Hedging assignment

Tuomas Myllymäki

December 31, 2021

1 Introduction

In this report, we'll examine the different methods of using call and put option contracts for hedging market risk in the index securities market. As a case study, we'll go through and apply both delta and delta-vega hedging strategies in order for us to obtain a situation in which the total changes of the replicating and underlying portfolios are cancelled.

As background we consider the vanilla Black-Scholes model for option pricing:

$$\begin{aligned} C^{\mathrm{BS}}(t, S_t; E; T; \sigma) &= S_t \mathcal{N}(d_1) - E e^{-r(T-t)} \mathcal{N}(d_2) \\ d_1 &= \frac{\log(S_t/E) + (r + \sigma^2/2)(T-t)}{\sigma\sqrt{T-t}} \\ d_2 &= \frac{\log(S_t/E) + (r - \sigma^2/2)(T-t)}{\sigma\sqrt{T-t}} \end{aligned}$$

We use this model to calculate the implied volatility and using that we can compute the required greeks for the different hedging strategies.

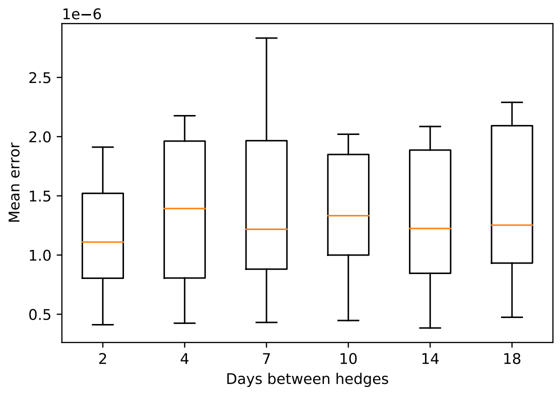
The main metric we will use for analyzing the performance of different strategies is the mean squared error (MSE):

$$\mathrm{MSE} = \frac{1}{N} \sum_{i=1}^{N} A_i^2$$

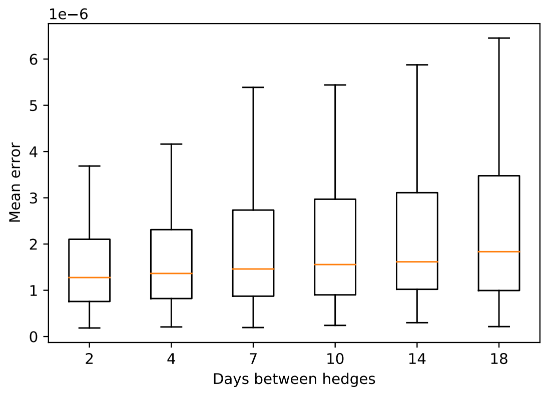
Where A_i is the error for one specific day. In our analyses we will also compute the mean of MSEs for different variables (strike price, time to maturity, etc.). At the end, we will also analyze the standard deviation of A^2 in order to compare the stability different hedging methods.

2 Delta hedging

In this section we'll first briefly cover the theoretical background for delta hedging and then use it for hedging a single option portfolio. We consider a portfolio P^D consisting of a long stock option (C_t) and short Δ amount of the underlying stock S. Therefore, the value of the portfolio at time t is $C_t - \Delta_t S_t$. The changes in the value of the call option are neutralized by the changing amount of the underlying stock.



(a) $T = 40$ ATM call option.



(b) The average hedging error for an ATM call.

Figure 1: The hedging error as a function of the hedging frequency.

2.1 Frequency analysis

In this section we'll analyze the hedging of a single call option with Delta-hedging. In particular, we analyze the hedging performance as a function of the rehedging frequency.

We pick a maturity time T, strike price E from a specific data sheet and see what the result is. We first picked $T = 40$ and a strike price closest to the current stock price (ATM). The amount of days in between hedges F was ranged ($F \in \{2, 4, 7, 10, 14, 18\}$). In figure 1 the Delta hedging performance is depicted as a function of the days in between hedges for different strike prices.

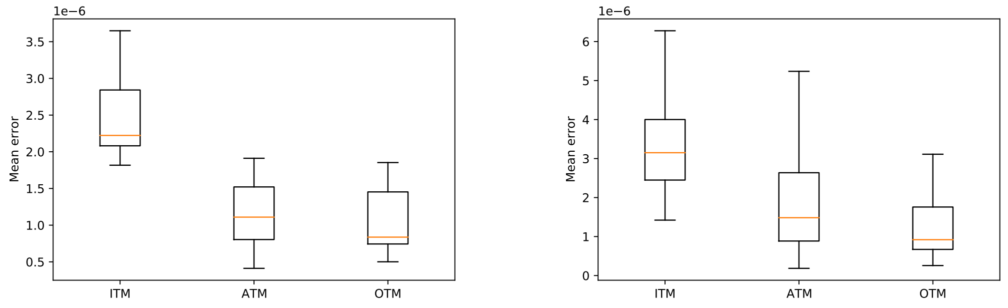
From figure 1(a), we can see that increasing the days in between hedging affects negatively on the performance. However, no significant change to the overall performance is exhibited. Next, we'll examine the same phenomenon but considering all possible maturity times. In other words we let T range from 20 to 44. The result can be found in figure 1(b).

From 1(b) we can clearly see that increasing the day in between hedges affects the overall performance negatively. This makes intuitive sense because with more days in between the value of the underlying has changed more that less days in between hedges. Although, the effect is still quite minor and even with 18 days in between hedges, we still get a fairly effective hedge.

2.2 Strike price analysis

In this section, we'll examine the effects strike price E has on the Delta-hedging performance. We'll again start with $T = 40$ and we now fix the days in between hedges $F = 2$. To analyze the effect of strike prices, we let the strike price E range in such a way that the price is either -10, 0, or 10 units more than the current ATM strike price. In other words, we compare ITM, ATM and OTM call options.

In figure 2(a) we can see that the best performance was gained by using OTM options. Their performance is quickly followed by the ones made using ATM options. ITM options performed the worst of the 3. We repeat the same analysis for many different days between hedges and maturity times. We let T range from 20 to 44 using increments of 2 and the hedging days are ranged in the previously mentioned range. The result can be seen from figure 2(b).



(a) $T = 40, F = 2$ (b) Average over all maturities and hedging days.

Figure 2: Strike price's effects on the error.

From the figure 2(b), we note that overall, the best options are OTM options and ITM options are the worst.

2.3 Combined analysis

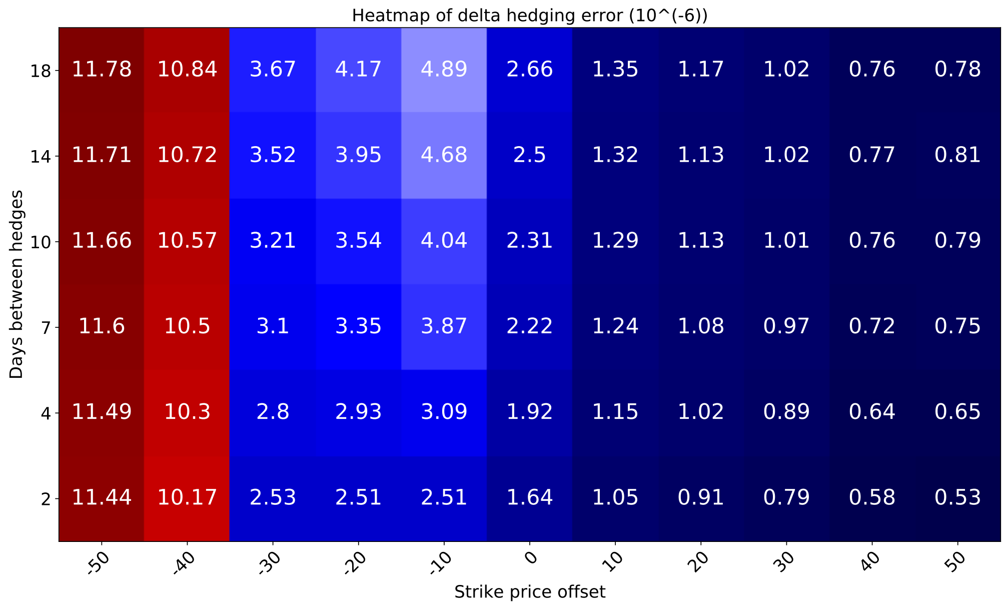


Figure 3: The Delta hedging error heatmap (10^{-6} scale).

In this section, we'll finish Delta hedging analysis by considering both the hedging days and the offset from ATM price. We now let the strike price E be offset from the ATM option price by 50 units in either direction. We also let the hedging days fluctuate similarly as before. Using those datapoints we can plot the hedging result in figure 3.

The heatmap shows the same behaviour as the previous analysis have indicated. Namely, increasing the days between hedges tends to worsen the hedging error. Also we note that the best hedging performance is achieved with OTM options. On the other hand, the ITM options clearly perform the worst. These findings are consistent with our previous analyses.

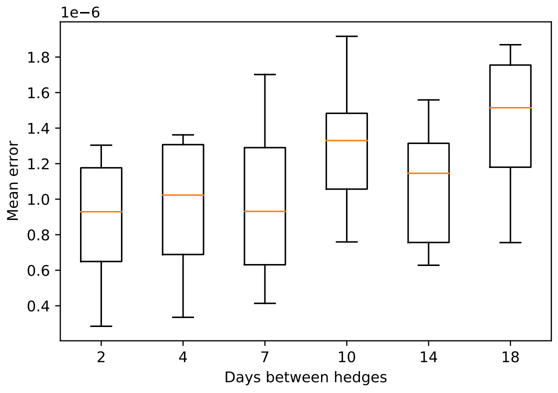
3 Delta-Vega Hedging

We have previously analyzed the delta hedging strategy thoroughly and we have seen how different variables affect the hedging error. Now we turn our attention to delta-vega hedging. Let P^{DV} be a portfolio consisting of a long hedged option C^H and short α amount of the underlying and η amount of short option C^{Rep}. The value of P^{DV} is therefore $C^H - (\alpha S + \eta C^{Rep})$. Crucially, the maturity of the hedged option T_H must be earlier than the maturity of the replicating option T_{Rep} $(T_H < T_{Rep})$.

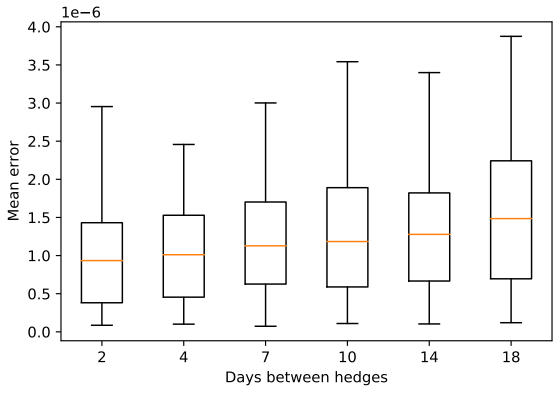
The next chapters are similar as to the ones made for the delta hedging. We analyze the delta-vega hedging strategy by examining how the squared mean error changes as we vary some variables.

3.1 Frequency analysis

As earlier, we first pick $T = 40$ ATM option to be hedged and we vary the hedging days in order to determine how the amount of days in between the hedges affects the error. The result is shown in figure 4(a).



(a) $T = 40$



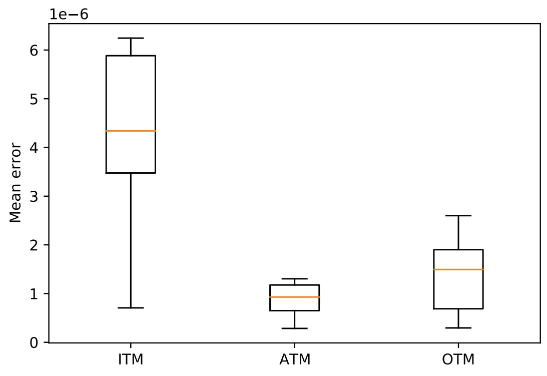
(b) Average over all T

Figure 4: Hedging error as function of days between hedges, ATM option.

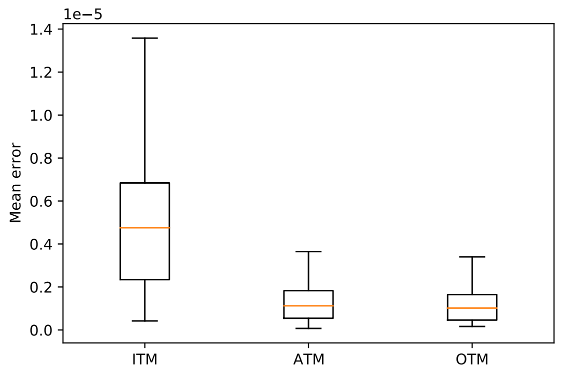
As we can see from the figure, the hedging error tends to increase when increasing the time between hedges. We saw a similar effect with plain delta hedging. We can repeat the same analysis but this time we consider all possible maturities and we average over them. The result can be seen in figure 4(b). Again, the less days between hedges the better the hedging performance.

3.2 Strike price analysis

Next, we will examine the affect different strike prices have on the delta-vega hedging error. We start the analysis by picking time to maturity $T = 40$ and days in between hedges $F = 2$. We then let E vary between -10, 0 and 10 units compared to the ATM strike price. This means that we are comparing similar ITM, ATM and OTM options. The plot is in figure 5(a). Again, we have similarly plotted the average over all maturities and hedging day intervals in 5(b).



(a) $T = 40, F = 2$



(b) Average over all

Figure 5: Error with different strike prices.

From the figures, we can clearly see that overall the ITM options perform the worst. For the specific case of $T = 40, F = 2$, ATM options are the best. When comparing the average of all maturities and hedging days, we can see that ATM and OTM options behave very similarly. However, by computing the mean of ATM and OTM errors, we note that OTM options are *slightly* better than ATM.

3.3 Combined analysis of delta-vega

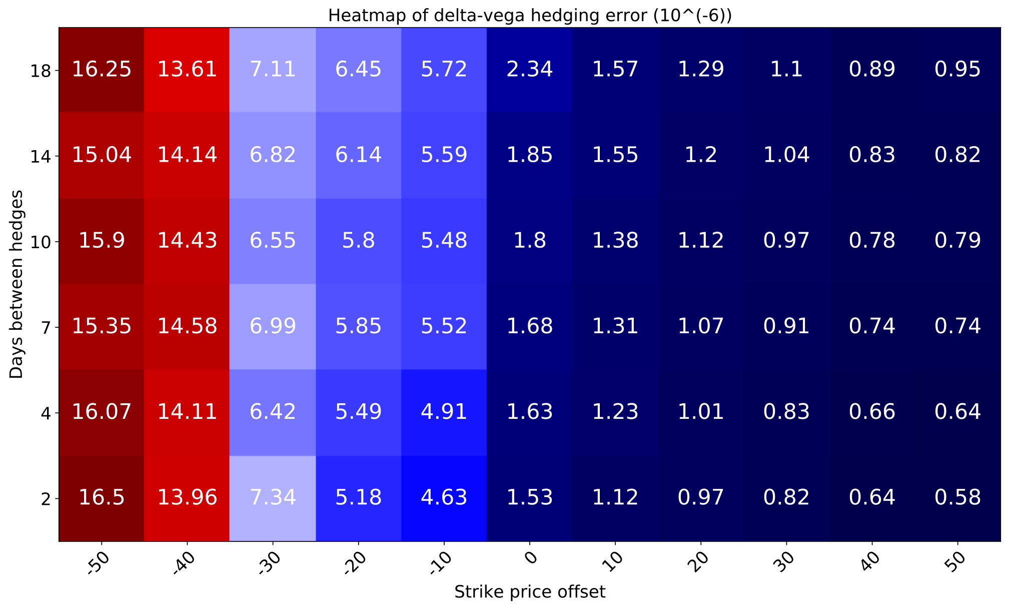


Figure 6: Delta-vega hedging error heatmap (scale 10^{-6}).

In figure 6 we have plotted the mean squared error of delta-vega hedging as a function of the strike price offset and the hedging days. Offset of 0 means we're hedging ATM options, negative offset means that we are hedging ITM options, and finally positive offset means we are hedging an OTM option.

Again we see a familiar structure, the hedging performance is better as we increase the price offset towards positive 50. Also, the less days between hedges, the better our performance is.

4 Strategy comparison

In this final section we'll examine how the delta and delta-vega hedging strategies compare in terms of the mean and standard error of MSE. We are particularly interested in how hedging ATM options differ.

4.1 Mean

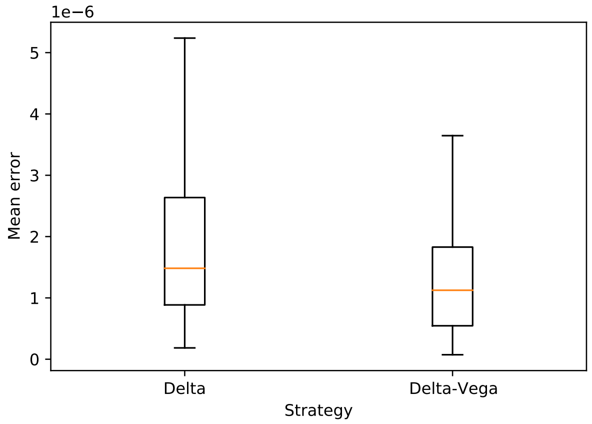
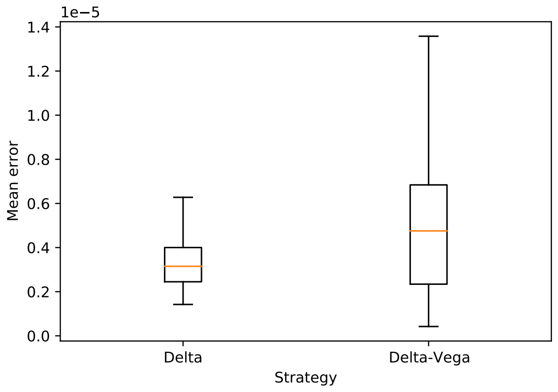
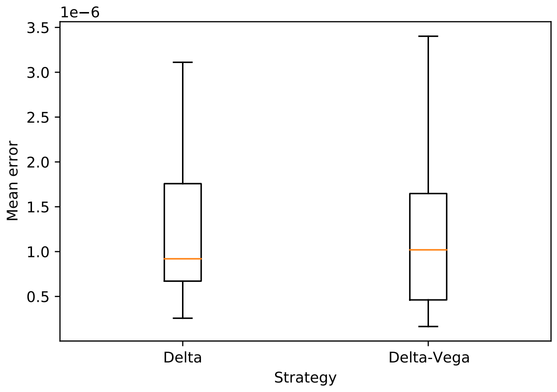


Figure 7: ATM Error comparison.

We have plotted the ATM option mean errors for the two strategies in figure 7. As we can see, the delta-vega strategy is better than the plain delta one. However, we do note that both strategies perform quite well and the difference is quite marginal. Next, we will compare ITM and OTM options, these are again options with offsets of -10 and 10 units respectively from the ATM call price. The results are in 8.



(a) ITM

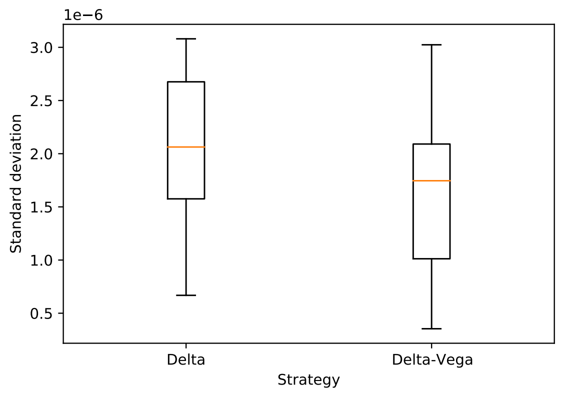


(b) OTM

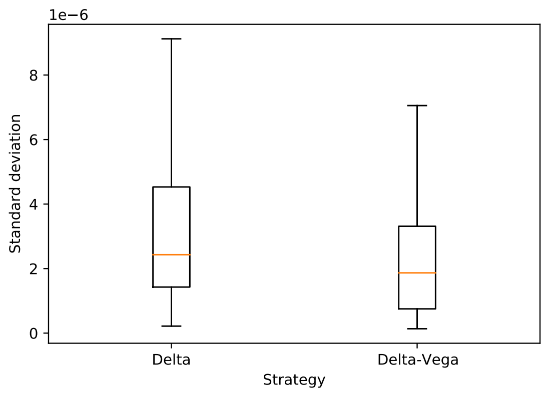
Figure 8: The ITM and OTM comparisons.

From figure 7 we can see that for ITM and OTM options, delta-vega hedging performs worse. For OTM options the delta-vega performance is quite similar to delta-hedging performance (figure 8(b)). However, for ITM options the performance of delta hedging is significantly better (note the different scale in the figure 8(a)).

4.2 Deviation



(a) $T = 40, F = 2$

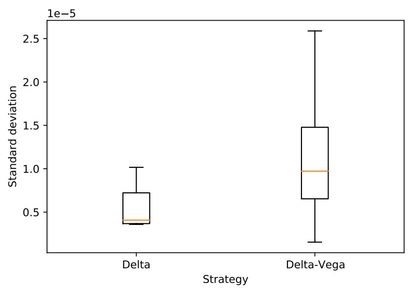


(b) Average

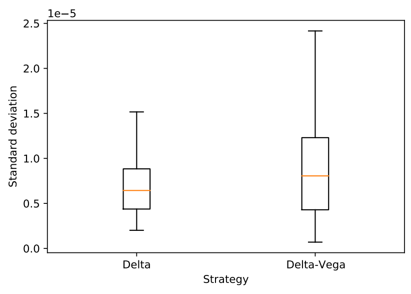
Figure 9: ATM option MSE standard deviations (scale 10^{-6}).

Finally, we'll compare the standard deviation of both strategies. We will first compare ATM options with $T = 40$ and $F = 2$. The results are in figure 9(a). Then similarly to other analyses, we again plot the average over all maturities T and days between hedges F. This plot is in figure 9(b).

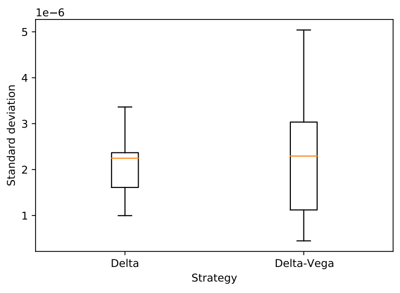
From the plots, we can clearly see that delta-vega hedging is more stable when it comes to ATM options. But what about ITM and OTM options? Next we will examine their standard deviations. We will consider options that have a strike price -10 or 10 units more than the ATM option strike. And again, we will consider the specific case of $T = 40, F = 2$ and the overall general case. The results are plotted in figure 10.



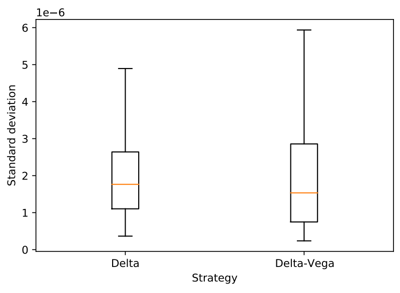
(a) $T = 40, F = 2$ ITM Option



(b) Average ITM Option



(c) $T = 40, F = 2$ OTM Option



(d) Average OTM Option

Figure 10: Standard deviations of both strategies.

From the plot 10 we can see that delta-hedging is better for ITM options, since its standard deviation is on average smaller than the deviation of delta-vega.

However for OTM options, the situation is reversed and the delta-vega strategy has smaller standard deviation on average. Although for the specific case ($T = 40, F = 2$), the standard deviations are very similar.

One other interesting observation is that the range of delta hedging standard deviation is smaller than the one for delta-vega.

5 Conclusions

We have now analyzed both delta and delta-vega hedging strategies very thoroughly. We have seen how they behave when the option price, time to maturity and hedging frequency are varied.

It is hard to effectively say which strategy is better but I think that at least for ATM options, delta-vega has some advantage.

6 Appendix: Code segments

Here are some code segments used to produce these analyses. I have not included everything, however all of the important parts are included.

6.1 Scaling and excel parsing

```
def transform_data(data):
    # Rename the columns
    column_len = len(data.columns.values)
    renamed_data = data[:]
    renamed_data.rename({
        data.columns.values[0]: "T",
        data.columns.values[column_len - 1]: 'date',
        data.columns.values[column_len - 2]: 'r',
        data.columns.values[column_len - 3]: 'S'
    }, inplace=True, axis="columns")
    # Transform into row form and drop NA rows:
    renamed_data = renamed_data.melt(id_vars=['T', 'date', 'r', 'S'], var_name="E")
    renamed_data = renamed_data.dropna()
    renamed_data.rename({'value':'Cobs'}, axis='columns', inplace=True)
    renamed_data['Cobs'] = renamed_data['Cobs'].replace(',','.', regex=True).astype(float)
    renamed_data['E'] = renamed_data['E'].astype(float)
    # Scale the time to maturity and risk-free rate. Convert to float
    scaled_data = renamed_data
    >> mutate(r = f.r/100, T = f.T/252, S = f.S/1000, E = f.E/1000, Cobs = f.Cobs/1000)

    # Calculate volatility and drop NA
    volatile_data = scaled_data.apply(lambda row: BSM(
        kind='call',
        S0=row['S'],
        K=row['E'],
        T= row['T'],
        r=row['r'],
        sigma=1.0
    ).implied_vol(value = row['Cobs']), axis = 1)
    scaled_data['volatility'] = volatile_data
    scaled_data = scaled_data.dropna()

    # Computing the Delta and vega values for each option at each point
    # in time using the implied volatilities
    deltas = scaled_data.apply(lambda row:
    BSM(kind='call',
        S0=row['S'],
        K=row['E'],
        T= row['T'],
        r=row['r'],
        sigma=row['volatility']
    ).delta(), axis=1)
    vegas = scaled_data.apply(lambda row:
    BSM(kind='call', S0=row['S'], K=row['E'], T= row['T'], r=row['r'], sigma=row['volatility'])
    .vega(), axis=1)
    scaled_data['delta'] = deltas
    scaled_data['vega'] = vegas
    scaled_data["timestamp"] = scaled_data.apply(lambda row: pd.to_datetime(
        row["date"], format="%d.%m.%Y"
    ).timestamp(), axis=1)

    # Sort by the time to maturity
    scaled_data = scaled_data.sort_values(by=["T"], ascending=False)

    # Again, dropping any NA rows
    scaled_data = scaled_data.dropna()
    return scaled_data
```

6.2 Delta hedging

```
# Delta hedging code:
def delta_hedge(total_data, day_index = 40, strike_price = 515/1000, day_delta = 2):
    # Default arguments:
    maturity = day_index/252

    # Get the data of the specified option for days until maturity:
    parsed_data = total_data[(total_data['T'] <= maturity) & (total_data['E'] == strike_price)]

    # If no data found for the specified strike or maturity, return NaN:
    if parsed_data.empty is True:
        return [np.nan, np.nan]

    # Setup the appropriate variables:
    errors, option_portfolios, replicating_portfolios, deltas = [], [], [], []

    # Form the first replicating and underlying option portfolios:
    current_option = parsed_data.iloc[0]
    current_delta = current_option['delta']
    current_price = current_option['S']
    current_replicating_port = current_delta * current_price
    current_option_port = current_option['Cobs']

    # First day t = 0:
    replicating_portfolios.append(current_replicating_port)
    option_portfolios.append(current_option_port)

    # The days until hedging:
    time_to_hedge = day_delta - 1

    i = 1
    # Each day until maturity
    for index, row in parsed_data.iloc[1:].iterrows():

        # Get the new option instance:
        current_option = row
        current_price = row['S']

        # Calculate the new replicating portfolio value:
        current_replicating_port = current_price * current_delta
        replicating_portfolios.append(current_replicating_port)

        # Calculate change of replicating portfolio
        replicating_value_change = (replicating_portfolios[i] - replicating_portfolios[i - 1])

        # Calculate new option portfolio value:
        current_option_port = current_option['Cobs']
        option_portfolios.append(current_option_port)
        option_value_change = (option_portfolios[i] - option_portfolios[i - 1])
        a = (option_value_change - replicating_value_change)
        errors.append(a)

        # Hedging:
```

```
        if time_to_hedge == 0:
            time_to_hedge = day_delta
            # Rehedge by setting the delta to a new value
            current_delta = current_option['delta']

            # Modify the previous replicating portfolio by taking the new delta into account:
            replicating_portfolios[i] = current_price * current_delta

        deltas.append(current_delta)
        time_to_hedge = time_to_hedge - 1
        i = i + 1

    # Return mean and std of A^2
    return (np.mean(np.square(errors)), np.nanstd(np.square(errors)))
```

6.3 Delta-Vega hedging

```
# Function for DV -hedging.
# Arguments: the dataframes of two different maturities and the common strike price.
def delta_vega_hedging(original_data,
    replicating_data, ttm, strike_price = 515/1000, day_delta = 2):
    # Calculate the first and last dates of the original dataset and ttm.
    last_date_original = original_data.iloc[original_data.shape[0] - 1]
    first_date_original = original_data[original_data["T"] <= ttm].iloc[0]

    # Restrict the replicating portfolio data to this range:
    replicating_parsed = replicating_data[(replicating_data["E"] == strike_price) &
    (replicating_data['timestamp'] >= first_date_original['timestamp'])
    & (replicating_data['timestamp'] <= last_date_original['timestamp'])]

    original_parsed = original_data[(original_data["T"] <= ttm)
    & (original_data["E"] == strike_price)]

    # The original call portfolio, replicating call portfolio and underlying stock portfolio:
    option_portfolios, replicating_portfolios, underlying_portfolios = [], [], []

    error_return = [np.nan, np.nan]
    if replicating_parsed.shape[0] < original_parsed.shape[0]:
        return error_return
    if original_parsed.empty:
        return error_return

    # The current option instances
    current_original = original_parsed.iloc[0]
    current_replicating = replicating_parsed.iloc[0]
    current_price = current_original["S"]

    # If not same pricepoint, error!
    if current_price != current_replicating["S"]:
        return error_return
    assert current_price == current_replicating["S"]

    # Calculate the first greeks
    current_option_delta = current_original["delta"]
    current_replicating_delta = current_replicating["delta"]
    current_option_vega = current_original["vega"]
    current_replicating_vega = current_replicating["vega"]

    # Store all computed greeks:
    option_deltas, replicating_deltas = [current_option_delta], [current_replicating_delta]
```

```
option_vegas, replicating_vegas = [current_option_vega], [current_replicating_vega]

# Compute ratios alpha and eta:
current_eta = current_option_vega / current_replicating_vega
current_alpha = current_option_delta - current_eta * current_replicating_delta

# Compute the first portfolios:
current_option_value = current_original["Cobs"]
current_underlying_value = current_alpha * current_price
current_replicating_value = current_eta * current_replicating["Cobs"]

# Store portfolios:
option_values = [current_option_value]
underlying_values = [current_underlying_value]
replicating_values = [current_replicating_value]
errors = []
time_to_hedge = day_delta - 1
i = 1

# Continue until end of original option data:
for index, row in original_parsed[1:].iterrows():
    # Get the current instance
    current_original = row
    current_price = current_original["S"]
    # Get the same from other sheet:
    current_replicating = replicating_parsed.iloc[i]

    # Calculate the new values and store new portfolios:
    current_option_value = current_original["Cobs"]
    option_values.append(current_option_value)
    current_underlying_value = current_alpha * current_price
    underlying_values.append(current_underlying_value)
    current_replicating_value = current_eta * current_replicating["Cobs"]
    replicating_values.append(current_replicating_value)

    # Calculate change:
    option_change = option_values[i] - option_values[i - 1]
    underlying_change = underlying_values[i] - underlying_values[i - 1]
    replicating_change = replicating_values[i] - replicating_values[i - 1]

    # Total change:
    a = option_change - (underlying_change + replicating_change)

    # Hedging:
    if time_to_hedge == 0:
        time_to_hedge = day_delta

        # Calculate the greeks delta and vega:
        current_option_delta = current_original["delta"]
        current_replicating_delta = current_replicating["delta"]
        current_option_vega = current_original["vega"]
        current_replicating_vega = current_replicating["vega"]

        # Compute ratios alpha and eta:
        current_eta = current_option_vega / current_replicating_vega
        current_alpha = current_option_delta - current_eta * current_replicating_delta

        # Compute the portfolios:
        current_underlying_value = current_alpha * current_price
        current_replicating_value = current_eta * current_replicating["Cobs"]
```

```
            # Modify the previous portfolios by taking the new delta into account:
            underlying_values[i] = current_underlying_value
            replicating_values[i] = current_replicating_value

        # Store the greeks used:
        option_deltas.append(current_option_delta)
        replicating_deltas.append(current_replicating_delta)
        option_vegas.append(current_option_vega)
        replicating_vegas.append(current_replicating_vega)

        time_to_hedge = time_to_hedge - 1
        i = i + 1
        errors.append(a)
    return (np.nanmean(np.square(errors)), np.nanstd(np.square(errors)))
```