.50554	0.81249
3.6	0.50635	1.42502	0.80014	4.6	0.59829	1.54375	0.83396
3.7	0.55333	1.33818	0.7964	4.7	0.49829	1.32922	0.81493

Trial	Metal (m/s)	Golf (m/s)	Lacrosse (m/s)
5.1	0.98503	1.89063	2.203279
5.2	0.66377	1.90467	1.742363
5.3	0.47646	1.89063	1.387676
5.4	0.39094	1.87397	1.021749
5.5	0.72231	1.44428	0.93499
5.6	0.41443	1.24989	0.933125
5.7	0.56314	0.91698	0.834851

```python
import pandas as pd
from scipy.signal import find_peaks
import numpy as np

g_meters = 9.8
inches_to_meters = 0.0254

df = pd.read_csv('/Users/jaivir/Downloads/EGR244Data/M1.csv', header=None, names=['time', 'height'])

df['height'] *= inches_to_meters

min_height_meters = 1 * inches_to_meters
indices, _ = find_peaks(df['height'], height=min_height_meters, distance=0.1*5000)

velocities = np.sqrt(2 * g_meters * df.loc[indices, 'height'])

results = pd.DataFrame({ 'Velocity Before Bounce (m/s)': velocities})

for velocity in velocities[:7]:
    print(f"{velocity:.5f}")
```

c) derive a method to compute the coefficient of restitution from each bounce

(for all experiments with that ball) and then write a script to do this.

The coefficient of restitution for each ball can be calculated using the expression:

$$e = \sqrt{h_{n+1}/h_n}$$

From here, we may calculate the COR using the following steps:

1. Identify entrance and exit velocity of the ball.

2. Calculate COR for Each Bounce: Use the v_entrance/v_exit formula the COR for each pair of consecutive bounces.

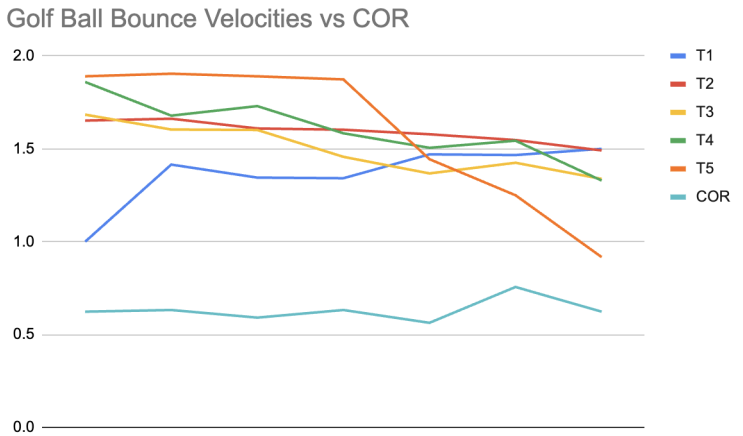
3. Average COR: To get a single COR value for each ball, calculate the average of the COR values obtained from all the bounces.

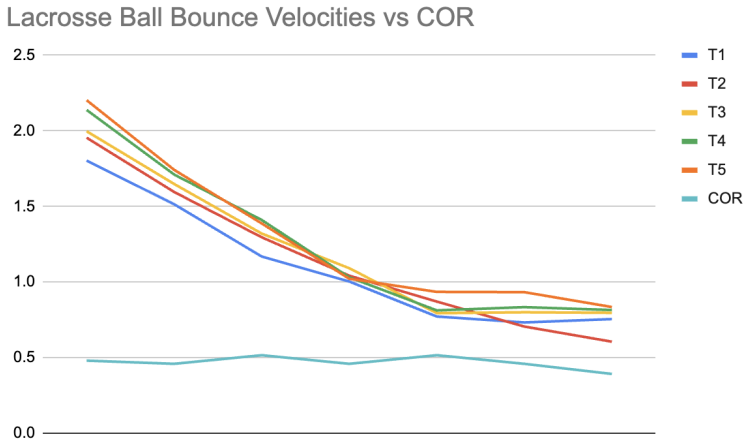
The average coefficient of restitution for the golf ball was 0.645, 0.574 for the metal ball, and 0.459 and for the lacrosse ball. The code for these calculations is displayed below:

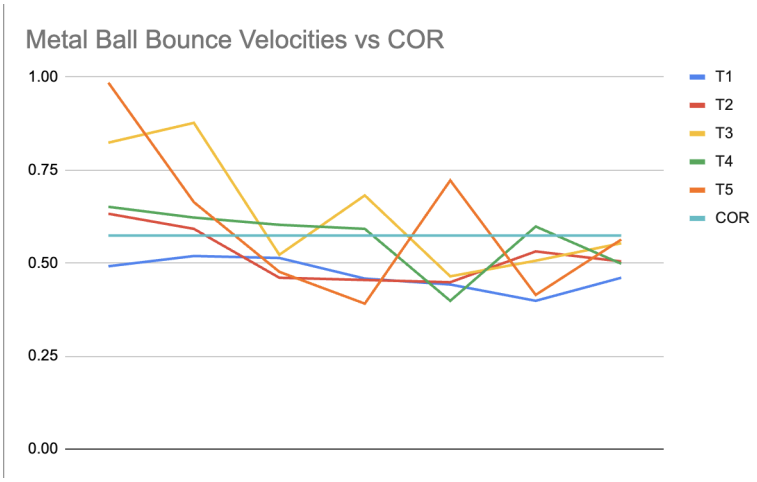
```
import pandas as pd
import numpy as np
from scipy.signal import find_peaks
from scipy.interpolate import interp1d

# Load data, assuming no column labels and the first column is time, second is height in inches
df = pd.read_csv('/Users/jaivir/Downloads/EGR244Data/M5.csv', header=None)
df.columns = ['time', 'height_inches']

# Convert height from inches to meters for calculation
df['height_meters'] = df['height_inches'] * 0.0254

# Constants
g = 9.8  # Acceleration due to gravity, m/s^2

# Find peaks with a minimum horizontal distance of 0.1s between them
sampling_rate = 5000  # Update this if your actual sampling rate differs
min_samples_between_peaks = int(0.1 * sampling_rate)
peaks, _ = find_peaks(df['height_meters'], distance=min_samples_between_peaks)

# Ensure we only process up to the first 7 peaks
peaks = peaks[:7]

# Initialize list to store COR values
cors = []

for i, peak in enumerate(peaks):
    # Calculate the initial velocity (before the bounce)
    h_initial = df['height_meters'].iloc[peak]
    v_initial = np.sqrt(2 * g * h_initial)

    # Find the time 0.05s after the peak and interpolate to find the corresponding height
    time_peak = df['time'].iloc[peak]
    time_exit = time_peak + 0.05
    # Ensure time_exit is within the bounds of our data
    if time_exit <= df['time'].iloc[-1]:
        # Linear interpolation to find height at time_exit
        interpolator = interp1d(df['time'], df['height_meters'], kind='linear')
        h_exit = interpolator(time_exit)

        # Calculate exit velocity
        v_exit = np.sqrt(2 * g * h_exit)

        # Calculate COR for this bounce
        cor = v_exit / v_initial
        cors.append(cor)

# Calculate and print average COR if we have calculated any CORs
if cors:
    average_cor = np.mean(cors)
    print(f"Average Coefficient of Restitution: {average_cor:.3f}")
    for i, cor in enumerate(cors, start=1):
        print(f"COR for bounce {i}: {cor:.3f}")
else:
    print("No valid bounces found for COR calculation.")
```

d) plot the coefficient of restitution vs the pre-bounce velocity on a single graph (for all bounces during all experiments for that ball).

The graphs for all balls are depicted below:







e) comment on how the computed coefficient of restitution does or does not depend on the pre-bounce velocity.

there was no obvious relationship between velocity and COR::

Material Non-Linearity: Real materials do not always behave in a perfectly elastic manner, and their response can be non-linear, especially at high strain rates. For balls made of materials like rubber, the deformation and energy dissipation mechanisms can vary with the impact velocity, leading to variations in COR at different speeds.

Energy Dissipation Mechanisms: At higher velocities, additional forms of energy dissipation might become significant. For example, internal friction, air resistance, and heat generation within the material can play a larger role, potentially reducing the COR at higher impact velocities.

Surface Deformation: The COR can also be influenced by the deformation of the collision surface. Hard surfaces might not deform much, but their microscopic or even macroscopic response could change with the impact energy, slightly affecting the COR.

Ball Deformation: The extent and nature of the deformation of the ball upon impact can vary with velocity. At higher velocities, a ball might compress more, engage more material in the deformation process, and thus experience different restitution characteristics.