Assignment6_hheyen2s

October 13, 2023

1 Scientific Programming with Python

1.1 Assignment 6: Galileo's Horizontal Ball Drop

1.1.1 Helena Heyen 9031786

Link to the Assignment

This assignment is about the Ball Drop Experiment conducted by Galileo in 1608, where a ball was released from different heights on an incline and its horizontal distance traveled was measured in units of "Galileo Points". The objective is to compare the experiment data to data computed by two equations for calculating the distance, one obtained from Galileo's writings, the other an adjusted formula to describe the same experiment. The solution should be developed using data interpolation/extrapolation with SciPy and presented using Matplotlib.

[2]:
```
import math
import matplotlib.pyplot as plt
from scipy.interpolate import interp1d
```

Task 1: Create user-defined functions for following equations:

[3]:
```
def galileo_distance_1 (height):
    return 2 * (5/7)**0.5 * height
```

[4]:
```
def galileo_distance_2 (height):
    return -0.0013 * (height**2) + 2.69 * height + 32.18
```

Task 2: Compute the Root-Mean-Square-Error of the experiment values versus calculated values by equations 1 and 2: Fist, I provide a method for calculating the RMSE according to the formula given in the assignment.

[5]:
```
def rmse_calculator (calculated, experiment):
    """
    Calculates the Root Mean Squared Error (RMSE) between calculated values and␣
 ↪experimental values.

    Parameters:
        calculated: List of calculated values.
        experiment: List of experimental values.
```

```
    Returns:
        float: Root Mean Squared Error (RMSE) value.
    """
    if len(calculated) == len(experiment):
        mean_squared_error = sum([(calculated - experiment)**2
                                  for calculated, experiment in zip(calculated,␣
 ↪experiment)]) / len(calculated)
    return math.sqrt(mean_squared_error)
```

Using the heights from the experiment the corresponding distances are calculated using Equations 1 and 2. Both the calculated values and experimental values are then passed to the previously defined method that calculates the Root Mean Squared Error (RMSE). This RMSE value represents the average distance between the two groups of values [1], both measured in Galileo Points.

[6]:
```
experiment_height = [300, 600, 800, 828, 1000]
experiment_distance = [800, 1172, 1328, 1340, 1500]

calculated_distance_1 = [galileo_distance_1(height) for height in␣
 ↪experiment_height]
calculated_distance_2 = [galileo_distance_2(height) for height in␣
 ↪experiment_height]

rmse_1 = rmse_calculator(calculated_distance_1, experiment_distance)
rmse_2 = rmse_calculator(calculated_distance_2, experiment_distance)

print ("Galileo's measurements vs. Equation 1:", rmse_1)
print ("Galileo's measurements vs. Equation 2:", rmse_2)
```

```
Galileo's measurements vs. Equation 1: 173.8129935702819
Galileo's measurements vs. Equation 2: 52.0236722745329
```

The results show that Equation 2 exhibits a smaller deviation from the experimental values compared to Equation 1. This suggests that Equation 2 provides a more accurate representation of Galileo's experiment, indicating a closer alignment between the calculated values and the experimental measurements.

Task 3: Utilize the 'interp1d' function from SciPy to interpolate and extrapolate new data based on the experiment values. Generate data using different kinds of extrapolations: 'linear', 'quadratic', 'cubic', 'nearest', 'previous', and 'next'. A function variable is defined for each function obtained based on the experiment values and the specified property. Setting the "bounds_error" marker to False allows for a continuous extrapolation exceeding the bounds of the input data [2].

[7]:
```
f_linear = interp1d(experiment_height, experiment_distance, kind='linear',␣
 ↪bounds_error=False)
f_quadratic = interp1d(experiment_height, experiment_distance,␣
 ↪kind='quadratic', bounds_error=False)
```

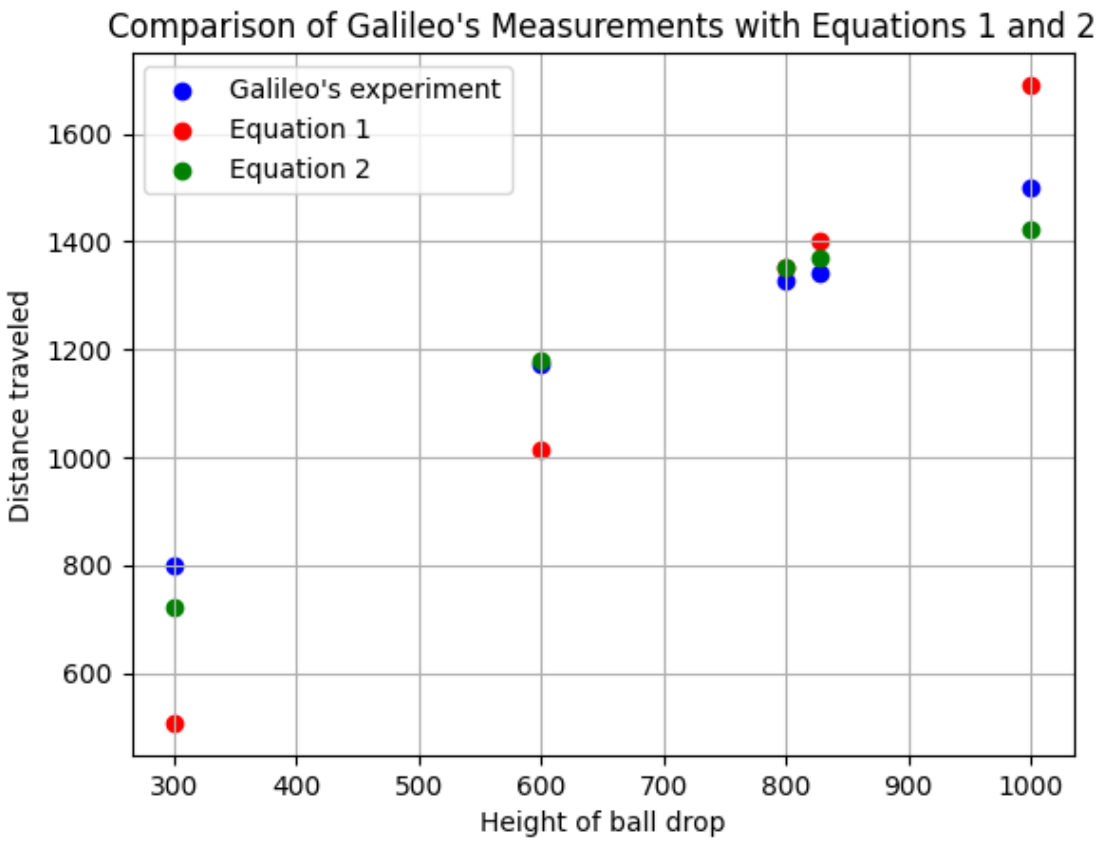
```
f_cubic = interp1d(experiment_height, experiment_distance, kind='cubic',␣
 ↪bounds_error=False)
f_nearest = interp1d(experiment_height, experiment_distance, kind='nearest',␣
 ↪bounds_error=False)
f_previous = interp1d(experiment_height, experiment_distance, kind='previous',␣
 ↪bounds_error=False)
f_next = interp1d(experiment_height, experiment_distance, kind='next',␣
 ↪bounds_error=False)
```

Use an input height range from 0 to 1000 "points" for the 'x' parameter to obtain both interpolated and extrapolated data.

[8]:
```
height_range = list(range(0, 1001))
y_linear = [f_linear(x) for x in height_range]
y_quadratic = [f_quadratic(x) for x in height_range]
y_cubic = [f_cubic(x) for x in height_range]
y_nearest = [f_nearest(x) for x in height_range]
y_previous = [f_previous(x) for x in height_range]
y_next = [f_next(x) for x in height_range]
```

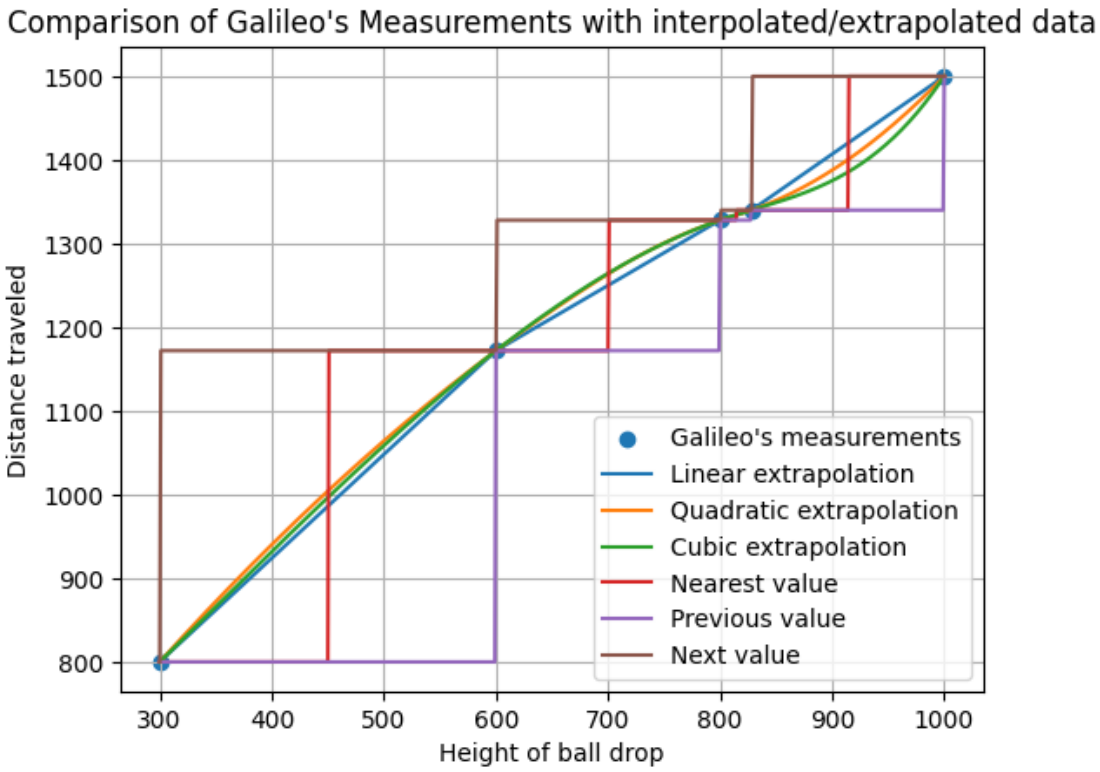
Task 4: Plot Galileo's experimentally measured data alongside the data from Equations 1 and 2 as well as the new data generated in Task 3.

[9]:
```
distance_plot, ax = plt.subplots()
ax.scatter(experiment_height, experiment_distance, label="Galileo's␣
 ↪experiment", c='blue')
ax.scatter(experiment_height, calculated_distance_1, label='Equation 1',␣
 ↪c='red')
ax.scatter(experiment_height, calculated_distance_2, label='Equation 2',␣
 ↪c='green')
ax.legend()
plt.xlabel('Height of ball drop')
plt.ylabel('Distance traveled')
plt.title("Comparison of Galileo's Measurements with Equations 1 and 2")
plt.grid()
plt.show()
```



The plot clearly illustrates that Equation 2 exhibits a closer alignment with the experimental values.

[10]:
```
interpolation_plot, ax = plt.subplots()
ax.scatter(experiment_height, experiment_distance, label="Galileo's␣
 ↪measurements")
ax.plot(height_range, y_linear, label='Linear extrapolation')
ax.plot(height_range, y_quadratic, label='Quadratic extrapolation')
ax.plot(height_range, y_cubic, label='Cubic extrapolation')
ax.plot(height_range, y_nearest, label='Nearest value')
ax.plot(height_range, y_previous, label='Previous value')
ax.plot(height_range, y_next, label='Next value')
ax.legend()
plt.xlabel('Height of ball drop')
plt.ylabel('Distance traveled')
plt.title("Comparison of Galileo's Measurements with interpolated/extrapolated␣
 ↪data")
plt.grid()
plt.show()
```



It can be observed, that linear, quadratic and cubic interpolation/extrapolation methods approximate the values between and beyond the data points, creating a straight or curved connecting line, suggesting a steady trend. The 'nearest', 'previous' and 'next' methods continue the trend of the nearest, previous and next data point, assuming the respective value as an adequate approximation [2], which provides a rather inaccurate representation in this particular case. Based on the results from Tasks 2 and 3, one can assume that a quadratic polynomial such as Equation 2 approximates Galileo's experiment best within the given range.

1.1.2 References:

[1] https://statisticsbyjim.com/regression/root-mean-square-error-rmse/ [2] https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.interp1d.html