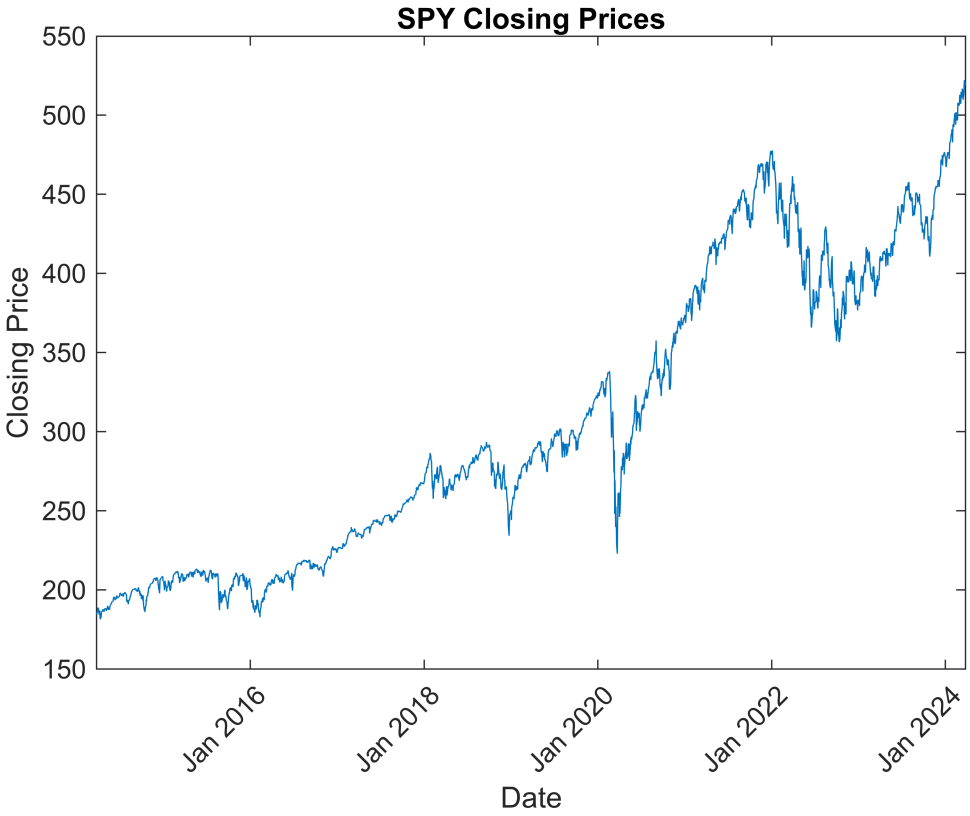
DSP Lab 11

Syed Asghar Abbas Zaidi 07201

Task 1

```
% Step 1: Import data from 'SPY.csv' into MATLAB
warning('off', 'all')
data = readtable('SPY.csv');

% Step 2: Extract 'Date' and 'Close' columns
date = data.Date;
close_price = data.Close;

% Step 3: Convert 'Date' column to datetime format
date = datetime(date, 'InputFormat', 'MM/dd/yyyy');

% Step 4: Plot the closing prices
plot(date, close_price);
xlabel('Date');
ylabel('Closing Price');
title('SPY Closing Prices');

% Adjust the date format on the x-axis
dateformat = 'mmm yyyy'; % Change as needed
datetick('x', dateformat, 'keeplimits');

% Rotate x-axis labels for better readability
xtickangle(45);
```



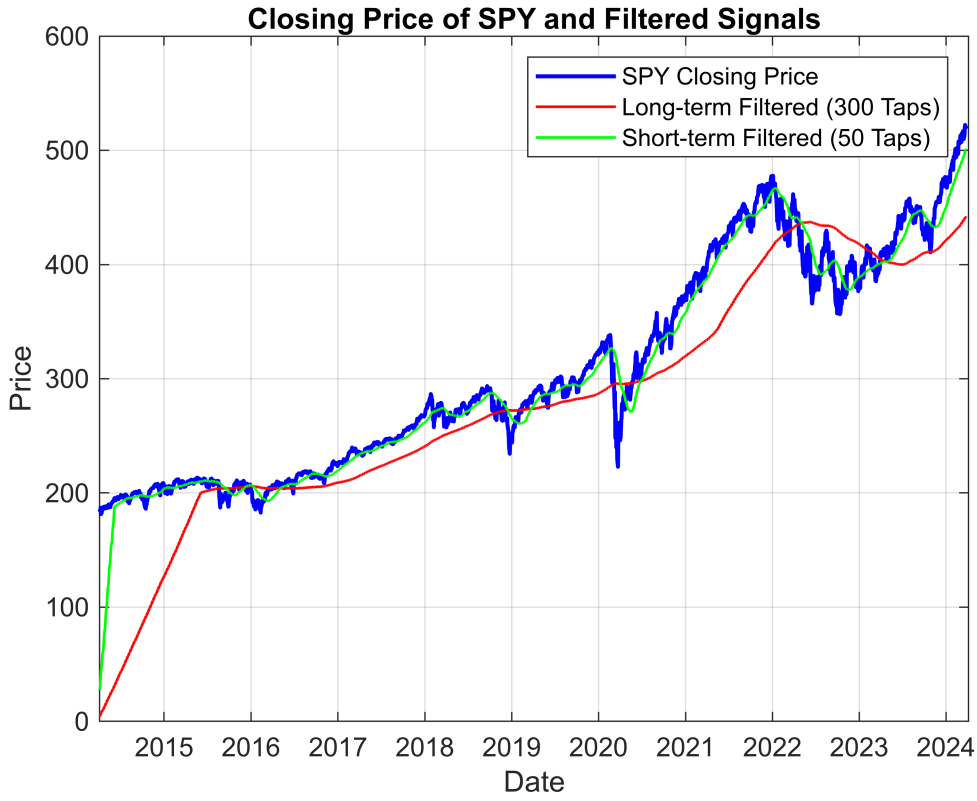
ONLINE CHART:

ANALYSIS:

As can be observed, our graph looks exactly the same as the one found online.

The MATLAB code provided performs several key steps to visualize the daily closing prices of the SPY ETF over the past 10 years. It begins by importing data from the "SPY.csv" file using `readtable()` and extracts the Date and Close columns to prepare for plotting. Conversion of the Date column to datetime format is carried out to facilitate compatibility with MATLAB's plotting functions. The plot() function is then employed to create a line plot, depicting closing prices against dates. Axis labels (Date and Closing Price), as well as a plot title (SPY Closing Prices), are added for clarity using xlabel(), ylabel(), and title() respectively. Furthermore, the date format on the x-axis is adjusted via datetick() for improved readability, while xtickangle(` is utilized to rotate x-axis labels. Ultimately, this MATLAB code delivers a comprehensive visualization of SPY's closing prices, allowing for comparison with data presented on Yahoo Finance.

```
% Design the filters
longTermFilter = ones(1, 300) ./ 300;
shortTermFilter = ones(1, 50) ./ 50;

% Apply the filters
longTermFiltered = filter(longTermFilter, 1, close_price);
shortTermFiltered = filter(shortTermFilter, 1, close_price);

% Plotting
figure;
plot(date, close_price, 'b', 'LineWidth', 1.5);
hold on;
plot(date, longTermFiltered, 'r', 'LineWidth', 1);
plot(date, shortTermFiltered, 'g', 'LineWidth', 1);
hold off;

% Formatting
xlabel('Date');
ylabel('Price');
title('Closing Price of SPY and Filtered Signals');
legend('SPY Closing Price', 'Long-term Filtered (300 Taps)', 'Short-term Filtered
(50 Taps)');
grid on;

% Limiting x-axis to last 10 years
datetick('x', 'yyyy');
xlim([datetime('now')-years(10), datetime('now')]);
```



Analysis:

To accomplish the task outlined, the provided MATLAB code first designs two filters, a long-term filter with 300 taps and a short-term filter with 50 taps, both normalized to ensure the sum of their coefficients equals one. These filters are then applied to the time-series of SPY's closing prices, generating two new time-series: one representing the output of the long-term filter (y_filtered1) and the other representing the output of the short-term filter (y_filtered2). These filtered time-series are then plotted alongside the original closing prices of SPY over the last 10 years. Finally, the plot is formatted with appropriate labels, titles, and legends to clearly visualize the closing prices and the output of the long-term and short-term filters. The x-axis is limited to display data from the last 10 years for clarity. This visualization allows for a comparison between the original closing prices and the filtered signals, aiding in the identification of trends and momentum signals for potential trading strategies.

Task 3

```
Momentum = shortTermFiltered - longTermFiltered; %Momentum of our signal
% The initial investment
ini_inv = 10000; %Our Initial Investment
C = ini_inv;
Sh = 0; %our shares
PV = zeros(size(date)); %Our Portfolio value
HV = zeros(size(date)); %Our Hold value
```

```
% Here we are simulating "trading strategy" and "holding strategy"
for j = 1:numel(date)
    if Momentum(j) > 0 && C > 0
        % Buying shares!
        Sh = C / close_price(j);
        C = 0;
    elseif Momentum(j) < 0 && Sh > 0
        % Selling shares!
        C = Sh * close_price(j);
        Sh = 0;
    end
    % Calculate portfolio value at each point (trading strategy)
    PV(j) = C + Sh * close_price(j);
    % Calculate hold value at each point (holding strategy)
    HV(j) = ini_inv * (close_price(j) / close_price(1));
end

clf;
% Plotting
plot(date, PV, 'm', 'DisplayName', 'Trading Strategy');
hold on;
plot(date, HV, 'k', 'DisplayName', 'Hold Strategy ');
hold off;
xlabel('Date');
ylabel('Value of Portfolio');
title('Trading Strategy vs Hold Strategy for SPY');
legend('Location', 'best');
grid on;
```

Analysis:

To accomplish the task outlined, the code first calculates the momentum signal by subtracting the output of the long-term filter from the output of the short-term filter. Then, it simulates a trading strategy over a 10-year period for the SPY ETF. Starting with an initial investment of $10,000, the script iterates through the time-series data, making decisions to buy or sell shares based on the momentum signal. If the momentum signal is positive and cash is available, shares are purchased; if the signal is negative and shares are held, they are sold.

Portfolio values are computed at each point for both the trading strategy and a simple holding strategy where SPY shares are held for the entire period. The resulting portfolio values are plotted over time, allowing for a comparison between the performance of the trading strategy and the holding strategy. This exercise provides insights into the effectiveness of the momentum-based trading strategy compared to a passive buy-and-hold approach, aiding in the evaluation of its potential for generating returns in real-world trading scenarios. It also helps in understanding the impact of timing decisions based on technical indicators on investment performance and reinforces concepts related to signal processing and algorithmic trading strategies.

Observation:

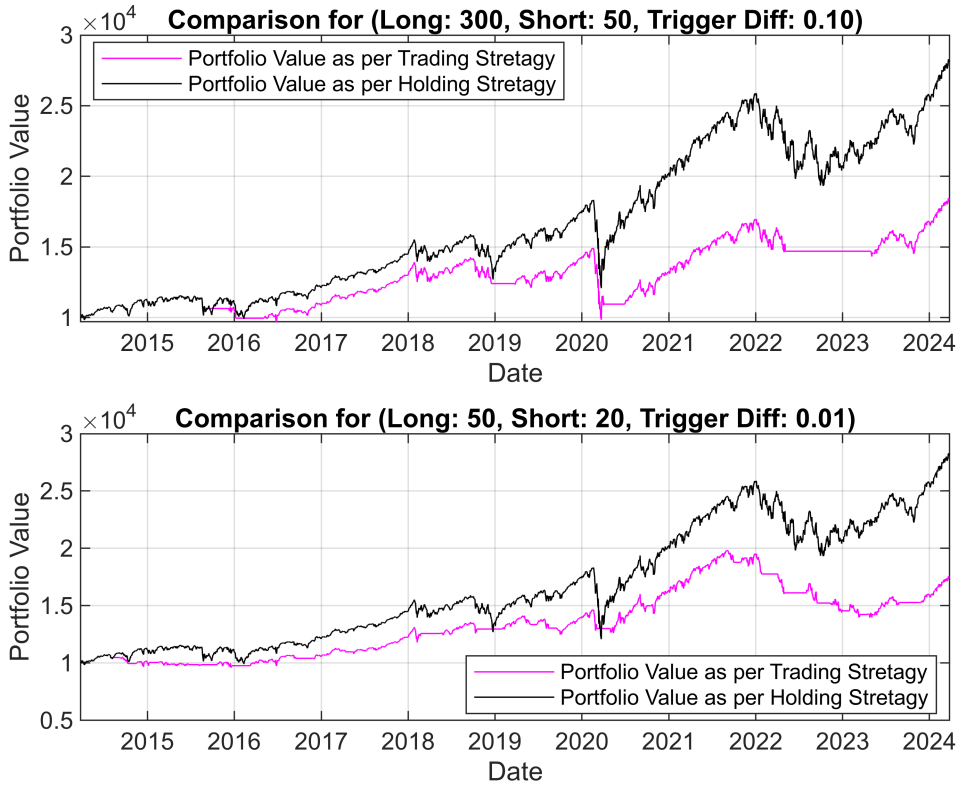
From the result we have gotten, we can observe that our "trading stategy" is not that much effective in giving us the high returns we want, especially when we compare it to the "holding strategy" which gives higher returns to this.

Task 4

```
clf;
% Here we are defining various combinations of the filter parameters
```

```
% %will allow us ease later on
FP = [
    struct('ltaps', 300, 'staps', 50, 'trig_diff', 0.1),
    struct('ltaps', 50, 'staps', 20, 'trig_diff', 0.01)
];
% Initialize the plot
figure;
for k = 1:numel(FP)
% depending on which loop number we are, the filter parameters will be
% extracted for the current combination accordingly
    params = FP(k);
% Here we are designing long-term and short-term filters
    Long_Term = ones(1, params.ltaps) ./ params.ltaps;
    Short_Term = ones(1, params.staps) ./ params.staps;
% We are applying filters to the closing price below
    Long_Term_Filtered = filter(Long_Term, 1, close_price);
    Short_Term_Filtered = filter(Short_Term, 1, close_price);
% Calculating the momentum
    MS = Short_Term_Filtered - Long_Term_Filtered; %Momentum Signal
% Initial investment amount
    Ini_Inv = 10000; %Initial Investment
    C = Ini_Inv; %Cash
    SH = 0; %Shares
    PV = zeros(size(date)); %Portfolio value
    HV = zeros(size(date)); %Hold Value


% Now are simulating "trading strategy" and "holding strategy"
for h = 1:numel(date)
if MS(h) > params.trig_diff && C > 0
% Buying shares!
           SH = C / close_price(h);
           C = 0;
elseif MS(h) < -params.trig_diff && SH > 0
% Selling shares!
           C = SH * close_price(h);
           SH = 0;
end
% Now at each point, calculate portfolio value using trading strategy
       PV(h) = C + SH * close_price(h);
% Now at each point, calculate hold value using holding strategy
       HV(h) = Ini_Inv * (close_price(h) / close_price(1));
end
% Below, we are just plotting things.
    subplot(numel(FP), 1, k);
    plot(date, PV, 'm', 'DisplayName', 'Portfolio Value as per Trading Stretagy');
    hold on;
    plot(date, HV, 'k', 'DisplayName', 'Portfolio Value as per Holding Stretagy');
    hold off;
    xlabel('Date');
```

```
    ylabel('Portfolio Value');
    title(sprintf('Comparison for (Long: %d, Short: %d, Trigger Diff: %.2f)', ...
        params.ltaps, params.staps, params.trig_diff));
    legend('Location', 'best');
    grid on;
end
```



Analysis:

The provided MATLAB code implements a trading strategy simulation using different combinations of filter parameters to analyze their impact on portfolio performance. By applying long-term and short-term filters to historical closing price data of the SPY ETF, the code generates momentum signals to inform buy or sell decisions. The simulation compares the portfolio value over time between the trading strategy and a simple holding strategy, providing insights into the effectiveness of the trading strategy under varying filter parameters.

This analysis aids in optimizing filter parameters for improved trading strategy performance and enhances understanding of how different technical indicators influence investment decisions. Additionally, by observing the plotted portfolio values, users can assess the trade-offs between active trading based on short-term market trends and passive buy-and-hold strategies. This approach enables investors to refine their trading strategies, potentially increasing returns or reducing risk by adapting to changing market conditions. Moreover, the code's modularity allows for easy experimentation with different parameter combinations, facilitating iterative refinement of trading strategies for enhanced financial outcomes.

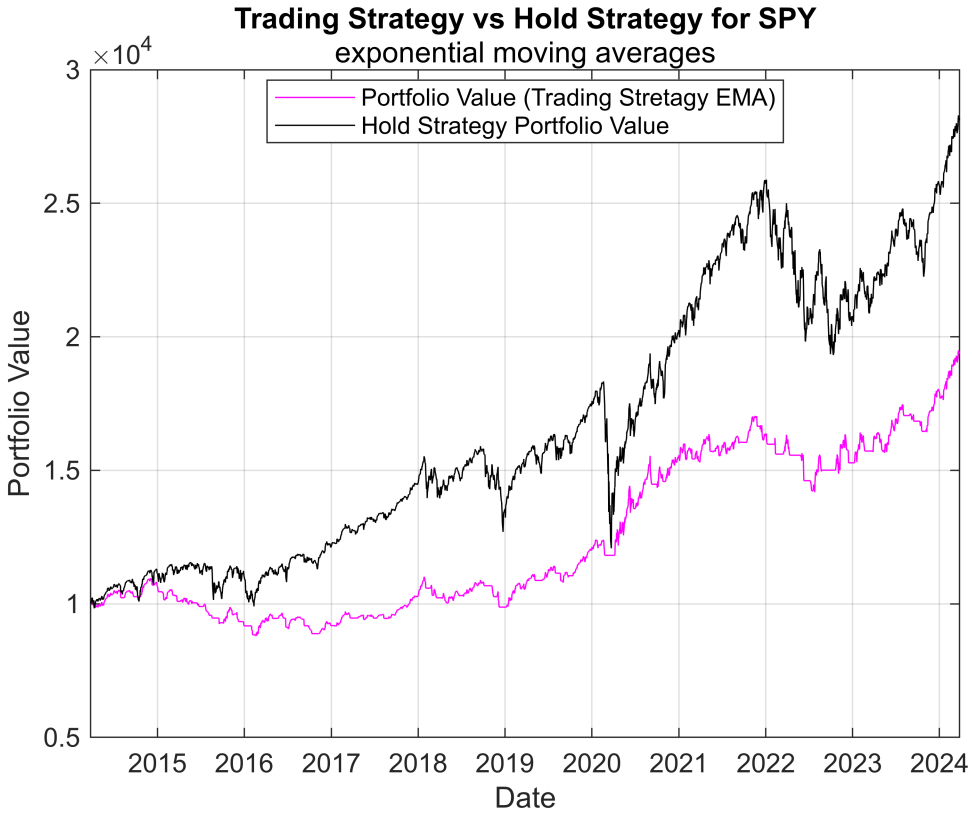
As I changed filter parameters from first to second combination, we see pretty less of a difference between the two strategies at hand however, holding out strategy still proves to be better in giving us better returns.This

reduced difference may be the result from the active trading based on shorter-term market fluctuations, which might improve the results.

Post Lab

```
 % Here we are defining EMA parameters
l_alpha = 0.15; % Long-term
s_alpha = 0.3; % Short-term

% Calculate exponential moving averages aka EMAs
l_ema = zeros(size(close_price));
s_ema = zeros(size(close_price));
for idx = 1:numel(close_price)
    if idx == 1
        l_ema(idx) = close_price(idx);
        s_ema(idx) = close_price(idx);
    else
        l_ema(idx) = l_alpha * close_price(idx) + (1 - l_alpha) * l_ema(idx-1);
        s_ema(idx) = s_alpha * close_price(idx) + (1 - s_alpha) * s_ema(idx-1);
    end
end

% momentum signal
MS = s_ema - l_ema;

% Initial investment amount
Ini_inv = 10000; C = Ini_inv; SH = 0;
PV = zeros(size(date));HV = zeros(size(date));

% trading strategy and holding strategy
for s = 1:numel(date)
    if MS(s) > 0 && C > 0
        % Buy shares
        SH = C / close_price(s);
        C = 0;
    elseif MS(s) < 0 && SH > 0
        % Sell shares
        C = SH * close_price(s);
        SH = 0;
    end

    % portfolio value at a point trading strategy
    PV(s) = C + SH * close_price(s);

    % hold value at a point, holding strategy
    HV(s) = Ini_inv * (close_price(s) / close_price(1));
end
```

```
clf;

% Plotting everything
plot(date, PV, 'm', 'DisplayName', 'Portfolio Value (Trading Stretagy EMA)');
hold on;
plot(date, HV, 'k', 'DisplayName', 'Hold Strategy Portfolio Value');
hold off;
xlabel('Date');
ylabel('Portfolio Value');
title('Trading Strategy vs Hold Strategy for SPY', 'exponential moving averages');
legend('Location', 'best');
grid on;
```



Analysis:

The code implements a trading strategy utilizing exponential moving average (EMA) filters to analyze price trends and make buy or sell decisions. By adjusting parameters for the long-term and short-term EMAs, the strategy aims to capture short-term price movements more effectively compared to traditional moving average filters. The EMA-based approach results in smoother portfolio value fluctuations, indicating potentially more stable performance and reduced sensitivity to market volatility.

This smoother nature of the EMA graph suggests that the trading strategy may offer improved risk management capabilities, as it can adapt more swiftly to changing market conditions. However, further analysis and parameter fine-tuning are necessary to comprehensively evaluate the effectiveness of the EMA-based strategy compared to traditional methods. In contrast to previous strategies, which relied on simple moving averages, the

use of EMA filters allows for more dynamic and responsive trading decisions, potentially leading to enhanced returns and risk-adjusted performance.

Overall, the adoption of EMA filters offers a promising avenue for refining trading strategies and navigating complex financial markets with greater precision and confidence.