PHY1112 Assignment 9

Project Fly Ball

Assigned: March 12th, 2024

Due: March 26th, 2024

|  |  |  |  |
| --- | --- | --- | --- |
| Part | 1 | 2 | Total |
| Points | 17 | 17 | 34 |
| Score |  |  |  |

Objectives

1. Use numerical differentiation to determine what planet you have woken up on, utilizing nothing but a baseball and your observations.
2. Check your work using numerical integration – you did just wake up on another planet after all!

Part 1: Something Feels *Different…* Where Am I?

You wake up in a room with no recollection of why or how you got here. You stand up from the floor and notice that movement feels *different*. “Huh,” you think to yourself, “I don’t look like I’ve lost weight, but standing up feels surprisingly much easier than I remember.” You sigh, and look around the room for some sort of indication as to where you could be. There are no windows, no doors, no… anything?

Well, there is *something*. Technically, some*things*. Three somethings, to be precise: a baseball, a stop watch, and a laptop. “How… convenient,” you think to yourself suspiciously, “this reminds me of that Physics course I took in my first year of undergraduate. Yea I remember that! The assignment where we pretended to observe the motion of a baseball to determine the gravitational constant of an unknown planet!” A sharp sense of concern jolts through your excited reminiscing. “Wait a minute… Does that mean… that I am… currently on ANOTHER PLANET!?”

“Why? HOW? What do I do?” you hopelessly ask to the room. Unfortunately for you, the baseball, stopwatch and laptop can’t answer two of those questions, but they can help you with the third one. You recall the first year assignment, “Right, I supposed all I *can* do is use these things to determine what planet I’m even on. It may not be much help, but at least it’s a start.”

Upon turning on the laptop, you notice that the only software installed on it is VSCode and Python 3.11, with the numpy and matplotlib packages already prepared. “I can’t help but feel like this is a setup,” you remark before quickly pulling yourself away from that line of thinking – you know that you need to focus on your task at hand. “Well, no better time than now to get started, I suppose.”

“Alright,” you mutter as you look at the baseball in one hand and stopwatch in the other, “the acceleration due to gravity is a constant, relatively speaking. But what things do I need to know to be able to determine it?” You look down at your laptop, “if I remember correctly, the one-dimensional kinematic equation should be:

where is the height of the baseball at any time[[1]](#footnote-2), and the initial height and upward velocity of the baseball, and,” you gasp, “AH! Perfect! The here is the acceleration due to gravity! Just what I need!”

Glancing over at the wall, you see there are multiple meter scales painted on it, with a 4 decimal point accuracy. “It’s really starting to seem like measuring the period of a pendulum would be a lot easier,” you cleverly remark, but there’s no TA to hear your complaints in this empty room.

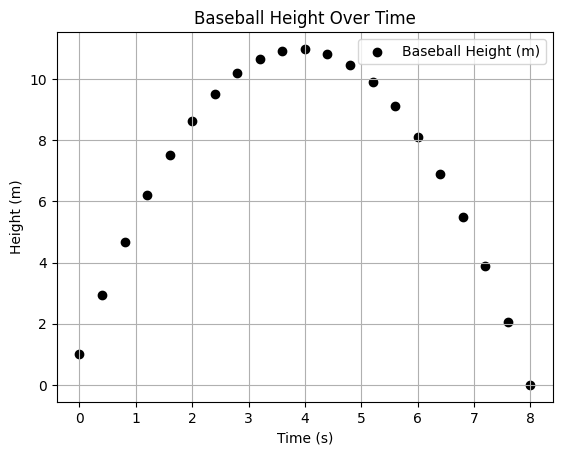
Aiming the camera of the laptop at this wall, you hit record at the same time you start the stopwatch within the frame, and then you throw the ball straight up into the air, and catch it again a few seconds later. A sense of pride strikes you, “That was a very vertical throw! I don’t think the ball moved laterally at all!” After carefully reviewing the footage and tabulating the height and time information of the baseball into the ‘baseball\_height.csv’ file.

1. (4 points)
   1. Create a file titled ‘Assignment9.py’ and load the ‘baseball\_height.csv’ data file into ndarrays using the np.genfromtxt() function.

A black background with white numbers

Description automatically generated

* 1. To get a better picture of what the data looks like, plot it using matplotlib with markers for the points, rather than lines, since you know that you’re interested in the discrete nature of the data in this case.



**Figure 1.** A scatterplot representation of the height of the baseball over time on the unknown planet. Each point represents the height (y-axis) at the corresponding time (x-axis).

* 1. “Interesting, the trend of this data look kind of like a parabola How odd, I remember that being my answer to question 1 part (c) from that assignment all those years ago…”

“Now that I have the data in Python, it should be a simple task to get out the value of g… How do I do that again?” Suddenly, you’re struck with the memory of an incredibly useful tool from rudimentary calculus: the derivative. “Yes, that’s it!” you exclaim, “I simply need to differentiate my equation from above! Let’s try that – so the first derivative is

,

but this isn’t it just yet. I need only , not . Well let’s try to differentiate once more

.

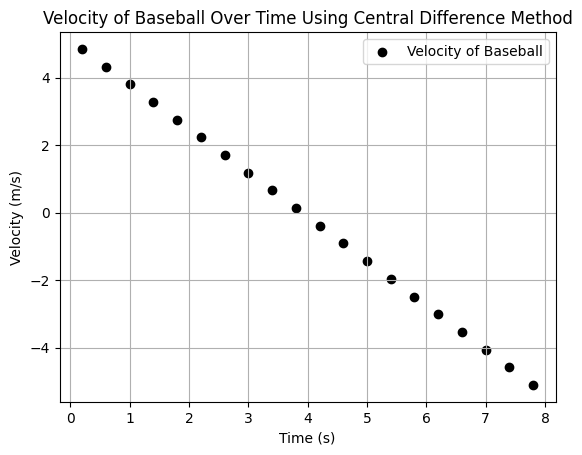
The joy of physics washes over you, “I’ve done it!” But the rush is fleeting. “Well… I’ve *almost* done it. I have this data, and it is discrete data, which means that it isn’t exactly the analytical function . Instead it’s more similar to evaluated at specific points. How am I supposed to take a derivative of a discrete data set?” Memories of the course come to mind again, “Discrete derivatives! Finite differences! I remember those, yes! They’re like the equivalent of analytical derivatives but for discrete data. This is perfect. But, there are multiple different types, right?”

“Yes, there’s forward, backward, and central differences.” You think for a moment, “I suppose I should use a central difference.”

1. (6 points)
   1. Write a Python function called central\_difference\_derivative() that computes the numerical derivative of a data set using central differencing. To be efficient when working with ndarrays, remember to write this custom function in a vectorized fashion.
   2. Use your central\_difference\_derivative() function to calculate the numerical derivative of your height vs time data, . Plot using markers for the data points rather than lines.

A black screen with white text

Description automatically generated



**Figure 2.** A scatterplot representation ofthe velocity of the baseball over time on the unknown planet using a central difference. Each point represents the velocity (y-axis) at the corresponding time (x-axis).

“Wow, now the data appears to be linear, which matches the equation for .”

* 1. To make sure you calculated the time points correctly, observe your first and last time point, and count how many time points you have for the differentiated data set.

“Oh perfect, my first time point is at 0.2, while my last time point is at 7.8, and I have 20 time points overall.”

“Now I know the velocity information of the baseball, and I also know the position information of the baseball from my original observation; it’s time to finally get this acceleration information! I know that I need to calculate another discrete derivative.”

“I’m glad that I remember how my range for time keeps getting smaller, otherwise I may be trying to calculate points using entries outside of the range of my original data set,” you acknowledge.

1. (6 points)
   1. Use your central\_difference\_derivative() function to calculate the numerical derivative of that you determined in question 2. Plot , using markers for the data points rather than lines.

A black screen with white text

Description automatically generated

A graph with numbers and dots

Description automatically generated

**Figure 3.** A scatterplot representation ofthe acceleration of the baseball over time on the unknown planet using a second central difference. Each point represents the acceleration (y-axis) at the corresponding time (x-axis).

“Wow, now the data appears to be constant, which makes sense physically, given that is a constant.

* 1. To make sure your data is correct, again check your first and last time point, and the total number of time points.

“Seems that my number of time points decreased by 1 again. Interestingly, the first time data point for is at 0.4, and the last time data point is at 7.6!”

How does this compare to the time points in the data set?

There is a missing time point at both the start and end of the data set, but otherwise the time data is preserved.

* 1. Turning your attention back to your values for you realize excitedly, “I finally know what the value of where I am is! According to my calculations, the value of here is -1.31m/s2”
  2. You notice that there is a sticky note attached to the underside of the laptop. The sticky note has the following table of some solid celestial bodies along with their accelerations due to gravity on their surfaces:

|  |  |
| --- | --- |
| Celestial body | Acceleration due to gravity at their surface |
| Earth | 9.82 m/s2 |
| Ceres (Dwarf planet) | 0.28 m/s2 |
| Europa (Jupiter’s moon) | 1.31 m/s2 |
| Enceladus (Saturn’s moon) | 0.11 m/s2 |
| Titan (Saturn’s moon) | 1.35 m/s2 |
| Triton (Neptune’s moon) | 0.77 m/s2 |

“Based on this table it seems like I must be on Europa!? Seriously?”

**(17 marks total, 1 for docstrings/file header/variable naming/comments**

Part 2: *Sum*-thing Still Feels Wrong, I Should Check My Work

“There’s no way I could possibly be on that celestial body. I must have made a mistake in my calculations. But how can I check them?”

Recollecting the memories from that undergrad assignment again, you consider if there are any tools you were given that you could use to check your work. “That’s it! If I did all of my calculations correctly, then performing a numerical integration on my gravity data should result in the same shape of plot as I saw in my original data set. Surely my mistake will be clear after doing that.” You remember that the integral you need to solve has the form

Reflecting on numerical integration methods, you remember that most of them are not so simple as numerical differentiation, so you decided that a good balance between simplicity and error would be to apply the Trapezoidal rule to calculate the integral

* 1. (6 points) Write a custom vectorized function that solves the integral above as a function of time, called trap\_integration(). It takes as input two 1D NumPy arrays containing the data set for , and uses the trapezoidal rule to calculate the integral as a function of time, as in the formula in Eq. 1 above, which is returned as a 1D array.

Hint: use np.cumsum() to perform the cumulative sum.

* 1. (3 points) Apply trap\_integration()to your data set from question 3 to calculate

What is ? Note that , as per the formula given in Eq. 1 above.

A black screen with white text

Description automatically generated

* 1. (3 points) Apply trap\_integration()to your data set from question 2 to calculate

What is now? Note that , as per the formula given in Eq. 1 above.

A black screen with white text

Description automatically generated

* 1. (4 points) Plot the data sets and you obtained in parts b) and c) in two separate graphs. Compare these to your graphs of data sets and in Part 1. The plots should give the correct shapes, but they may be vertically off-set from the true values, since we have not included the constants.

A graph of a baseball

Description automatically generated

**Figure 4.** A scatterplot representation ofthe velocity of the baseball over time on the unknown planet using an integral from the second central difference. Each point represents the velocity (y-axis) at the corresponding time (x-axis).

A graph of a baseball height

Description automatically generated

**Figure 5.** A scatterplot representation ofthe height of the baseball over time on the unknown planet using a consecutive integration from the second central difference. Each point represents the height (y-axis) at the corresponding time (x-axis).

A graph with black dots

Description automatically generated

**Figure 6.** A scatterplot representation ofthe height of the baseball over time on the unknown planet using a consecutive integration from the second central difference with an adjusted initial velocity value. Each point represents the height (y-axis) at the corresponding time (x-axis).

This affirms your fear. “I really am on Europa. But why?”

“Now what? Sure, I may be on Europa, but that doesn’t tell me how I got here, and what I could possibly be doing here?” Looking at the list of celestial bodies, you wonder to yourself “Is there something that relates all of these celestial bodies to each other?” No matter how hard you try to solve the problem, you feeling like you’re *drowning* in possibilities. *Drowning*… interesting...[[2]](#footnote-3)

**(17 marks total, 1 for docstrings/file header/variable naming/comments**

**CODE:**

'''

Filename:       Assignment9.py

Author:         Patrick Geraghty

Date Created:   2024-03-18

Date Modified:  2024-03-18

Description:    Contains relevant functions for Assignment 9.

'''

import numpy as np

import matplotlib.pyplot as plt

# Part 1

*def* load\_data():

    '''

    () -> (np.array, np.array)

    Load the data from the file and return the x and y values as arrays.

    Preconditions: The file is in the same directory as the program.

    '''

    # Load the data from the file

    data = np.genfromtxt('baseball\_height.csv', *delimiter*=',')

    # Split the data into x and y arrays

    x\_data = data[:, 0]

    y\_data = data[:, 1]

    return x\_data, y\_data

*def* height\_plot():

    '''

    () -> None

    Create a scatter plot of the data from 'baseball\_height.csv'.

    Preconditions: None

    '''

    # Load the data from the file

    x\_data, y\_data = load\_data()

    # Initialize the first figure

    plt.figure(1)

    # Create a scatter plot of the data with relevant labels and titles

    plt.scatter(load\_data()[0], load\_data()[1], *c*='k', *marker*='o', *label*='Baseball Height (m)')

    plt.title('Baseball Height Over Time')

    plt.xlabel('Time (s)')

    plt.ylabel('Height (m)')

    plt.legend()

    plt.grid()

    plt.show()

*def* central\_diff(*x*, *y*):

    '''

    (np.array, np.array) -> np.array, np.array

    Uses the central difference method to approximate the derivative of f at x.

    Preconditions: x and y are np.arrays of the same length.

    '''

    # Compute the differences in x and y using the same indices seen in class

    df = y[1:] - y[:-1]

    dx = x[1:] - x[:-1]

    # Return the x values and the approximations of the derivative of f at x

    # The x values are shifted by 0.5 \* dx to align with the central difference method

    return x[:-1] + (0.5 \* dx), df/dx

*def* first\_derivative\_plot():

    '''

    () -> None

    Create a scatterplot of the first central difference for the data in 'baseball\_height.csv'.

    Preconditions: None

    '''

    # Load the data from the file

    x\_data, y\_data = load\_data()

    # Compute the first central difference of the data

    dx, dy = central\_diff(x\_data, y\_data)

    # Initialize the second figure

    plt.figure(2)

    # Create a scatter plot of the data with relevant labels and titles

    plt.scatter(dx, dy, *c*='k', *marker*='o', *label*='Velocity of Baseball')

    plt.title('Velocity of Baseball Over Time Using Central Difference Method')

    plt.xlabel('Time (s)')

    plt.ylabel('Velocity (m/s)')

    plt.legend()

    plt.grid()

    plt.show()

*def* second\_derivative\_plot():

    '''

    () -> None

    Creates a scatterplot of the second central difference for the data in 'baseball\_height.csv'.

    Preconditions: None

    '''

    # Load the data from the file

    x\_data, y\_data = load\_data()

    # Compute the first central difference of the data

    dx, dy = central\_diff(x\_data, y\_data)

    # Compute the second central difference from the first difference

    ddx, ddy = central\_diff(dx, dy)

    # Round all values to account for floating point errors making the plot unbelievably wrong

    dx = np.around(dx, 1)

    dy = np.around(dy, 3)

    ddx = np.around(ddx, 1)

    ddy = np.around(ddy, 3)

    # Initialize the third figure

    plt.figure(3)

    # Create a scatter plot of the data with relevant labels and titles

    plt.scatter(ddx, ddy, *c*='k', *marker*='o', *label*='Acceleration of Baseball (m/s\u00B2)')

    plt.title('Acceleration of Baseball Over Time Using Central Difference Method')

    plt.xlabel('Time (s)')

    plt.ylabel('Acceleration (m/s\u00B2)')

    plt.legend()

    plt.grid()

    plt.show()

# Part 2

*def* trap\_integration(*x*, *y*):

    '''

    (np.array, np.array) -> np.array

    Approximate the definite integral of y with respect to x using the trapezoidal rule.

    Preconditions: x and y are arrays of the same length.

    '''

    # Compute the width of each subinterval

    delta\_x = np.diff(x)

    # Compute the cumulative sum of y, multiplied by the width of the subintervals

    integral = np.cumsum(0.5 \* (y[:-1] + y[1:]) \* delta\_x)

    # Prepend a zero to the start of the integral array, because the integral at the first point is always zero

    integral = np.concatenate(([0], integral))

    return integral

*def* first\_integral\_plot():

    '''

    () -> None

    Create a scatterplot of the first integral of the data in 'baseball\_height.csv'.

    Preconditions: None

    '''

    # Load the data from the file

    x\_data, y\_data = load\_data()

    # Compute the first central difference of the data

    dx, dy = central\_diff(x\_data, y\_data)

    # Compute the second central difference from the first difference

    ddx, ddy = central\_diff(dx, dy)

    # Round all values to account for floating point errors making the plot unbelievably wrong

    dx = np.around(dx, 1)

    dy = np.around(dy, 3)

    ddx = np.around(ddx, 1)

    ddy = np.around(ddy, 3)

    # Compute the first integral of the data from the second central difference

    integral = trap\_integration(dx[:-1], ddy)

    # Initialize the fourth figure

    plt.figure(4)

    # Create a scatter plot of the data with relevant labels and titles

    plt.scatter(dx[:-1], integral, *c*='k', *marker*='o', *label*='Velocity of Baseball (m/s)')

    plt.title('Velocity of Baseball Over Time Using Trapezoidal Rule')

    plt.xlabel('Time (s)')

    plt.ylabel('Velocity (m/s)')

    plt.legend()

    plt.grid()

    plt.show()

*def* second\_integral\_plot():

    '''

    () -> None

    Create a scatterplot of the second integral of the data in 'baseball\_height.csv'.

    Preconditions: None

    '''

    # Load the data from the file

    x\_data, y\_data = load\_data()

    # Compute the first central difference of the data

    dx, dy = central\_diff(x\_data, y\_data)

    # Compute the second central difference from the first difference

    ddx, ddy = central\_diff(dx, dy)

    # Round all values to account for floating point errors making the plot unbelievably wrong

    dx = np.around(dx, 1)

    dy = np.around(dy, 3)

    ddx = np.around(ddx, 1)

    ddy = np.around(ddy, 3)

    # Compute the first integral of the data from the second central difference

    i = np.around(trap\_integration(dx[:-1], ddy), 3)

    # Compute the second integral of the data from the first integral

    ii = np.around(trap\_integration(x\_data[:-2], i), 3)

    # Initialize the fifth figure

    plt.figure(5)

    # Create a scatter plot of the data with relevant labels and titles

    plt.scatter(x\_data[:-2], ii, *c*='k', *marker*='o', *label*='Baseball Height (m)')

    plt.title('Baseball Height Over Time Using Trapezoidal Rule')

    plt.xlabel('Time (s)')

    plt.ylabel('Height (m)')

    plt.legend()

    plt.grid()

    plt.show()

*def* second\_integral\_plot\_adjusted():

    '''

    () -> None

    Create a scatterplot of the second integral of the data in 'baseball\_height.csv'.

    Preconditions: None

    '''

    # Load the data from the file

    x\_data, y\_data = load\_data()

    # Compute the first central difference of the data

    dx, dy = central\_diff(x\_data, y\_data)

    # Compute the second central difference from the first difference

    ddx, ddy = central\_diff(dx, dy)

    # Round all values to account for floating point errors making the plot unbelievably wrong

    dx = np.around(dx, 1)

    dy = np.around(dy, 3)

    ddx = np.around(ddx, 1)

    ddy = np.around(ddy, 3)

    # Compute the first integral of the data from the second central difference

    i = np.around(trap\_integration(dx[:-1], ddy), 3) + 4.853    # Constant added to make the graph look as expected; found by dy[0] - i[0]

    # Compute the second integral of the data from the first integral

    ii = np.around(trap\_integration(x\_data[:-2], i), 3)

    # Initialize the sixth figure

    plt.figure(6)

    # Create a scatter plot of the data with relevant labels and titles

    plt.scatter(x\_data[:-2], ii, *c*='k', *marker*='o', *label*='Baseball Height (m)')

    plt.title('Baseball Height Over Time with Adjusted Initial Velocity Using Trapezoidal Rule')

    plt.xlabel('Time (s)')

    plt.ylabel('Height (m)')

    plt.legend()

    plt.grid()

    plt.show()

*def* main():

    '''

    () -> None

    Executes relevant functions for Assignment 9.

    Preconditions: None

    '''

    # Load the data from the file

    x, y = load\_data()

    # Print the data

    print(*f*'Baseball Height Over Time:\n x: {x}\n y: {y}')

    # Show the scatter plot of the data

    height\_plot()

    # Compute the first central difference of the data

    dx, dy = central\_diff(x, y)

    # Print the first central difference

    print(*f*'First Central Difference of Baseball Height Over Time:\n dx: {dx}\n dy: {dy}')

    # Show the scatter plot of the first central difference

    first\_derivative\_plot()

    # Compute the second central difference from the first difference

    ddx, ddy = central\_diff(dx, dy)

    # Print the second central difference

    print(*f*'Second Central Difference of Baseball Height Over Time:\n ddx: {ddx}\n ddy: {ddy}')

    # Show the scatter plot of the second central difference

    second\_derivative\_plot()

    # Compute the first integral of the data from the second central difference

    i = trap\_integration(dx[:-1], ddy)

    # Print the first integral

    print(*f*'First Integral of Baseball Height Over Time:\n dx: {dx[:-1]}\n dy: {i}')

    # Show the scatter plot of the first integral

    first\_integral\_plot()

    # Compute the second integral of the data from the first integral

    ii = trap\_integration(x[:-2], i)

    # Adjust the initial velocity to make the graph look as expected

    ii\_adjusted = trap\_integration(x[:-2], i + 4.853)

    # Print the second integral

    print(*f*'Second Integral of Baseball Height Over Time:\n x: {x[:-2]}\n y: {ii}')

    # Print the second integral with adjusted initial velocity

    print(*f*'Second Integral of Baseball Height Over Time w/ Adjusted Initial Velocity:\n x: {x[:-2]}\n y: {ii\_adjusted}')

    # Show the scatter plot of the second integral

    second\_integral\_plot()

    # Show the scatter plot of the second integral with adjusted initial velocity

    second\_integral\_plot\_adjusted()

1. The up direction is chosen to be positive [↑](#footnote-ref-2)
2. Much of the story of this lab was inspired by the novel Project Hail Mary by Andy Weir. It is a hard science fiction story where the main character carries out quite a few experiments that a physics-minded reader may find fun to read.

   **Thanks for the book recommendation :D** [↑](#footnote-ref-3)