Week 7 Activites

Graded

Student

Harper Chen

Total Points

99 / 100 pts

Question 1

Sketch Accumulation Function

10 / 10 pts

✔ **– 0 pts** Correct

Question 2

Using the Graph - 5 parts

10 / 10 pts

✔ **– 0 pts** Correct

– 0 pts Close enough - 215

Question 3

Sketch again with 25

10 / 10 pts

✔ **– 0 pts** Correct

– 5 pts 2nd sketch?

Question 4

blank

0 / 0 pts

✔ **– 0 pts** Correct

Question 5

Sketch a smooth curve

9 / 10 pts

– 0 pts Correct

– 5 pts Did you preserve the total number of deaths? I think a lot of folks just lost their lives. Imagine the area above your line and how much it would fill in below...

– 5 pts This should be turning the stacks into a curve - so it should go down again.

✔ **– 1 pt** Okay, but how smooth is this really?

Question 6

Using the curve - 4 parts

10 / 10 pts

✔ **– 0 pts** Correct

Question 7

Consider the smooth functions - 3 parts

10 / 10 pts

✔ **– 0 pts** Correct

Question 8

Matching

10 / 10 pts

✔ **– 0 pts** Correct

Question 9

Coding and demo TABLE

15 / 15 pts

✔ **– 0 pts** Correct

Question 10

Coding and Demo PLOT

15 / 15 pts

✔ **– 0 pts** Correct

– 0 pts Use smaller steps

Question assigned to the following page: 1

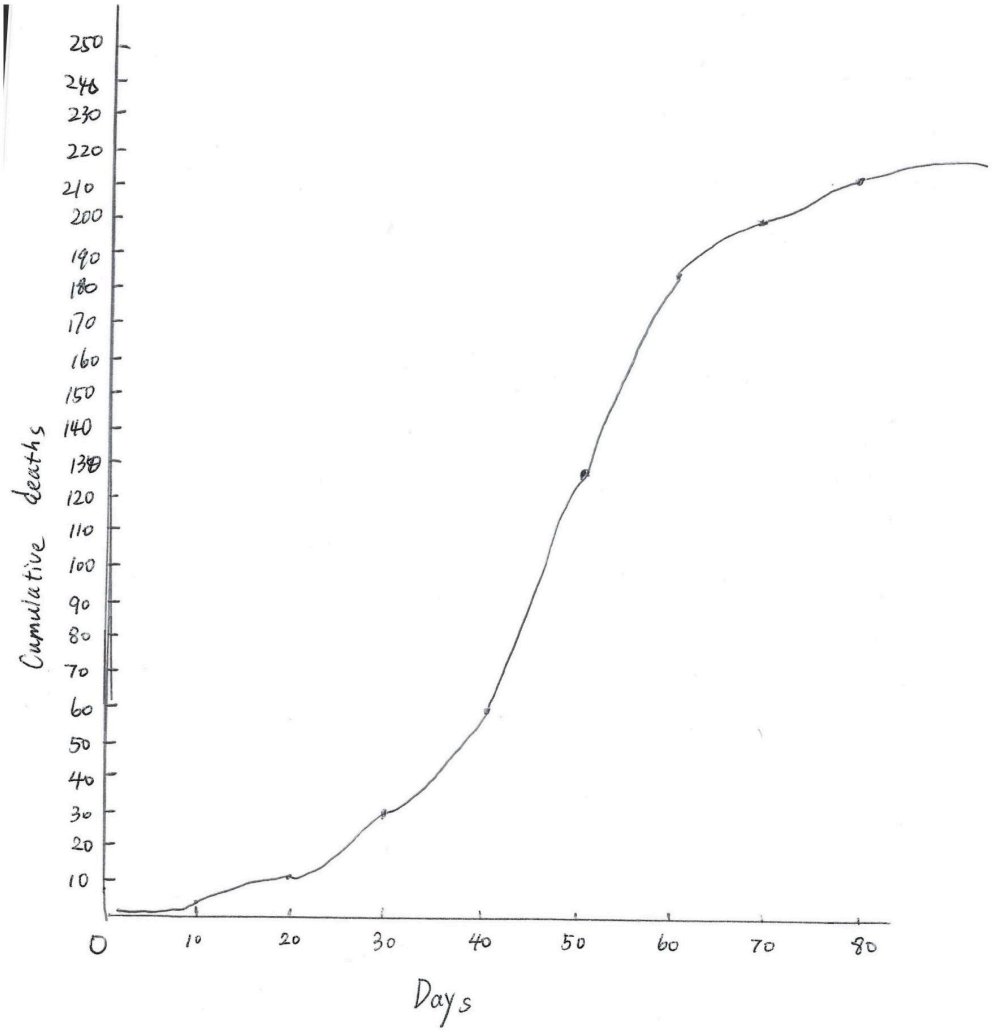
week 7

October 17, 2024

0.1 Activity 1

[1]:
```
from IPython.display import Image
Image('1.jpg')
```

[1]:



Questions assigned to the following page: 2 and 3

0.2 2

- Based on the graph, the total number of deaths is 215 by day 85.
- The peak of the illness occurred around day 41 to day 45, where the number of new daily deaths was the highest.
- I decided this based on the steepest part of the accumulation curve. In this time range, the slope of the curve is the steepest, indicating the highest rate of deaths per day.
- On the original histogram (daily deaths graph), the peak is represented by the tallest bar, which shows the highest number of deaths per day. On the accumulation graph I created, the peak is represented by the steepest slope, which indicates the rapid accumulation of deaths during that time.
- The derivative of the accumulation function represents the rate of change of cumulative deaths, which is the number of new deaths per day. In other words, the derivative of the accumulation function is the histogram of daily deaths.

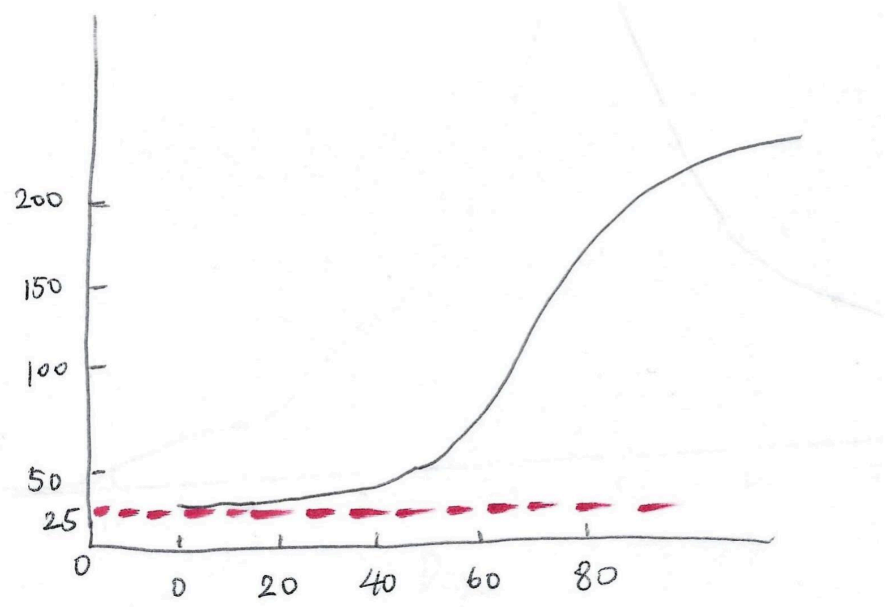
0.3 3

- The total number of deaths, including the 25 pre-recording deaths, would be the total from the accumulation graph (215) plus the 25 deaths, making the total 240 deaths by time 89.
- At time 0, there were 25 deaths already, which serves as the starting point for the accumulation.
- To find the actual total at time 89, simply add the 25 pre-recording deaths to the cumulative total from the graph at day 89. Thus, the actual total number of deaths at day 89 is 215 + 25 = 240 deaths.

[6]:
```
from IPython.display import Image
Image('3.jpg')
```

[6]:

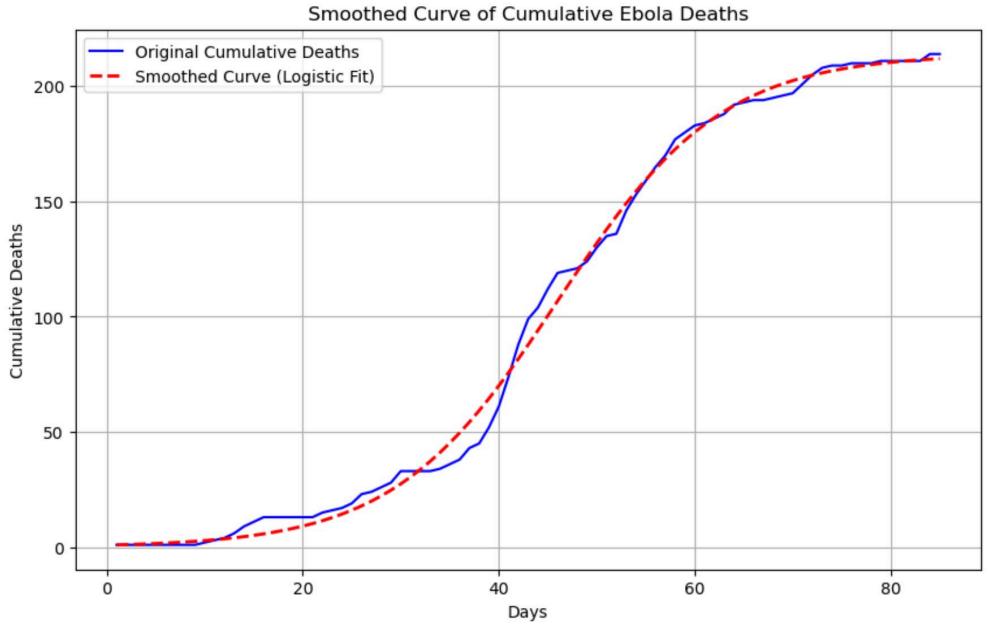
Question assigned to the following page: 3



[2]:
```
## 4
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit

deaths_per_day = [
    1, 0, 0, 0, 0, 0, 0, 0, 0, 1,
    1, 1, 2, 3, 2, 2, 0, 0, 0, 0,
    0, 2, 1, 1, 2, 4, 1, 2, 2, 5,
    0, 0, 0, 1, 2, 2, 5, 2, 7, 9,
    13, 14, 11, 5, 8, 7, 1, 1, 3, 6,
    5, 1, 10, 7, 6, 6, 5, 7, 3, 3,
    1, 2, 2, 4, 1, 1, 0, 1, 1, 1,
    4, 4, 3, 1, 0, 1, 0, 0, 1, 0,
    0, 0, 0, 3, 0
]

cumulative_deaths = np.cumsum(deaths_per_day)
correct_days = np.arange(1, len(deaths_per_day) + 1)

def logistic_function(x, L, k, x_0):
    return L / (1 + np.exp(-k * (x - x_0)))
```

No questions assigned to the following page.

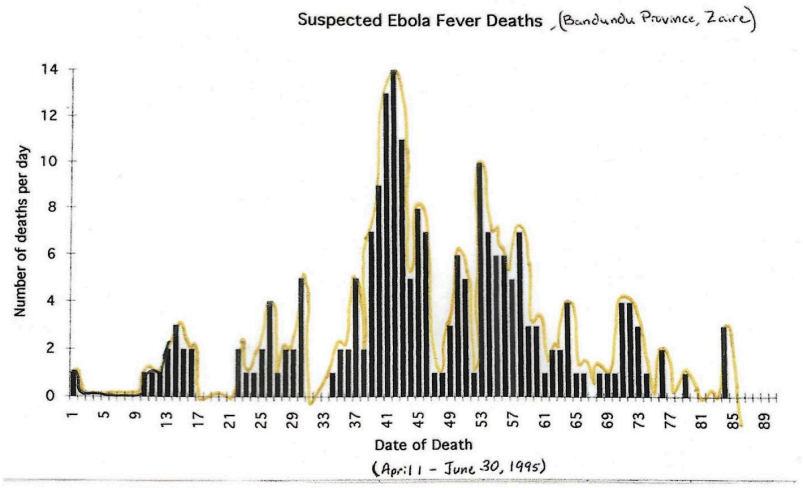
```
initial_guess = [240, 0.1, 45]  # L (max deaths), k (growth rate), x_0␣
↪(midpoint)

popt, _ = curve_fit(logistic_function, correct_days, cumulative_deaths,␣
↪p0=initial_guess)

L_opt, k_opt, x_0_opt = popt

smoothed_cumulative = logistic_function(correct_days, L_opt, k_opt, x_0_opt)

fig, ax = plt.subplots(figsize=(10, 6))
ax.plot(correct_days, cumulative_deaths, 'b-', label="Original Cumulative␣
↪Deaths")
ax.plot(correct_days, smoothed_cumulative, 'r--', label="Smoothed Curve␣
↪(Logistic Fit)", linewidth=2)
ax.set_xlabel("Days")
ax.set_ylabel("Cumulative Deaths")
plt.title("Smoothed Curve of Cumulative Ebola Deaths")
plt.grid(True)
plt.legend()
plt.show()
```



[3]:
```
# Display the image
Image('5.jpg')
```

Questions assigned to the following page: 5, 6, and 7

[3]:



0.4 6

- The peak of the deaths occurs around **day 43**. This is where the curve reaches its highest point, indicating the day with the maximum number of deaths (14 deaths).
- The accumulation function of the new smooth graph will start at zero, then rise steadily, following the curve's shape. It will be steepest during the period around the peak (days 40 to 50) when the deaths were increasing most rapidly. After that, the slope will gradually flatten as the number of deaths per day decreases, leading to a cumulative total close to the original graph's total deaths. The overall shape would be similar to an S-curve, with rapid growth followed by a leveling off.
- The average number of deaths per day can be calculated by dividing the total number of deaths by the total number of days. Based on the original data, the total number of deaths is **215** over **85 days**. Therefore, the average number of deaths per day is:

$$\text{Average deaths per day} = \frac{215}{85} \approx 2.53 \,\text{deaths/day}.$$

- I calculated the average by dividing the total number of deaths (215) by the total number of days (85). This gives an approximation of the daily death rate throughout the outbreak.

0.5 7

Function A: - Good: The curve captures the right side most dots and aligning with the central dots around day 55. - Bad: The tails (before day 20 and after day 70) are flat, and the curve misses many data points in the early and mid stages. Overall, it underestimates deaths outside the peak region.

Questions assigned to the following page: 7, 8, and 9

Function B: - Good: The curve fits the peak perfectly and aligns with many dots from day 30 to day 60. The curve also smoothly follows the data in the tail regions, capturing early and late deaths more accurately than Function A. - Bad: The fit is very close to the dots in the middle and the tails. But still have some dots did not cover well.

Function C: - Good: This function captures more dots in the central area (days 30 to 50) and represents a broader peak, which helps capture the middle data points. - Bad: The tail on the right overestimates deaths after the peak, especially around day 60 and beyond, leading to a longer tail and peak that doesn't match the data points.

Function D: - Good: This function is centered around day 44 and fits the peak similarly to Function A. It captures some of the central data points around the peak. - Bad: The narrowness of the curve leads to underestimating deaths in the tail regions, especially after day 50. The early and late data points don't fit well.

Function E: - Good: The curve fits the most reasonably with the dots between days 30 and 60. - Bad: The curve declines too quickly on both sides of the peak, leading to a poor fit with the early and late data points. The curve doesn't capture many dots outside the central peak.

Function F: - Good: The curve fits the dots similarly to Function D. - Bad: Same like Function D.

Do they each represent the same number of total deaths? No, each function represents a slightly different total number of deaths. The shape of each curve affects the area under the curve, which corresponds to the total number of deaths. Functions with wider peaks or extended tails will result in higher total death estimates, while narrower curves will represent fewer deaths. The total deaths can be considered by integrating the area under each curve.

Which is the best fit to the data? The best fit to the data appears to be Function B. It captures a good balance between the peak and the gradual decline in deaths over time. It aligns well with the data points before and after the peak, covering a wider range of the outbreak and not falling off too abruptly like some of the narrower functions. This makes it a more realistic representation of the overall death trend.

0.6 8

- Function A → Accumulation 1
- Function B → Accumulation 2
- Function C → Accumulation 6
- Function D → Accumulation 5
- Function E → Accumulation 3
- Function F → Accumulation 4

0.7 Activity 2

[4]:
```
import math

def derivative(t): # calculate the derivative dy/dt = cos(t^2)
    return math.cos(t ** 2)

def euler_method_table(start_t, end_t, num_steps): # Function to approximate␣
 ↪the solution using Euler's Method
```

Question assigned to the following page: 9

```
    delta_t = (end_t - start_t) / num_steps  # Calculate the step size (Δt)
    t = start_t  # Initialize the starting values
    accumulated_y = 0

    print(f"{'Time t':<10}{'Δy (change)':<20}{'Accumulated y':<20}")
    print("=" * 50)

    for step in range(num_steps):
        delta_y = derivative(t) * delta_t # Calculate the change in y (Δy)␣
↪using the derivative function
        accumulated_y += delta_y     # Update the accumulated y by adding Δy
        print(f"{t:<10.2f}{delta_y:<20.6f}{accumulated_y:<20.6f}")  # Print the␣
↪current values in a formatted way
        t += delta_t    # Update t to the next time step

euler_method_table(0, 4, 8)  #initial time = 0, final time = 4, and 8 steps
```

```
Time t    Δy (change)         Accumulated y
==================================================
0.00      0.500000            0.500000
0.50      0.484456            0.984456
1.00      0.270151            1.254607
1.50      -0.314087           0.940521
2.00      -0.326822           0.613699
2.50      0.499725            1.113423
3.00      -0.455565           0.657858
3.50      0.475185            1.133044
```

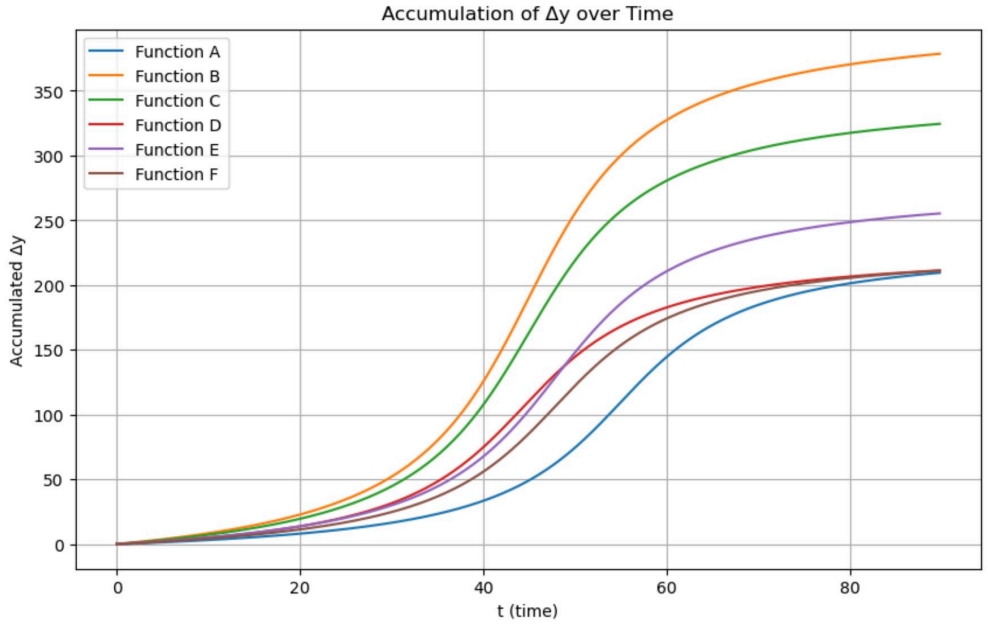
0.8 Activity 3

[5]:
```
import math
import matplotlib.pyplot as plt

def A(x):
    return 827 / ((10.5)**2 + (x - 55)**2)

def B(x):
    return 1400 / ((10)**2 + (x - 45)**2)

def C(x):
    return 1200 / ((10)**2 + (x - 45)**2)

def D(x):
    return 827 / ((10.5)**2 + (x - 44.4)**2)

def E(x):
```

Question assigned to the following page: 10

```
    return 1000 / ((10.5)**2 + (x - 48)**2)

def F(x):
    return 827 / ((10.5)**2 + (x - 48)**2)

def plot_accumulation(t_initial, t_final, num_steps, func, label):
    delta_t = (t_final - t_initial) / num_steps   # Calculate the time step
 ↪(Δt)
    t = t_initial  # Initial time
    accumulation = 0  # Accumulation starts at 0
    times = []  # Store time steps for plotting
    accumulations = []  # Store accumulated Δy for plotting

    for k in range(num_steps): # Perform Euler's method to approximate
 ↪accumulated Δy
        delta_y = func(t) * delta_t  # Calculate Δy = func(t) * Δt
        accumulation += delta_y # Add Δy to the accumulation
        times.append(t) # Store values for plotting
        accumulations.append(accumulation)
        t += delta_t  # Increment t by Δt

    plt.plot(times, accumulations, label=label)
    plt.xlabel("t (time)")
    plt.ylabel("Accumulated Δy")
    plt.title("Accumulation of Δy over Time")
    plt.grid(True)

# Example usage of PLOT program:
t_initial = 0
t_final = 4
num_steps = 400  # Steps same to the example in Program: PLOT

plt.figure(figsize=(10, 6))
plot_accumulation(0, 90, 500, A, 'Function A')
plot_accumulation(0, 90, 500, B, 'Function B')
plot_accumulation(0, 90, 500, C, 'Function C')
plot_accumulation(0, 90, 500, D, 'Function D')
plot_accumulation(0, 90, 500, E, 'Function E')
plot_accumulation(0, 90, 500, F, 'Function F')
plt.legend()
plt.show()
```

Question assigned to the following page: 10



This code uses **Euler's method** to approximate and graph the accumulation of y over time for different functions. It calculates the change in y at each small time step, adds these changes together, and shows how the total y grows over time.The six Ebola-related functions, each modeling the accumulation of deaths over time. The output is a graph comparing how these functions behave, allowing us to visualize and compare the growth or accumulation for each function.

[ ]: