Numerical Analysis Project

Predicting stationary stock price & volume
using Interpolation methods and the composite trapezoidal rule

Presented By:
Kornpasit Theerathitayangkul, Jirapit Sripitak, Arthakorn Raksenawanich

Presented To:
Asst. Prof. Pallop Huabsomboon

AGENDA

INTRODUCTION

In today's world, numerous investment options exist, including mutual funds, gold, bonds, cryptocurrencies, and stocks. However, these options come with varying levels of risk and return. To mitigate risk, this presentation will focus on predicting stock prices using historical data trends and two numerical methods: interpolation and the composite trapezoidal rule. Combining these methods can help increase prediction accuracy and minimize uncertainty.

METHODS AND OPERATIONS

Cubic Spline Interpolation

First, using the following equation to calculate the variable M

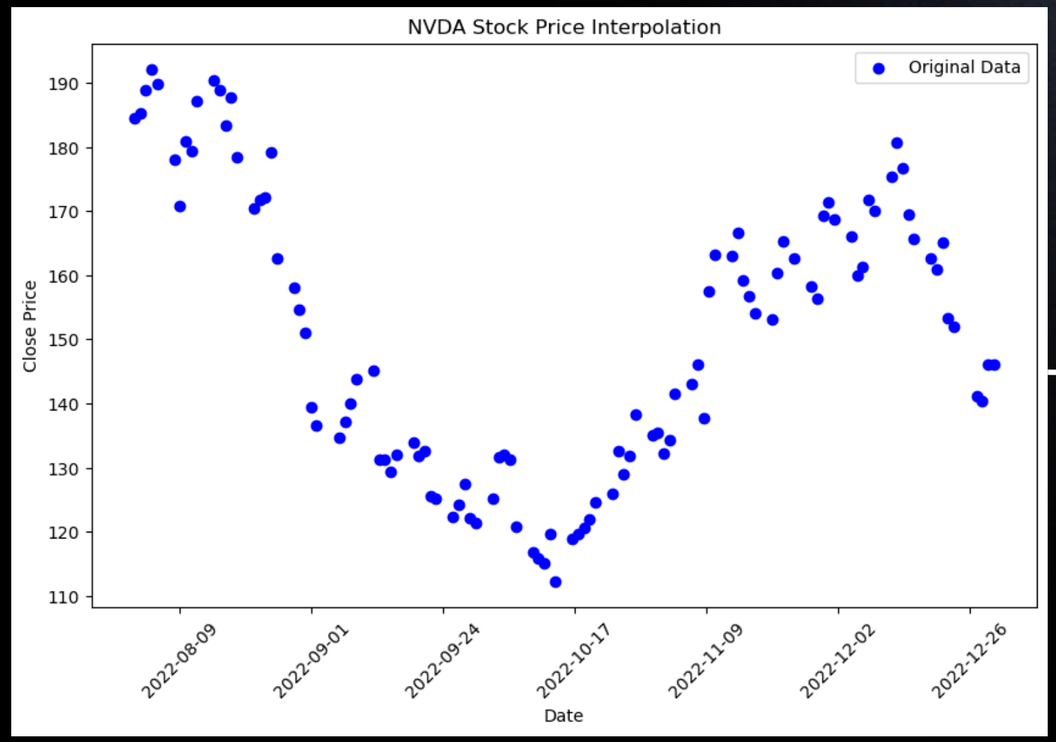
$$\frac{x_j - x_{j-1}}{6} M_{j-1} + \frac{x_{j+1} - x_{j-1}}{3} M_j + \frac{x_{j+1} - x_j}{6} M_{j+1} = \frac{y_{j+1} - y_j}{x_{j+1} - x_j} - \frac{y_j - y_{j-1}}{x_j - x_{j-1}},$$
$$j = 2, 3, \ldots, n-1$$

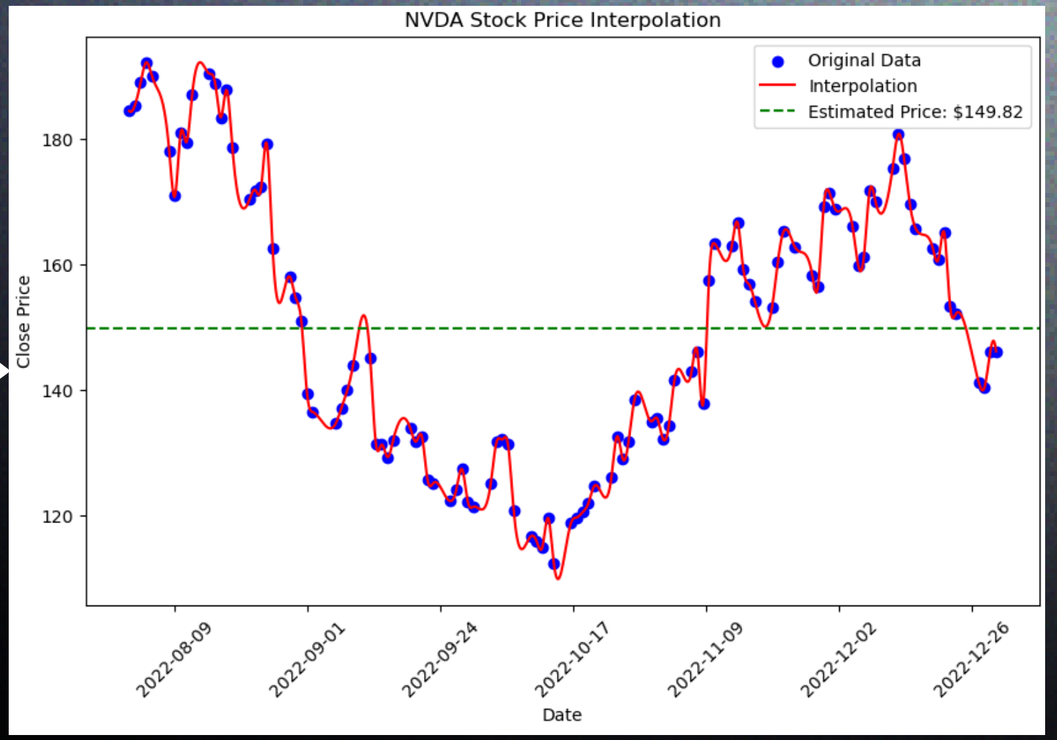
Then, we use the following equation to construct the cublic spline interpolation between each data points

$$S(x) = \frac{(x_j - x)^3 M_{j-1} + (x - x_{j-1})^3 M_j}{6(x_j - x_{j-1})} + \frac{(x_j - x) y_{j-1} + (x - x_{j-1}) y_j}{x_j - x_{j-1}}$$
$$-\frac{1}{6}(x_j - x_{j-1}) \left[(x_j - x) M_{j-1} + (x - x_{j-1}) M_j \right], \quad x_{j-1} \le x \le x_j.$$

METHODS AND OPERATIONS

Cubic Spline Interpolation





METHODS AND OPERATIONS

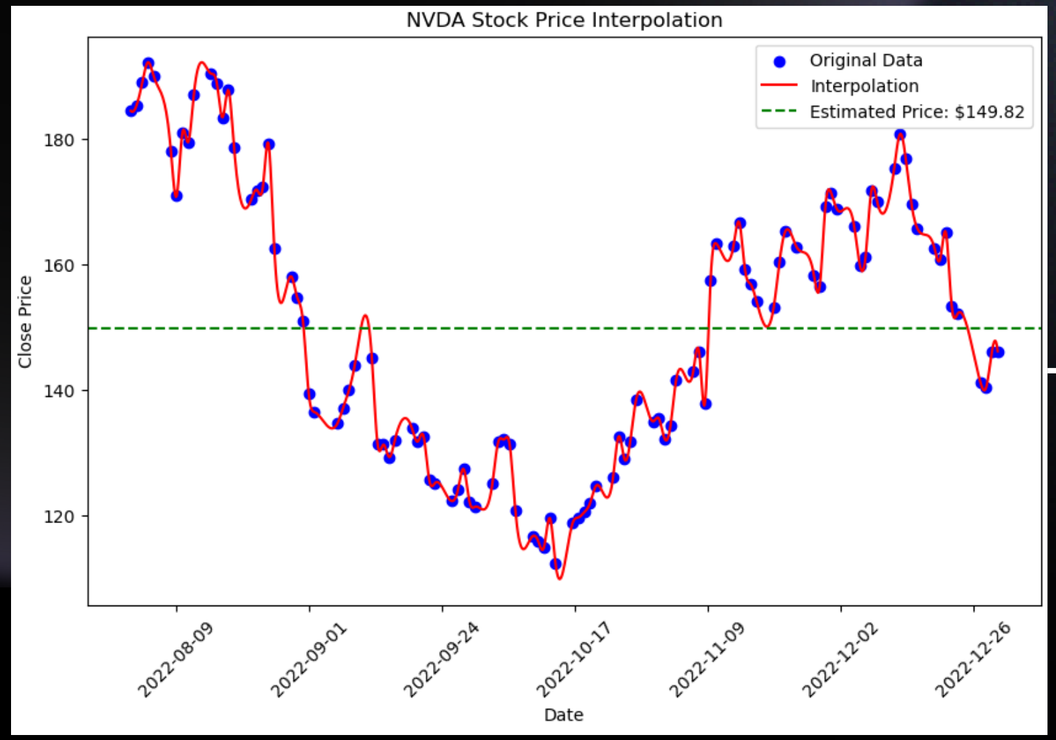
Composite Trapezoidal Rules

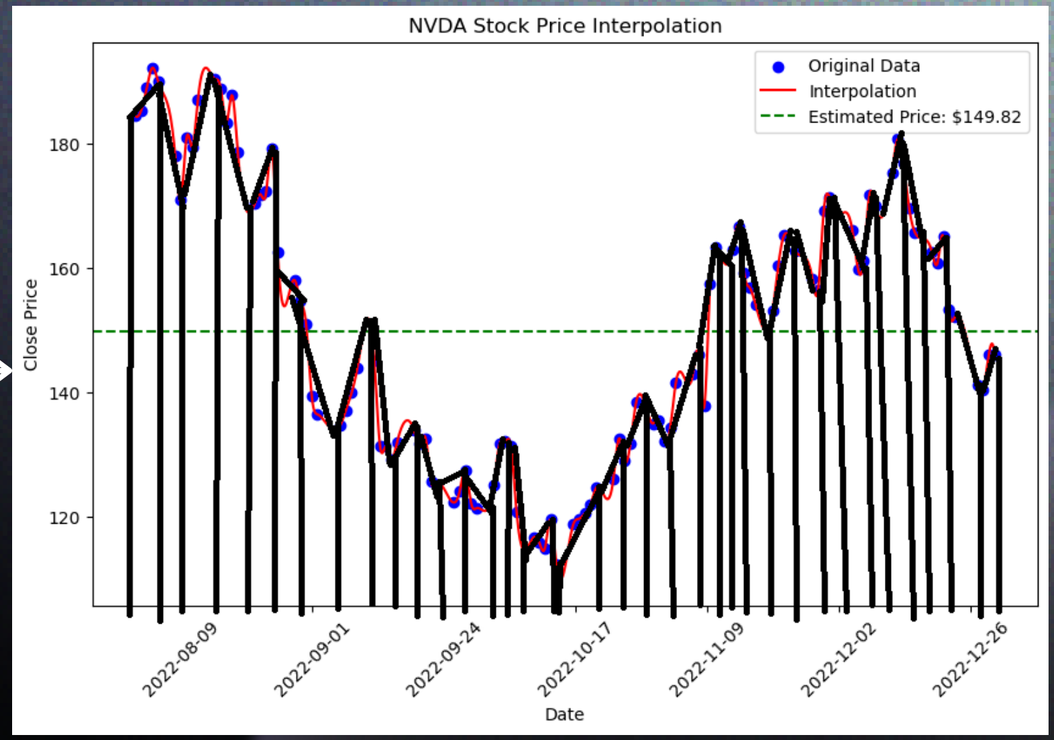
By calculating each part of the space approximated by multiple trapezoid shape, we can approximate the area of the shape

$$\int_a^b f(x)\,dx \approx \frac{b-a}{2n}\left[f(x_0) + 2f(x_1) + 2f(x_2) + \cdots + 2f(x_{n-1}) + f(x_n)\right]$$

METHODS AND OPERATIONS

Composite Trapezoidal Rules





Applications of the Numerical Analysis in Python

```python
import yfinance as yf
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import CubicSpline
from datetime import datetime
```

1. **Import necessary libraries:**
    - **yfinance:** Fetches stock data from Yahoo Finance.
    - **numpy:** Numerical operations and handling arrays.
    - **matplotlib.pyplot:** Plotting the stock data and interpolation.
    - **scipy.interpolate:** Provides the CubicSpline class for interpolation.
    - **datetime:** Handles date and time operations.

Applications of the Numerical Analysis in Python

```python
def get_stock_data(ticker, start, end):
    df = yf.download(ticker, start=start, end=end)
    return df

def interpolate_stock_data(dates, prices):
    interp = CubicSpline(dates, prices)
    return interp
```

```python
def estimate_stock_price(interp, start_date, end_date, num_intervals):
    time_interval = np.linspace(start_date, end_date, num_intervals)
    estimated_prices = interp(time_interval)
    integral = np.trapz(estimated_prices, time_interval)
    return integral / (end_date - start_date)
```

2. Define functions:

- **get_stock_data**

Fetches stock data from Yahoo Finance for a given ticker between the specified start and end dates.

- **interpolate_stock_data**

Interpolates stock data using the cubic spline method given dates and prices.

- **estimate_stock_price**

Estimates the average stock price over a given period using the trapezoidal rule for numerical integration, given an interpolation function, a start date, an end date, and a number of intervals.

Applications of the Numerical Analysis in Python

```python
ticker = 'NVDA'
start = datetime(2022, 1, 1)
end = datetime(2022, 12, 31)
num_intervals = 100
prediction_date = datetime(2023, 1, 1)
```

3. **Set up parameters:**
   - **ticker:** The stock ticker symbol (**e.g.**, 'NVDA' or 'NVIDIA Corporation').
   - **start:** The start date for fetching the stock data.
   - **end:** The end date for fetching the stock data.
   - **num_intervals:** Number of intervals for interpolation.
   - **prediction_date:** The date for which the estimated stock price is displayed.

```python
stock_data = get_stock_data(ticker, start, end)
stock_data.reset_index(inplace=True)

stock_data['Timestamp'] = stock_data['Date'].apply(lambda x:
x.timestamp())

interp = interpolate_stock_data(stock_data['Timestamp'],
stock_data['Close'])

start_timestamp = stock_data['Timestamp'].iloc[0]
end_timestamp = stock_data['Timestamp'].iloc[-1]
estimated_price = estimate_stock_price(interp, start_timestamp,
end_timestamp, num_intervals)

print(f"Estimated stock price for {ticker} on {prediction_date}:
${estimated_price:.2f}")

plt.figure(figsize=(10, 6))
plt.scatter(stock_data['Timestamp'], stock_data['Close'],
label='Original Data', color='blue')
date_range = np.linspace(stock_data['Timestamp'].min(),
stock_data['Timestamp'].max(), num_intervals)
plt.plot(date_range, interp(date_range), label='Interpolation',
color='red')
plt.axhline(y=estimated_price, color='green', linestyle='--',
label=f'Estimated Price: ${estimated_price:.2f}')
plt.gca().xaxis.set_major_formatter(plt.FuncFormatter(lambda x, pos:
datetime.fromtimestamp(x).strftime('%Y-%m-%d')))
plt.xlabel('Date')
plt.ylabel('Close Price')
plt.title(f'{ticker} Stock Price Interpolation')
plt.legend()
plt.xticks(rotation=45)
plt.show()
```

4. Main code execution:

- Fetch stock data using the `**get_stock_data**` function.
- Reset the index of the stock data DataFrame to ensure proper indexing.
- Convert the dates in the stock data to numerical values (timestamps) for interpolation purposes.
- Interpolate the stock prices using the `**interpolate_stock_data**` function, which returns a cubic spline interpolation function.
- Estimate the stock price for the prediction date using the `**estimate_stock_price**` function and the trapezoidal rule.
- Print the estimated stock price.
- Plot the original stock data, the interpolation, and the estimated stock price using matplotlib.pyplot for visualization purposes.

POSSIBLE IMPROVEMENTS

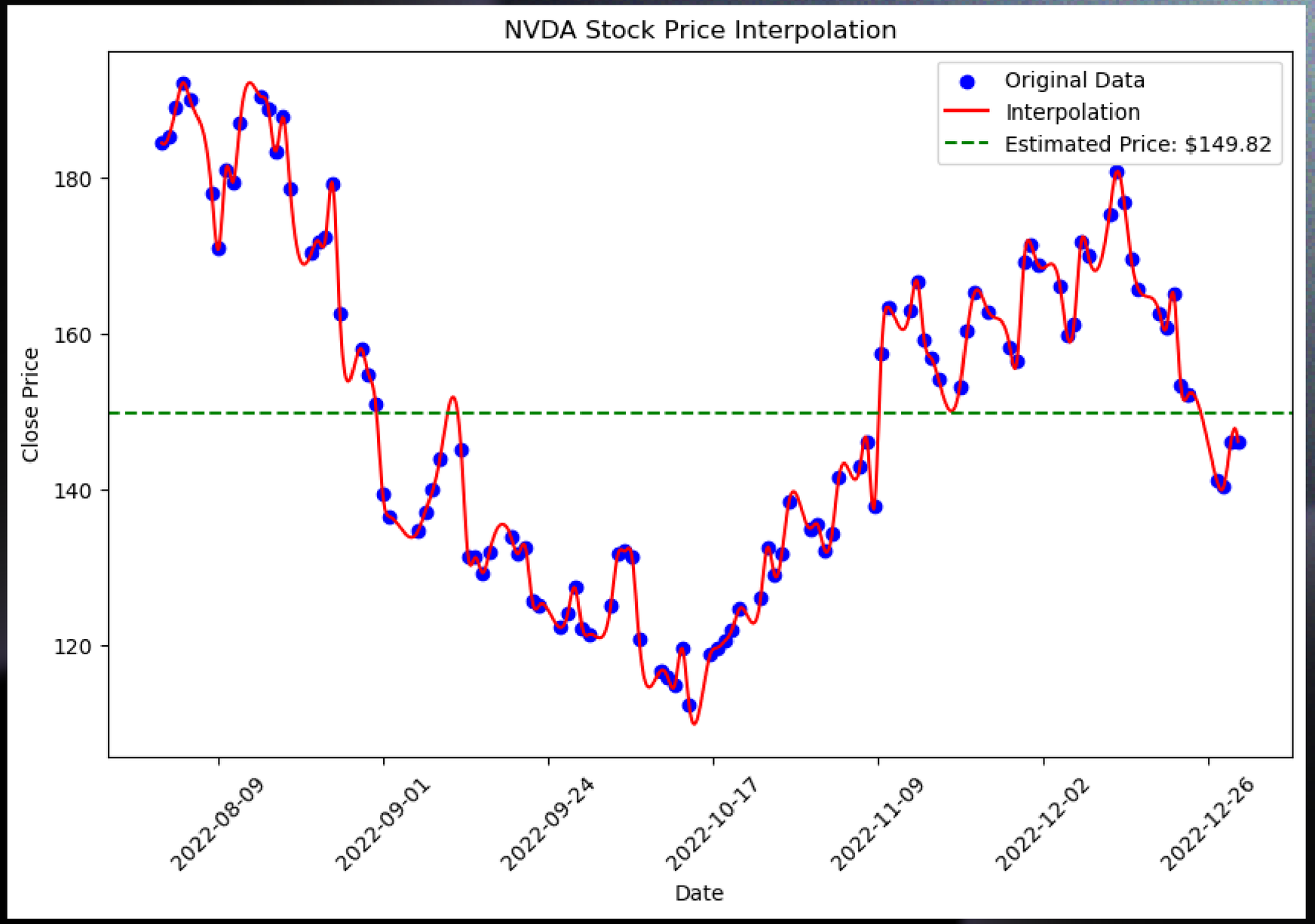
Expected Outcome and Results

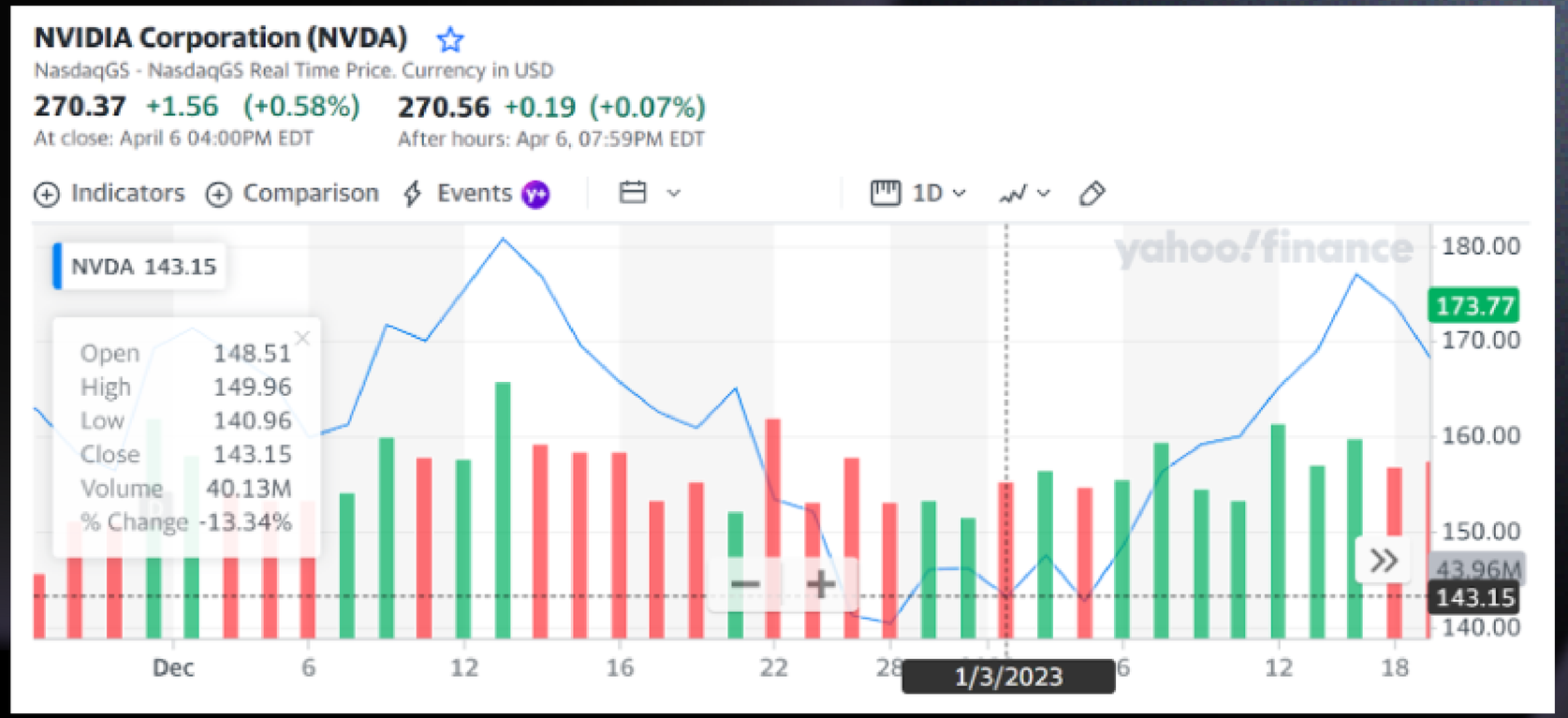
Using the following settings of parameters:

```python
ticker = 'NVDA' # NVIDIA Corporation
start = datetime(2022, 8, 1) # 1st of August 2022
end = datetime(2022, 12, 31) # 31st of December 2022
num_intervals = 1000
prediction_date = datetime(2023, 1, 3) # 3rd of January 2023
```

The results of the execution of the Python script are as follows:

```
Estimated stock price for NVDA on 2023-01-03 00:00:00: $149.82
```


NVDA Stock Price Interpolation
Original Data
Interpolation
Estimated Price: $149.82
Close Price
180
160
140
120
2022-08-09
2022-09-01
2022-09-24
2022-10-17
2022-11-09
2022-12-02
2022-12-26
Date



1. **Absolute Error**

$$|Estimated - Actual|$$
$$= |149.82 - 143.15|$$
$$= 6.67$$

2. **Relative Error**

$$\frac{|Estimated - Actual|}{Actual}$$
$$= \frac{|149.82 - 143.15|}{143.15}$$
$$\approx 0.046594$$

3. **Percentage Error**

$$\frac{|Estimated - Actual|}{Actual} \times 100$$
$$= \frac{|149.82 - 143.15|}{143.15} \times 100$$
$$\approx 4.659448\ \%$$

CONCLUSION

Thank You For Your Attention

Feel free to ask us anything!

Presented By:
Kornpasit Theerathitayangkul, Jirapit Sripitak, Arthakorn Raksenawanich

Presented To:
Asst. Prof. Pallop Huabsomboon