Digital Signal Processing HW5

Part 2: Discrete Fourier Transform, Applications

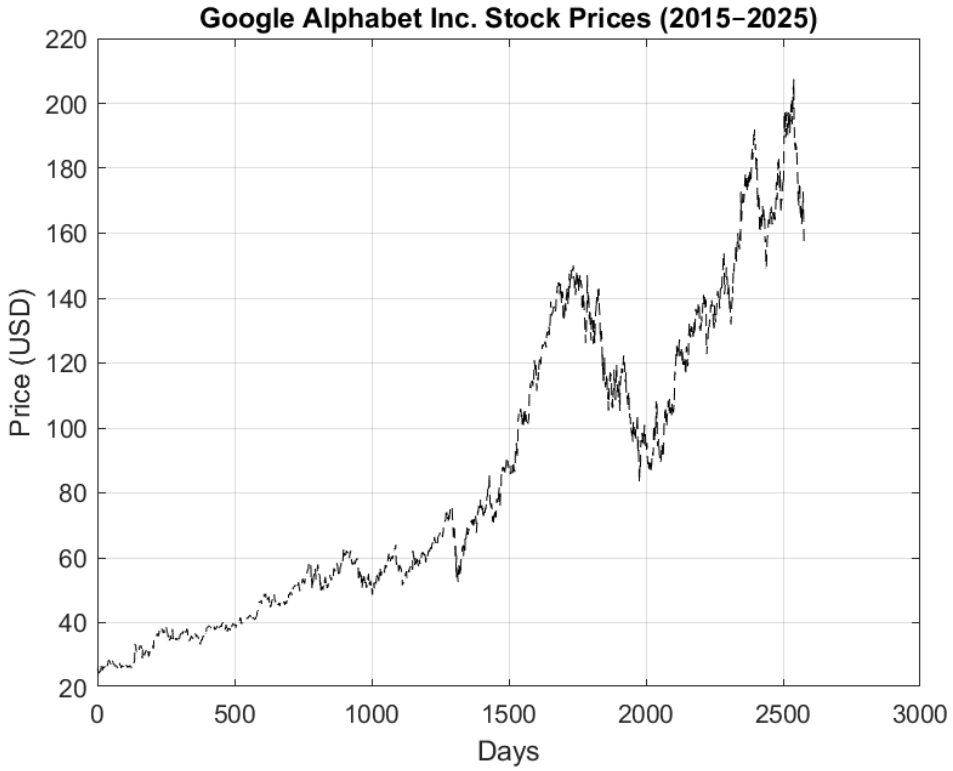
Problem 1: Download and Graph Data

The file `StockData_2015_2025.csv` contains daily stock prices of Google Alphabet Inc. from 2015 to 2025. The first column represents the day index, and the second column contains the corresponding closing prices in USD.

The data was imported using the `csvread` function in Octave and plotted using the following commands.

Complete Octave code:

```
data = csvread('StockData_2015_2025.csv');
plot(data(:,1), data(:,2), '--', 'color', 'black');
xlabel('Days');
ylabel('Price (USD)');
title('Google Alphabet Inc. Stock Prices (2015{2025)');
grid on;
print('stock_price_plot.png', '-dpng');
```



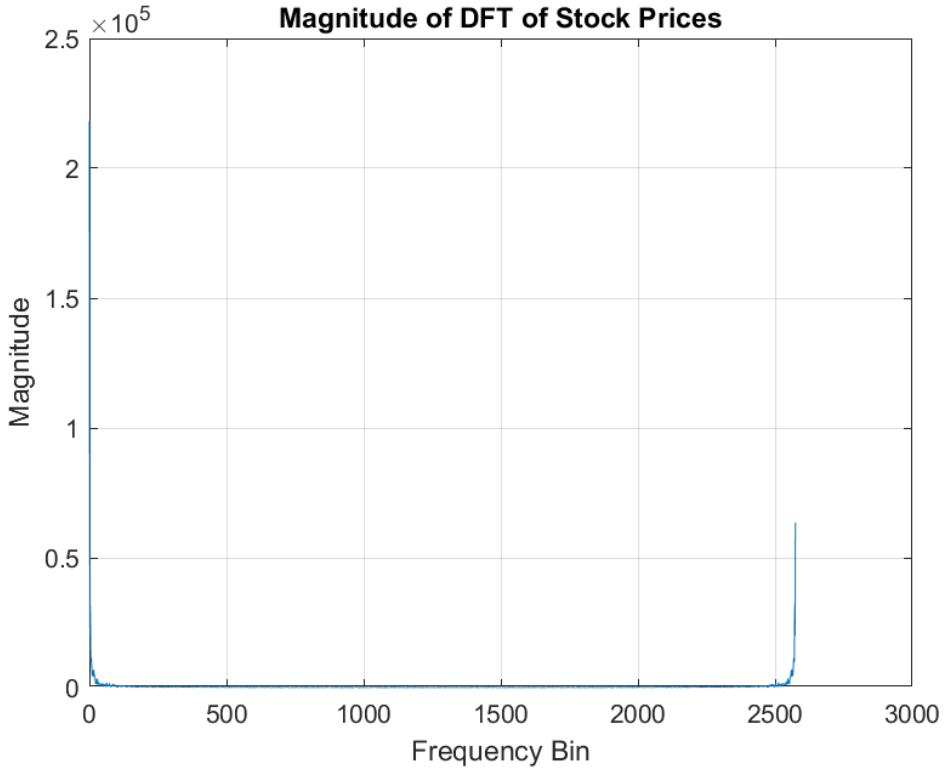
The graph illustrates the trend of Google Alphabet Inc.'s stock price over a 10-year period. The dashed black line shows how the price generally rises over time, with noticeable fluctuations due to market conditions and external events.

Problem 2: Create the Discrete Fourier Transform

Using the daily stock closing prices from the previous problem, we apply the Discrete Fourier Transform (DFT) to analyze the frequency content of the data. The goal is to compute and graph the magnitude spectrum.

Complete Octave code:

```
data = csvread('StockData_2015_2025.csv');
prices = data(:, 2);
N = length(prices);
X = fft(prices);
f = (0:N-1);
magnitude = abs(X);
figure;
plot(f, magnitude);
xlabel('Frequency Bin');
ylabel('Magnitude');
title('Magnitude of DFT of Stock Prices');
grid on;
print('stock_dft_magnitude.png', '-dpng');
```



The plot shows the magnitude of the Fourier transform of the stock price data. The large spike at low frequencies indicates a slow-changing trend or long-term market momentum. High-frequency components are much smaller, showing minimal rapid fluctuation in the daily data.

Problem 3: Identify Peaks and Frequencies

To analyze the periodic behavior of the stock price, we computed the magnitude of the Discrete Fourier Transform and identified the strongest frequency components.

The following Octave code was used to sort and report the five largest peaks (excluding the DC component):

```
data = csvread('StockData_2015_2025.csv');
prices = data(:, 2);
N = length(prices);
X = fft(prices);
magnitude = abs(X);
magnitude(1) = 0;
[sorted_vals, sorted_idx] = sort(magnitude, 'descend');
peak_bins = sorted_idx(1:5);
peak_mags = sorted_vals(1:5);
for i = 1:5
    fprintf('Peak %d: Bin %d, Magnitude %.2f\n', i, peak_bins(i), peak_mags(i));
end
```

Output:

```
Top 5 Frequency Peaks:
Peak 1: Bin 2, Magnitude 63553.60
Peak 2: Bin 2575, Magnitude 63553.60
Peak 3: Bin 4, Magnitude 32002.85
Peak 4: Bin 2573, Magnitude 32002.85
Peak 5: Bin 5, Magnitude 26039.32
```

Frequency Interpretation:

Assuming the stock data includes one sample per trading day (approximately 252 trading days/year), we convert the bins into frequency in cycles per year using:

$$\text{Frequency (cycles/year)} = \frac{\text{Bin}}{N} \times 252$$

- Bin 2 $\rightarrow$ 0.196 cycles/year
- Bin 4 $\rightarrow$ 0.392 cycles/year
- Bin 5 $\rightarrow$ 0.490 cycles/year

These low-frequency peaks suggest the stock exhibits long-term cyclical trends, with dominant periodicities on the scale of 2 to 5 years.

Part 3: Filter Design

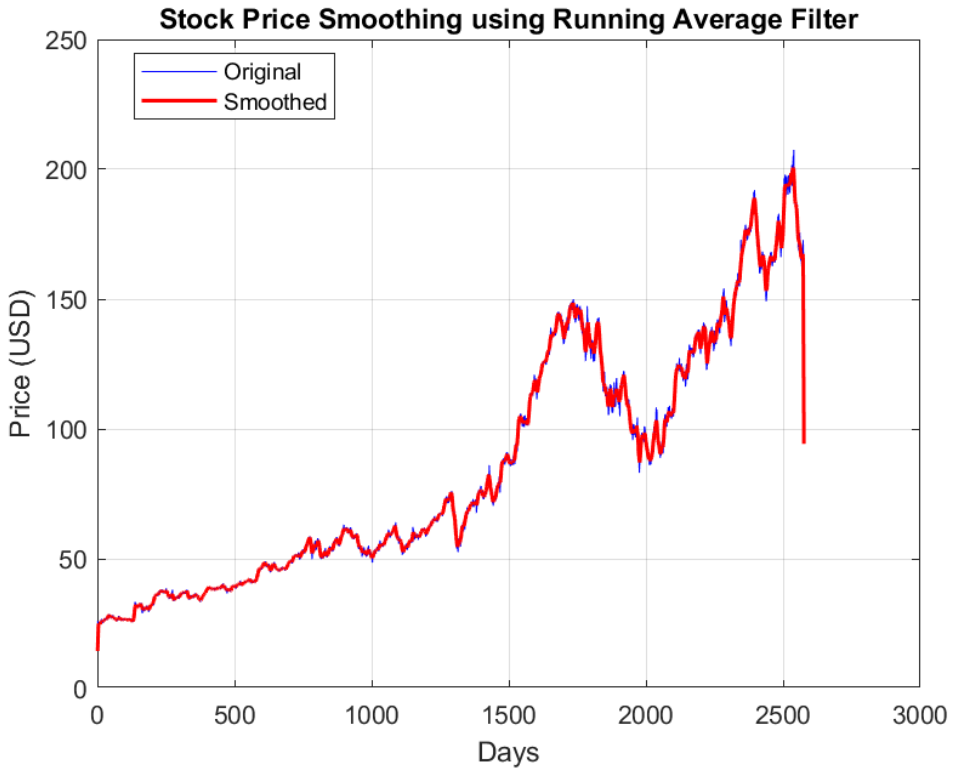
Problem 1: Smoothing the Time Series Data

To smooth the stock price time series, we implemented a simple running average filter using convolution. This filter kernel averages each data point with its neighbors in a 7-day window, thereby removing short-term fluctuations while preserving the overall trend.

Complete Octave code:

```
data = csvread('StockData_2015_2025.csv');
days = data(:, 1);
prices = data(:, 2);

window_size = 7;
h = ones(1, window_size) / window_size;

smoothed = conv(prices, h, 'same');

figure;
plot(days, prices, 'b', 'DisplayName', 'Original');
hold on;
plot(days, smoothed, 'r', 'LineWidth', 1.5, 'DisplayName', 'Smoothed');
```

```
xlabel('Days');
ylabel('Price (USD)');
legend('Location', 'best');
title('Stock Price Smoothing using Running Average Filter');
grid on;

print('stock_running_average.png', '-dpng');
```



The red line represents the smoothed version of the original time series. The filter reduces daily volatility and helps reveal underlying trends in the stock price.

Problem 2: Graph the Filtered Spectrum

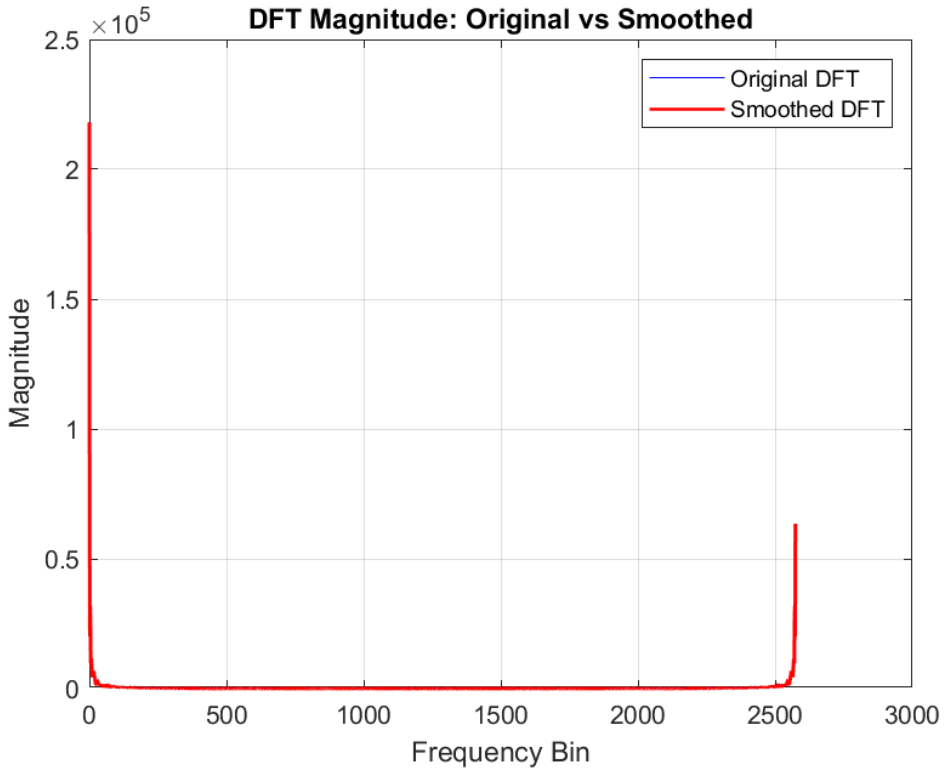
We computed the Discrete Fourier Transform (DFT) of both the original and smoothed stock price data and compared their magnitude spectra.

A 7-point running average filter was applied using convolution. As expected, the smoothed signal had reduced high-frequency content due to the filter's low-pass characteristics.

Complete Octave code:

```
data = csvread('StockData_2015_2025.csv');
prices = data(:, 2);
N = length(prices);

X_raw = fft(prices);
```

```
mag_raw = abs(X_raw);

h = ones(1, 7) / 7;
smoothed = conv(prices, h, 'same');

X_smooth = fft(smoothed);
mag_smooth = abs(X_smooth);

f = (0:N-1);

figure;
plot(f, mag_raw, 'b', 'DisplayName', 'Original DFT');
hold on;
plot(f, mag_smooth, 'r', 'LineWidth', 1.2, 'DisplayName', 'Smoothed DFT');
xlabel('Frequency Bin');
ylabel('Magnitude');
legend('Location', 'northeast');
title('DFT Magnitude: Original vs Smoothed');
grid on;

print('stock_dft_comparison.png', '-dpng');
```



The graph shows the DFT magnitude of both the raw and smoothed signals. The red

(smoothed) spectrum closely overlaps the blue (original) at low frequencies but is significantly reduced at higher frequencies, confirming that the running average acts as a low-pass filter.