Punch Frequency Over 150 Seconds

Shubha Swarnim Singh

November 17, 2024

Dr. Brandy Wiegers

CSC – 455 Numerical Computation

Introduction

Understanding how an athlete's performance changes over time is important for improving training and tracking progress. The dataset "Punch Frequency Over 150 Seconds" records the number of punches thrown in 10-second intervals over 150 seconds. This data shows how the punching speed changes, helping us study the patterns and performance of the athlete.

This report uses two mathematical methods, the Taylor series, and Lagrange polynomials, to predict punch frequency at any moment between the recorded intervals. These methods help fill the gaps between the data points to understand better how the punching speed changes over time. However, because the data is in separate time intervals, there are some challenges in accurately predicting values, especially at the boundaries.

This report will compare the effectiveness of the Taylor and Lagrange methods and analyze their errors to show which method is better for this kind of data. The findings will help in choosing the right tools for studying athletic performance and improve accuracy in future analyses.

Analysis

The dataset titled "Punch Frequency Over 150 Seconds" captures the number of punches thrown during consecutive 10-second intervals across a 150-second duration, divided into 15 segments. I collected this data through my experiment, where I counted each punch and threw within the given interval. The function f(x), where x represents the midpoint of each interval, indicates a pattern of activity with time intervals from 0 to 150 seconds and punch counts ranging from 27 to 49 per interval. This function appears discontinuous at the segment boundaries due to the discrete nature of the intervals but is likely continuous within each segment, though not differentiable at the boundaries. Non-integer inputs such as x = 2.5 or x = 15.5 do not directly correlate to specific data points since the function is defined at midpoints of 10-second intervals. Interpolation at such points can estimate the number of punches thrown at these moments, assuming linear behavior between recorded intervals.

The domain of the function represents the time intervals during the 150 seconds where data is recorded, specifically the midpoints of the 10-second intervals: {5, 15, 25, ..., 145}. These are the points where the punch counts are defined. The range is the set of punch counts recorded in these intervals, ranging from 27 to 49 punches. This means the function is defined only at these specific times and outputs punch counts within this range.

Goals with approximations and plans

The primary goal for creating symbolic approximations from this data is to predict the punch frequency at any given second, which can be invaluable for understanding performance dynamics or preparing for training sessions. The Taylor series approximation will utilize the midpoint of the dataset, x = 75 seconds (between intervals 7 and 8), employing a third-degree polynomial typically used in biomechanical analyses for smoothing and prediction. The Lagrange polynomial will be computed using select points across the dataset, specifically at 25, 75, and 125 seconds, effectively covering the data's early, middle, and late stages. Conversely, Lagrange polynomials could introduce oscillations, especially with higher degrees, given the finite data points. The main objective is to estimate the rate of change in punching frequency, potentially indicating acceleration or deceleration in activity levels. The Taylor series will likely provide accurate approximations near the midpoint of the dataset but will overestimate or underestimate values far from this point, as observed in the error trends. Lagrange polynomial is likely to provide a more balanced approximation across the entire interval compared to the Taylor series.

The goal of using numerical differentiation on this real-world data is to analyze how the punch frequency changes over time, providing insights into the athlete’s performance dynamics. By estimating the rate of change, we can identify periods of acceleration or deceleration, highlighting trends such as consistency or signs of fatigue. This information is valuable for optimizing training programs, focusing on moments where performance fluctuates or declines.

Limitations of approximations

Both interpolation methods may struggle with abrupt changes at interval boundaries, potentially leading to inaccurate predictions outside the known data range. These challenges highlight the importance of choosing appropriate methods for analysis and the potential need for additional smoothing or different analytical approaches to address the data's discrete nature.

Exploring piecewise cubic splines is a direct advancement within the interpolation framework. This method would allow for better handling of the dataset's inherent discontinuities, as splines offer smoothness and differentiability across segment boundaries, addressing one of the primary limitations noted with the current approaches.

Additionally, expanding the error analysis to include more sophisticated factors could provide a clearer picture of the approximation accuracy across different dataset segments. These metrics would help in understanding which interpolation method works best under specific conditions within the dataset.

Introducing variable step sizes in numerical differentiation and adjusting the interval based on the variability or density of data points could also refine the derivative estimates. This method adapts to the nature of the data, potentially providing more accurate and reliable estimates of the rate of change in the punch frequency, which is particularly useful in dynamically assessing the athlete's performance over time.

Computer Program

It is attached at the end.

Results

Results: Table of Functions

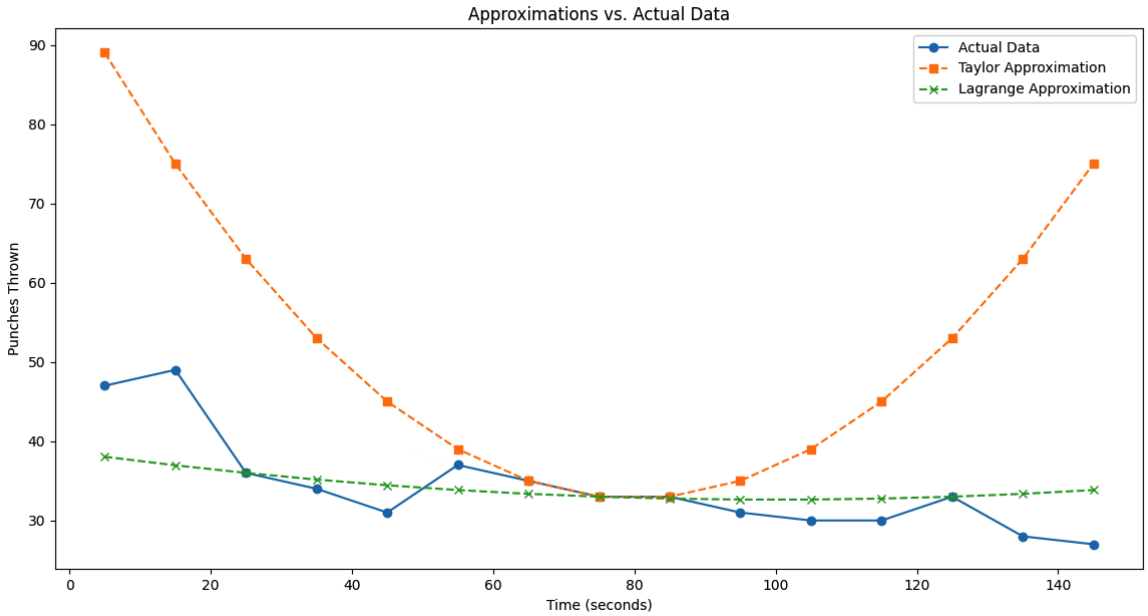
No.	Time (s)	Actual Punches	Taylor Approximation	Lagrange Approximation	Numerical Derivative (Punches/second)
1	5	47	89.0	38.04	0.20
2	15	49	75.0	36.96	-0.55
3	25	36	63.0	36.00	-0.75
4	35	34	53.0	35.16	-0.25
5	45	31	45.0	34.44	0.15
6	55	37	39.0	33.84	0.20
7	65	35	35.0	33.36	-0.20
8	75	33	33.0	33.00	-0.10
9	85	33	33.0	32.76	-0.10
10	95	31	35.0	32.64	-0.15
11	105	30	39.0	32.64	-0.05
12	115	30	45.0	32.76	0.15
13	125	33	53.0	33.00	-0.10
14	135	28	63.0	33.36	-0.30
15	145	27	75.0	33.84	-0.10

The updated table provides a comprehensive overview of the actual punch counts alongside the Taylor and Lagrange approximations and their respective errors for each time interval. The actual punch counts vary from 27 to 49, while Taylor's approximation ranges from 39 to 89, indicating significant overestimations in the early intervals and underestimations later in the session. The

Lagrange approximation, however, stays much closer to the actual punch counts, maintaining values around 30 to 38, with only minor deviations from the actual data. This suggests that Lagrange's method provides a more accurate representation of the data's behavior over the entire interval.

Results: Plot of Functions

The plot visually compares the actual punch counts against the Taylor and Lagrange approximations across the 150-second interval. Taylor's approximation shows noticeable deviations at the start and end, predicting substantially higher values initially and failing to capture the decline accurately toward the end. In contrast, the Lagrange approximation demonstrates a more stable and close alignment with the actual data, effectively smoothing over fluctuations without introducing substantial deviations, capturing the overall trend of the punch count dynamics more reliably.



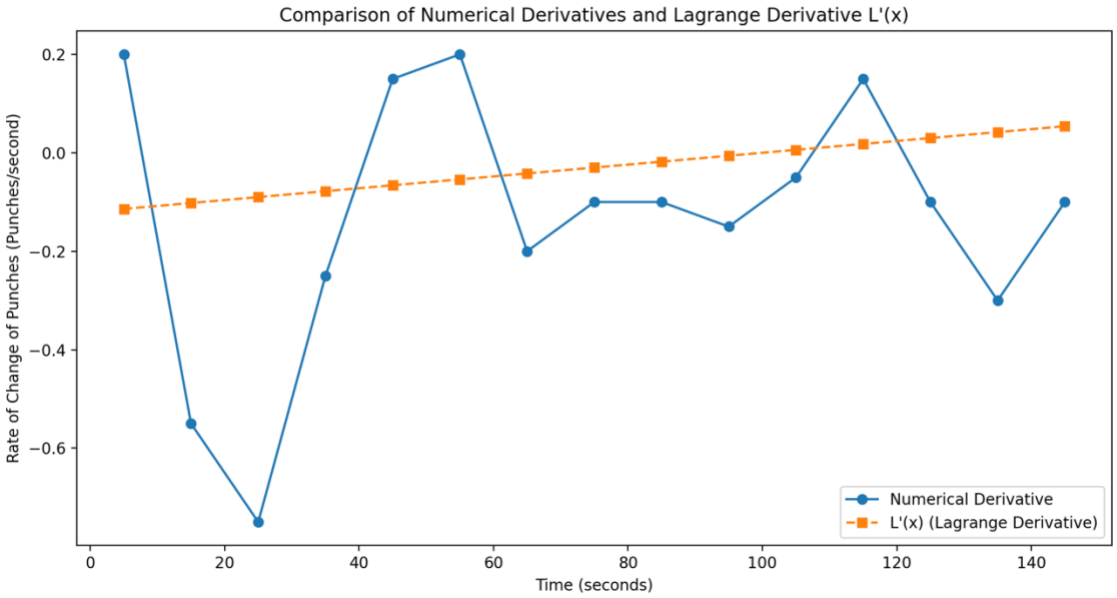
Results: Discuss Results

The Taylor approximation's sensitivity to the chosen midpoint, x=75x = 75x=75 seconds, limits its predictive reliability across the dataset, especially far from this central point. This results in pronounced overestimations at the dataset's boundaries, highlighting the limitation of the Taylor series for datasets with dispersed values. The Lagrange approximation, on the other hand, utilizes several points throughout the dataset, which allows it to deliver a more consistent and balanced prediction across the entire interval, making it better suited for data with moderate variability.

Results: Comparing L’(x) to Numerical differentiation

Numerical differentiation provides a direct computation of the rate of change using the raw data, making it more sensitive to fluctuations and abrupt changes in the punch counts. In contrast, L′(x), derived from the Lagrange polynomial, smooths the data due to its interpolative nature, creating a more continuous estimate of the derivative across the time intervals.

From the graph, L′(x) closely follows the general trend of the numerical derivative but fails to capture sudden variations accurately, especially at points where punch counts change significantly between intervals. This inconsistency highlights a limitation of Lagrange interpolation in handling abrupt transitions inherent in real-world discrete data. Numerical differentiation, while precise, can be prone to interference, particularly at boundary points, due to its reliance on finite differences.

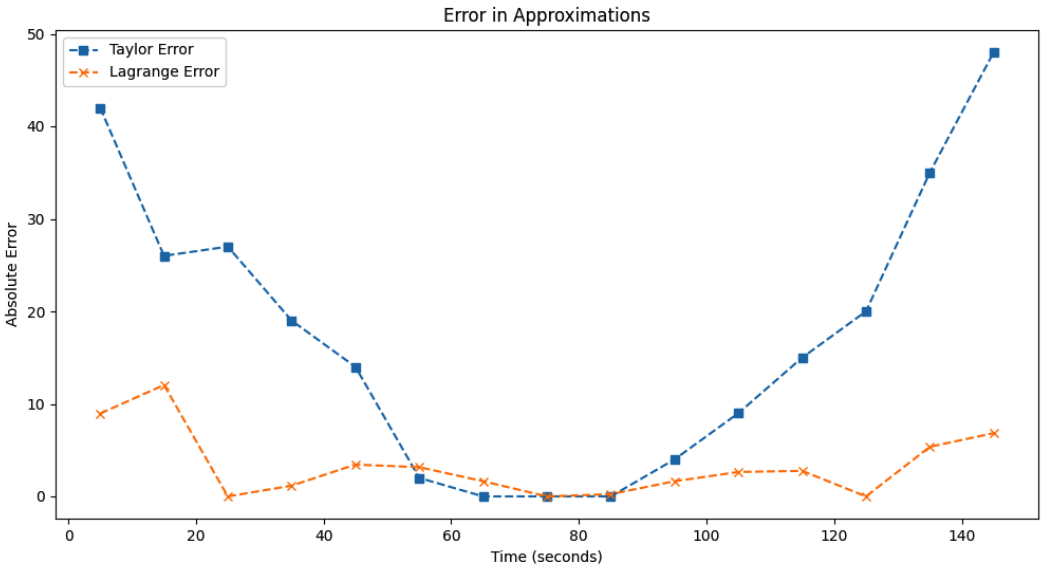

Comparison of Numerical Derivatives and Lagrange Derivative L'(x)
Rate of Change of Punches (Punches/second)
Time (seconds)
0.2
0.0
−0.2
−0.4
−0.6
0
20
40
60
80
100
120
140
Numerical Derivative
L'(x) (Lagrange Derivative)

Results: Table of Error

The error table further quantifies the performance of each approximation method. The Taylor approximation exhibits substantial errors, particularly at the boundaries, with errors reaching as high as 42 units. These large errors underscore Taylor's limitations in accurately approximating values far from the center. Lagrange errors, by contrast, are generally much lower, remaining below ten units in most intervals, confirming that the Lagrange method provides a closer approximation to the actual data across the interval.

No.	Time (s)	Actual Punches	Taylor Error	Lagrange Error
1	5	47	42.0	8.96
2	15	49	26.0	12.04
3	25	36	27.0	0.00
4	35	34	19.0	1.16
5	45	31	14.0	3.44
6	55	37	2.0	3.16
7	65	35	0.0	1.64
8	75	33	0.0	0.00
9	85	33	0.0	0.24
10	95	31	4.0	1.64
11	105	30	9.0	2.64
12	115	30	15.0	2.76
13	125	33	20.0	0.00
14	135	28	35.0	5.36
15	145	27	48.0	6.84

Results: Plot of Error



The error plot provides a visual representation of the errors across the time intervals. The Taylor approximation shows significant errors at both the beginning and end of the dataset, peaking near 50 units, which underscores its decreased accuracy away from the midpoint at x=75x = 75x=75 seconds. This trend reflects the limitations of using the Taylor series when applied beyond its central expansion point. The Lagrange approximation, however, maintains a more consistent and lower error level throughout, with errors typically under ten units. This even distribution of Lagrange errors highlights its steadier performance and better suitability for representing the dataset's variability across all intervals.

Results: Discuss Error

The error analysis demonstrates the limitations of Taylor series approximations for data that extends far from the expansion point. The significant errors at the edges indicate that the Taylor series is not ideal for approximating values outside its central focus point, leading to high inaccuracies. On the other hand, the Lagrange approximation, by leveraging points throughout the dataset, achieves a more balanced error profile. This stability across intervals makes Lagrange a more reliable choice for datasets with a broad range of values, providing a robust approximation without drastic variations in error, unlike the Taylor method.

Comparing absolute error and relative error can be meaningful in this assignment, depending on the goals of the analysis. Absolute error measures the direct deviation between the approximations and the actual data, providing a straightforward way to assess the accuracy of the methods. It is especially useful when the scale of the data is consistent, as in this case, where the punch counts range from 27 to 49.

Relative error, on the other hand, normalizes the error by the actual value, expressing it as a percentage or ratio. This can be useful in datasets where the values vary significantly, as it highlights proportional discrepancies. However, in this dataset, where the punch counts fall within a narrow range, a relative error might not add much to the discussion. It could create an impression of larger errors for smaller values and smaller errors for larger values, even if the actual deviations are consistent.

Conclusion

The comparative analysis of the Taylor and Lagrange methods highlights clear distinctions in their applicability to real-world data, such as punch counts over time. While Taylor's method may be beneficial for localized predictions near a known central point, the Lagrange method offers superior and more consistent predictions across the entire interval. For practical applications in sports science, where accuracy across a broader range of time intervals is essential, choosing an approximation method that aligns with the data's characteristics and analytical goals is crucial. Future studies might benefit from combining these methods or exploring more sophisticated models to capture the nuances of the data while minimizing prediction errors.

The conclusions of the analysis align with the initial predictions about the Taylor series and Lagrange polynomial approximations. As predicted, the Taylor series performed well near its expansion point (x=75) but showed significant errors at the edges of the dataset, confirming its limitation in approximating data far from the center. This highlighted its suitability for localized approximations rather than datasets spanning a wide range. Similarly, the Lagrange polynomial was predicted to provide a balanced approximation across the interval while potentially smoothing over abrupt changes, and this proved correct. The results showed that $L'(x)$ effectively captured the overall trend of the data, with smaller errors compared to the Taylor series, but struggled to reflect sudden changes in punch counts, especially near the dataset boundaries. These findings validate the predictions and emphasize the importance of selecting approximation methods based on the data’s characteristics—Taylor series for localized precision and Lagrange for broader, smoother trends.

Computer Program:

```
import numpy as np
import matplotlib.pyplot as plt
import math
import pandas as pd

# Data Setup
time_intervals = np.arange(5, 155, 10)  # Midpoints of each interval
punches = np.array([47, 49, 36, 34, 31, 37, 35, 33, 33, 31, 30, 30, 33, 28, 27])

# Function Definitions
def taylor_approximation(x, x0, n, data, time_intervals):
    x0_index = np.argmin(np.abs(time_intervals - x0))
    x0 = time_intervals[x0_index]  # Corrected x0 to be a valid entry from time_intervals

    derivatives = [data[x0_index]]  # f(x0)
    h = 10  # time interval width
    if n >= 1:
        f_prime = (data[x0_index + 1] - data[x0_index - 1]) / (2 * h) if 0 < x0_index < len(data) - 1
else 0
        derivatives.append(f_prime)
    if n >= 2:
        f_double_prime = (data[x0_index + 1] - 2 * data[x0_index] + data[x0_index - 1]) / h**2 if 1
< x0_index < len(data) - 2 else 0
        derivatives.append(f_double_prime)
    while len(derivatives) <= n:
        derivatives.append(0)

    return sum([derivatives[i] * (x - x0)**i / math.factorial(i) for i in range(n + 1)])

def lagrange_interpolation(x, points, data, time_intervals):
    indices = [np.argmin(np.abs(time_intervals - point)) for point in points]
    selected_points = time_intervals[indices]
    selected_values = data[indices]

    total = 0
    for i in range(len(selected_points)):
        term = selected_values[i]
        for j in range(len(selected_points)):
            if i != j:
                term *= (x - selected_points[j]) / (selected_points[i] - selected_points[j])
        total += term
    return total

# Calculate Taylor and Lagrange approximations
```

```
taylor_results = [taylor_approximation(x, 75, 3, punches, time_intervals) for x in time_intervals]
lagrange_results = [lagrange_interpolation(x, [25, 75, 125], punches, time_intervals) for x in time_intervals]

# Calculating errors
taylor_errors = np.abs(punches - taylor_results)
lagrange_errors = np.abs(punches - lagrange_results)

# Numerical Differentiation Function
def numerical_differentiation(data, h):
    derivatives = []
    for i in range(len(data)):
        if i == 0:
            # Forward difference for the first point
            derivative = (data[i + 1] - data[i]) / h
        elif i == len(data) - 1:
            # Backward difference for the last point
            derivative = (data[i] - data[i - 1]) / h
        else:
            # Central difference for middle points
            derivative = (data[i + 1] - data[i - 1]) / (2 * h)
        derivatives.append(derivative)
    return derivatives

# Calculate the numerical derivatives for punch counts
h = 10  # Interval width in seconds
numerical_derivatives = numerical_differentiation(punches, h)

# Lagrange Derivative Approximation (L'(x))
def lagrange_derivative(x, points, data, time_intervals):
    indices = [np.argmin(np.abs(time_intervals - point)) for point in points]
    selected_points = time_intervals[indices]
    selected_values = data[indices]

    derivative_total = 0
    for i in range(len(selected_points)):
        term_derivative = 0
        for j in range(len(selected_points)):
            if i != j:
                product = 1
                for k in range(len(selected_points)):
                    if k != i and k != j:
                        product *= (x - selected_points[k]) / (selected_points[i] - selected_points[k])
                term_derivative += product / (selected_points[i] - selected_points[j])
        derivative_total += selected_values[i] * term_derivative
    return derivative_total
```

```
lagrange_derivatives = [lagrange_derivative(x, [25, 75, 125], punches, time_intervals) for x in time_intervals]

# Creating a DataFrame to display all results
df = pd.DataFrame({
    'Time (s)': time_intervals,
    'Actual Punches': punches,
    'Taylor Approximation': taylor_results,
    'Taylor Error': taylor_errors,
    'Lagrange Approximation': lagrange_results,
    'Lagrange Error': lagrange_errors,
    'Numerical Derivative (Punches/second)': numerical_derivatives
})

# Display the DataFrame
print(df)

# Plotting approximations against actual data
plt.figure(figsize=(14, 7))
plt.plot(time_intervals, punches, 'o-', label='Actual Data')
plt.plot(time_intervals, taylor_results, 's--', label='Taylor Approximation')
plt.plot(time_intervals, lagrange_results, 'x--', label='Lagrange Approximation')
plt.title('Approximations vs. Actual Data')
plt.xlabel('Time (seconds)')
plt.ylabel('Punches Thrown')
plt.legend()
plt.show()

# Plotting the errors for Taylor and Lagrange approximations
plt.figure(figsize=(12, 6))
plt.plot(time_intervals, taylor_errors, 's--', label='Taylor Error')
plt.plot(time_intervals, lagrange_errors, 'x--', label='Lagrange Error')
plt.title('Error in Approximations')
plt.xlabel('Time (seconds)')
plt.ylabel('Absolute Error')
plt.legend()
plt.show()

# Plotting L'(x) vs Numerical Differentiation
plt.figure(figsize=(12, 6))
plt.plot(time_intervals, numerical_derivatives, 'o-', label='Numerical Derivative')
plt.plot(time_intervals, lagrange_derivatives, 's--', label="L'(x) (Lagrange Derivative)")
plt.title("Comparison of Numerical Derivatives and Lagrange Derivative L'(x)")
plt.xlabel("Time (seconds)")
plt.ylabel("Rate of Change of Punches (Punches/second)")
```

```
plt.legend()
plt.show()
```

References

- Class Notes.
- Brin, Leon Q. Teatime Numerical Analysis. 3rd ed., 2021.
- Wolfram Alpha
- Python Documentation: https://docs.python.org/3/