FUTURES PROP. TRADING DEPT., CAPITAL SECURITIES CORP. SPECULATION OF NTD FOREIGN EXCHANGE

Yu-Ching Liao
Master of Science in Financial Engineering
University of Illinois at Urbana-Champaign
Illinois, 61801

July 28, 2023

ABSTRACT

This research delves into an intriguing aspect of the Taiwan foreign exchange market—the potential to speculate the 3pm price of TPFT, the USD/NTD price at Taipei Forex Inc. (TPEFX), following the standard 12:00 to 14:00 (GMT+8) trading halt. Given the CMPN prices traded on the New York Stock Exchange will continue the trade between 12:00 to 14:00, we are to utilize the price of CMPN to speculate the price path that halting TPFT should have.

The previous model utilized Exponential Moving Average (EMA) value of CMPN price to fill the trading halt between 12:00 to 14:00, which provides the 0.017060 for the accuracy calculated by the absolute value of the difference between the estimated value at 12:15 and 15:00.

Our model can be divided into four parts: 1) Fitting 9:00 to 12:00 CMPN Price on TPFT Price to train the model, 2) Get the estimation value of halting TPFT based on CMPN prices 3) Denoise the estimated price path and 4) Shift the estimated price path based on the TPFT spot price at 12 and estimation price at the same moment. For the first step, we utilize Simple Linear Regression to fit the CMPN price from 9:00 to 12:00 on TPFT price in the same time period to train our model. After so doing, we can then predict the price path by implementing new input of CMPN on our trained model as the second step. For the third step, we denoise the estimated price path. In here, we tested two different denoising methods: Wavelet Signal Denoiser and Exponential Moving Average. We will discuss the utilization of both two methods later in the following sections. And for the last step, we shift the estimated path based on the actual spot price at 12 of TPFT and estimated spot price at 12, so that, at 12, our estimated path can have same initilization with actual 12 spot price.

The findings reveal that the 13:15 estimated price applying a linear regression model, denoised via the EMA model, performs the best, reducing the error between the actual 15:00 price and estimated price down to only 0.014977 per day. This outcome not only informs our understanding of effective price speculation in intra-day trading suspensions but also contributes to practical investment strategies in the Taiwan foreign exchange market. This research could potentially serve as a valuable tool for investors navigating the complexities of the Taiwan foreign exchange market.

Keywords Machine Learning · Linear Regression · Exponential Moving Average (EMA) · Wavelet Signal Denoiser · Foreign Exchange Rate · TPEFX

1 Problem Formulation

This research targets a significant challenge in the Taiwan foreign exchange market. Specifically, we aim to address the difficulty in speculating the 3pm price of TPFT, the USD/NTD price at Taipei Forex Inc. (TPEFX), during the standard 12:00 to 14:00 (GMT+8) trading halt.

To solve this problem, we exploit the continuous trading of CMPN prices on the New York Stock Exchange during the same halt period. However, we recognize that straightforwardly applying the price movement of CMPN to the TPFT could yield inaccurate results due to the potential volatility and noise during the halt period. Therefore, we are tasked with creating a model that accurately applies the CMPN price data to predict the TPFT price path during the halt period.

The problem comprises several layers. First, we need to establish an effective model for predicting the halted TPFT price based on the CMPN price. Second, this model needs to be able to denoise the estimated price path to obtain a more reliable prediction. Furthermore, the model must consider the spot price at 12:00 to ensure that our estimated price path is aligned with the actual spot price.

Historically, Exponential Moving Average (EMA) values of CMPN price have been used to estimate the trading halt price with an accuracy of 0.017060, calculated by the absolute value of the difference between the estimated value at 12:15 and 15:00. Our goal is to develop a model that improves upon this accuracy, providing investors with a more reliable tool for navigating intra-day trading suspensions in the Taiwan foreign exchange market.

The problem, therefore, involves both theoretical and practical aspects. The theoretical side requires developing a robust model for predicting TPFT prices during trading halts based on CMPN prices. From a practical perspective, the solution must contribute to effective investment strategies by providing more accurate price predictions, thereby helping investors make more informed decisions in the Taiwan foreign exchange market.

2 The data of experiment

The experiment utilized tick record data of prices provided by TPEFX and NYSE, and the estimated price that previous model generate ranged from December 1st 2022 to May 26th 2023.

2.1 Description of data

We are provided with 8 txt files containing the BEST_BID, BEST_ASK and TRADE price records, which are seperated as following:

Table 1: Dataset Overview and Description.

$TPFT$_1201_0526.txt	The price of NTD traded in TPEFX from 2022/12/01 to 2023/05/26
$CMPN$_0101_0131.txt	The price of NTD traded in NYSE from 2023/01/01 to 2023/01/31
$CMPN$_0201_0228.txt	The price of NTD traded in NYSE from 2023/02/01 to 2023/02/28
$CMPN$_0301_0331.txt	The price of NTD traded in NYSE from 2023/03/01 to 2023/03/31
$CMPN$_0401_0430.txt	The price of NTD traded in NYSE from 2023/04/01 to 2023/04/30
$CMPN$_0501_0526.txt	The price of NTD traded in NYSE from 2023/05/01 to 2023/05/26
$CMPN$_1201_1231.txt	The price of NTD traded in NYSE from 2022/12/01 to 2022/12/31
$BNTD$_1201_0526.txt	The previous estimated price of NTD traded in TPFT from 2022/12/01 to 2023/05/26

2.2 Data pre-processing

Since some of the original data from same institution is not stored in one txt file due to the large traded volume, we first import them with separating them by the institution provided. Besides, for convenience of model fitting process, we transform the tick record to second-interval record; the missing value will be filled with the stale quote. Lastly, since we are provided with BEST_BID, BEST_ASK and TRADE price records, we are to calculate the MID price of each second as our feature. Below we provide the original dataset overview and the converted dataset overview.

Table 2: Original data-set overview ($CMPN$_0101_0131$.txt$).

Date	Type	Price
2023/1/3 AM 09:01:20	BEST_BID	30.718
2023/1/3 AM 09:01:20	BEST_ASK	30.753
2023/1/3 AM 09:01:20	TRADE	30.735
⋮	⋮	⋮
2023/1/13 PM 03:59:59	TRADE	30.282
2023/1/13 PM 03:59:59	BEST_BID	30.269
2023/1/13 PM 03:59:59	TRADE	30.282

We then convert and transform those separated CMPN data into one dataframe with the format shown below:

Table 3: Converted data-set overview (df_CMPN).

Date	BID	MID	ASK	TRADE
2022-12-01 09:01:22	30.772	30.79050	30.8090	30.7910
2022-12-01 09:01:23	30.772	30.79050	30.8090	30.7910
2022-12-01 09:01:24	30.772	30.79050	30.8090	30.7910
⋮	⋮	⋮	⋮	⋮
2023-05-26 15:56:47	30.714	30.72225	30.7305	30.7220
2023-05-26 15:56:48	30.714	30.72325	30.7325	30.7235
2023-05-26 15:56:49	30.714	30.72325	30.7325	30.7235

We can discover that now the index 'Date' is the unique every second between the first trade moment and the last trade moment, and since the missing value is filled with the stale quote, some prices will remain the same as previous quote.

As shown below, we can see there is a trading halt in TPEFX between 12:00 pm and 14:00 pm:

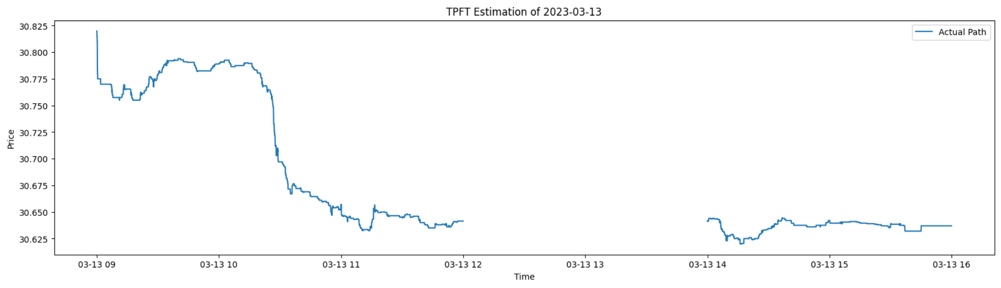

Figure 1: NTD Foreign Exchange in TPEFX.

We are to utilize the price from NYSE to speculate the price path that TPEFX should have during the trading halt period:

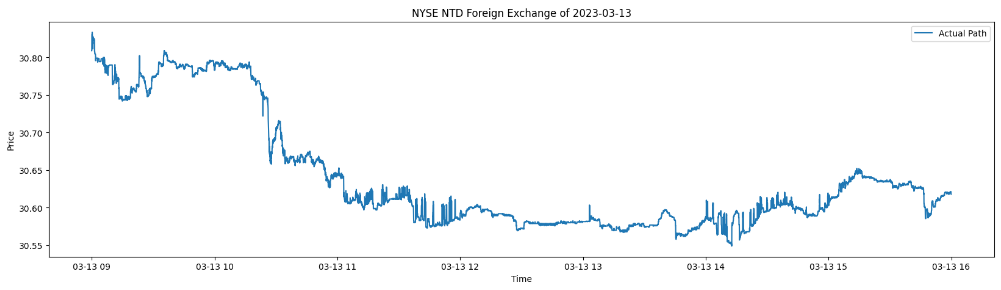

Figure 2: NTD Foreign Exchange in NYSE.

3 Methods Review

3.1 Linear Regression

Linear regression is a straightforward and widely used statistical method that is prevalent in various fields. It models the relationship between two variables by fitting a linear equation to the observed data. The simplicity of linear regression, coupled with its power to effectively estimate dependent variables, makes it a valuable tool for forecasting and trend analysis.

The fundamental assumption behind the linear regression model is that there exists a linear relationship between the dependent and independent variables. Given a data set of n points, the equation for linear regression is

$$y = \beta_0 + \beta_1 x + \epsilon \tag{1}$$

where y is the dependent variable, x is the independent variable, β_0 and β_1 are parameters of the model which represent the intercept and slope of the line, respectively, and ϵ is the error term.

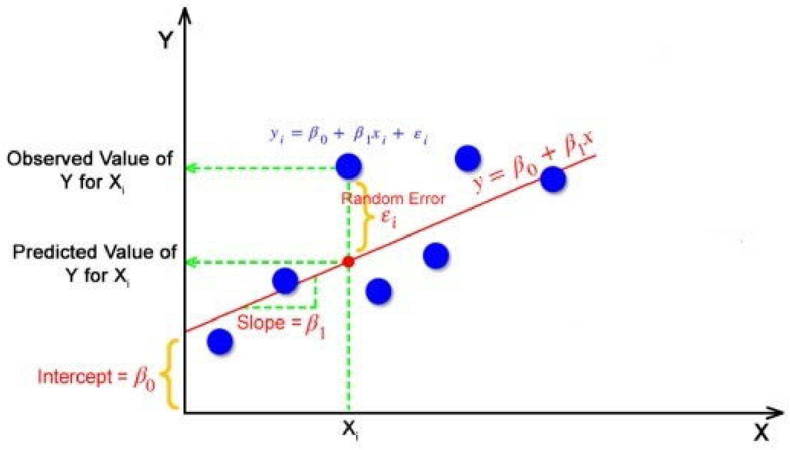
Y
$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i$
Observed Value of Y for X_i
$y = \beta_0 + \beta_1 x$
Random Error ε_i
Predicted Value of Y for X_i
Slope = β_1
Intercept = β_0
X_i
X

Figure 3: A simple linear regression model.

3.2 Exponential Moving Average

The exponential moving average (EMA) is a specific type of moving average that gives more weight to recent prices, making it more responsive to new information. It's particularly effective in markets which experience frequent volatility as it reacts more significantly to recent price changes.

The EMA is calculated using the following formula:

$$EMA = (C - EMA_{previous}) \times K + EMA_{previous} \tag{2}$$

where C is the current price, $EMA_{previous}$ is the EMA from the previous period, and K is the smoothing constant. The smoothing constant is typically calculated as $2/(n+1)$, where n is the number of periods.

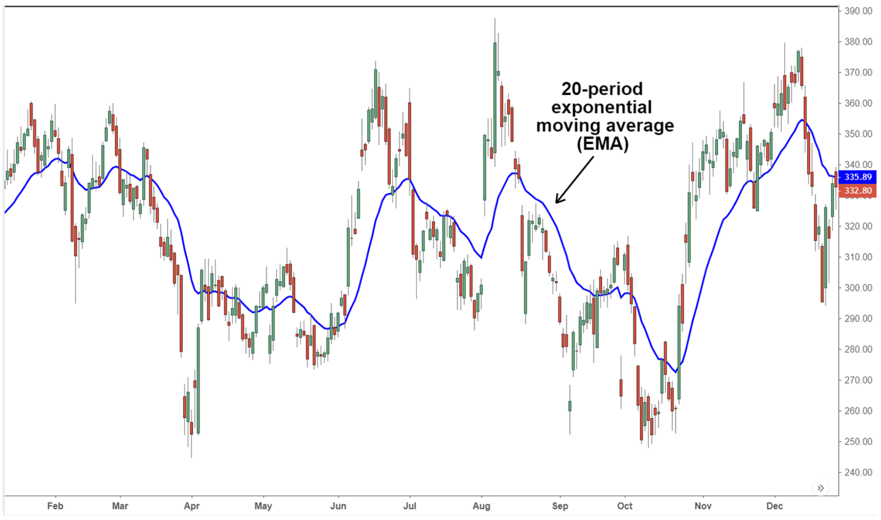

Figure 4: Exponential Moving Average on a price chart.

3.3 Wavelet Signal Denoiser

Wavelet signal denoising, or wavelet thresholding, is a method used to remove noise from a data signal. This is achieved by performing a wavelet decomposition on the signal, setting some of the smaller wavelet coefficients (assumed to be noise) to zero, and then reconstructing the signal. The aim is to eliminate the random noise fluctuations while preserving the true signal components.

The wavelet signal denoising process is composed of three main steps:

1. Decompose: This step involves the application of the wavelet transform to the original noisy signal to obtain the wavelet coefficients. The wavelet transform is a mathematical function used to break down a signal into its different frequency components. It uses a set of functions, called wavelets, to represent data or functions. Unlike Fourier transforms, which only provide the frequency information, wavelet transform provides both frequency and location (time) information, making it especially suited for non-stationary signals. Given a data signal $x(t)$, its continuous wavelet transform (CWT) is defined as

$$CWT_x(a,b) = \frac{1}{\sqrt{|a|}} \int_{-\infty}^{+\infty} x(t)\psi\left(\frac{t-b}{a}\right) dt \tag{3}$$

where a and b are the scale and translation parameters, and ψ is the wavelet mother function.

2. Threshold: Apply a threshold to the wavelet coefficients. Coefficients with an absolute value less than the threshold are set to zero, while others are kept intact. Various thresholding strategies such as hard and soft thresholding can be used.

3. Reconstruct: The signal is then reconstructed from the modified wavelet coefficients, resulting in a denoised version of the original signal.

This denoising approach is especially useful when the signal of interest is thought to have a sparse representation in some wavelet basis, and the noise is assumed to be Gaussian and independent. The quality of denoising depends on the choice of the wavelet, the number of decomposition levels, the threshold selection rule, and the thresholding strategy.

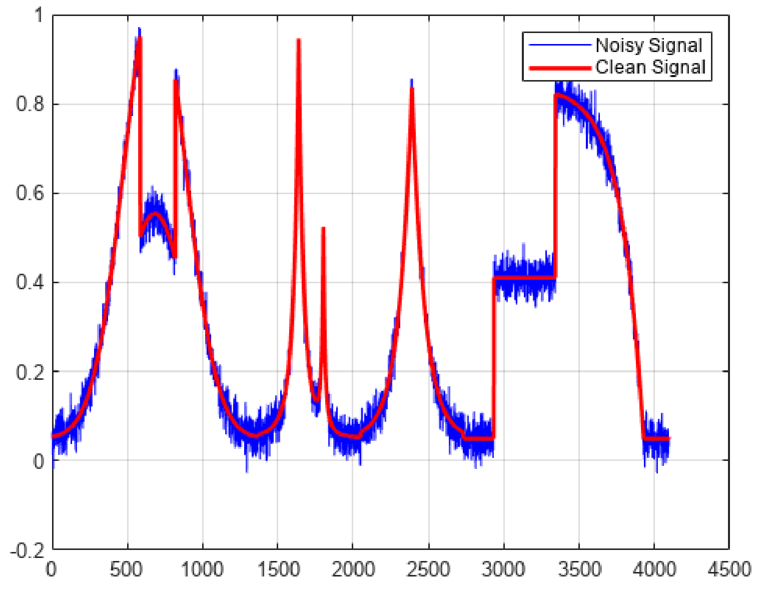

Figure 5: Wavelet denoising on a signal.

This is achievable by the following code:

```python
def wavelet_denoise(data, wavelet, level):
    # Remember the original data length
    original_length = len(data)

    # Ensure the data length is even
    if len(data) % 2:
        data = np.pad(data, (0, 1), 'edge')

    # Decompose to get wavelet coefficients
    coeff = pywt.wavedec(data, wavelet, mode="per", level=level)

    # Calculate sigma for threshold as defined in
    # Donoho and Johnstone 1994, p. 12
    sigma = (1/0.6745) * np.median(np.abs(coeff[-level] - np.median(coeff[-level])
        ))

    # Calculate universal threshold
    uthresh = sigma * np.sqrt(2*np.log(len(data)))
    coeff = list(map(lambda x: pywt.threshold(x, value=uthresh, mode="soft"),
        coeff))

    # Reconstruct the signal using the thresholded coefficients
    denoised_data = pywt.waverec(coeff, wavelet, mode="per")

    # Truncate back to the original length if necessary
    if original_length != len(denoised_data):
        denoised_data = denoised_data[:original_length]

    return denoised_data
```

Listing 1: Wavelet Signal Denoiser

4 Design of experiments

We will show how we design the experiments in this section. As the flow chart below, the main process of this experiment can be divided into three parts based on different timestamp index of each new input. Before 12:00, we collect the CMPN and TPFT prices as the dateset of the following process. At the spot 12:00, we trained the Linear Regression model to get the α and β we need. Since we only have one feature, we will not utilize other method, such as Ridge Regression, Lasso Regression or Deep Learning methodology. And after 12:00, we process our model to get the estimation path of each moment.

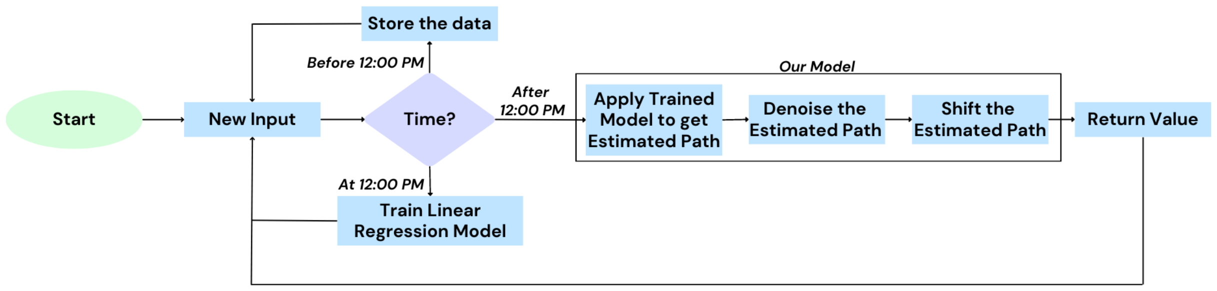

Figure 6: Flow Chart of the experiment.

Our model can be separated into three parts as the flow chart below: 1) Fitting 2) Denoising and 3) Shifting. For the first part, we fit the trained Linear Regression model on new CMPN model to get the estimation of TPFT. After so, we apply the denoising model. In here, we try EMA and Wavelet and will test their performance separtely. And lastly, we shift the estimated path between 12:00 to 14:00 based on the actual closing price at 12 and estimation closing point at 12, so that our generated path can have same initial condition at 12.

4.1 Before 12:00: Collection of the data

Before 12:00 of everyday, we store as much as date we can. In our experiment, we imitating that we collect the price as the input data for each second. We then first setup the in-need variables we need for following experiment.

```
df = pd.DataFrame({"TPFT": df_TPFT["MID"],
            "CMPN": df_CMPN["MID"]})
chosen_date = datetime.date(2023, 3, 13)
X_record = np.array([])
y_record = np.array([])
```

Listing 2: Variables Setup

We, in here, define *df* as the mid price of TPFT and CMPN, and we choose the data of March, 13th, 2023 as the data of our demonstration. We, as well, define two *numpy* arrays, X_record and y_record to store the new input each second.

After the definition of the variables, we can then move on to the imitation of real input of the data. This imitation, as shown below, can be done with simple for loop.

```
for date_idx in df[(df.index.date == chosen_date) & (df.index.time <= datetime.
    time(14, 0))].index:
    TPFT_new_input = df[df.index == date_idx]["TPFT"].values[0]
    CMPN_new_input = df[df.index == date_idx]["CMPN"].values[0]
    X_record = np.append(X_record, [CMPN_new_input])

    if date_idx.time() <= datetime.time(12, 0): # Storing all training sets
        y_record = np.append(y_record, [TPFT_new_input])
```

Listing 3: Imitation of real-time environment and Storing the Inputs

Note that, for CMPN, we continue storing their prices even after 12:00 since we still need the input of feature to get the estimation of TPFT after then. The only ceased input will then be that of TPFT.

4.2 At 12:00: Training of the Linear Regression model

At 12:00, TPEFX will halt their trading, thus no prices updated. We then train our model with the CMPN and TPFT we collected from last step. In here, we have to reshape the X_record for the prerequisite of utilizing the $Scikit - learnLinearRegression$ model, which we have to reshape back afterward for storing the new input. We also store the actual closing price at 12 as $real\_closing$, the fitted price path as y_pred, and the index of the input at 12:00 as $index\_of\_12$ for following steps.

```
    if date_idx.time() == datetime.time(12, 0): # Fit and get Parameters
        model = LinearRegression()
        X_record = X_record.reshape(-1, 1)
        model.fit(X_record, y_record)
        y_pred = model.predict(X_record)
        X_record = X_record.reshape(-1)

        real_closing = TPFT_new_input
        index_of_12 = len(X_record) - 1
```

Listing 4: Training the Linear Regression model

4.3 After 12:00 to 14:00: Speculation of TPFT

Between 12:00 to 14:00 we process our model to get the estimated price. As I have discussed above, our model can be divided into three parts: 1) Fitting 2) Denoising and 3) Shifting. For the fitting part, we simply apply the trained model on new input to get the estimated prices. And the third part, which as well is simple, is to shift the whole estimated path to match the $real\_closing$ and the 12:00 spot of estimated path, so that, from 12:00, our estimated path can have the initial start point that is identical to 12:00 actual spot.

For the second part, we try two different ways to achieve the denoising (smoothing), which is EMA and Wavelet Signal Denoise. We have discussed both EMA and Wavelet Filtering in the previous section, as well the code of implementing the Wavelet Signal Denoiser. Thus we will directly look into the code of implementing these two methods.

```
    if (date_idx.time() > datetime.time(12, 0)) & (date_idx.time() <= datetime.
       time(14, 0)):
        # Fit, Denoise and Shift
        y_pred = np.append(y_pred, model.predict([[CMPN_new_input]])) # Fit

        # Convert the numpy array to pandas Series
        y_pred_series = pd.Series(y_pred)

        # Calculate the Exponential Moving Average
        smoothed_y_pred = y_pred_series.ewm(span=100, adjust=False).mean().values

        est_closing = smoothed_y_pred[index_of_12]
        smoothed_y_pred = smoothed_y_pred - (est_closing - real_closing) # Shift
```

Listing 5: Estimation using Linear Regression, EMA and Shifting

```
    if (date_idx.time() > datetime.time(12, 0)) & (date_idx.time() <= datetime.
       time(14, 0)):
        # Fit, Denoise and Shift
        y_pred = np.append(y_pred, model.predict([[CMPN_new_input]])) # Fit
        smoothed_y_pred = wavelet_denoise(y_pred, wavelet="db8", level=5) #
            Denoise
        est_closing = smoothed_y_pred[index_of_12]
        smoothed_y_pred = smoothed_y_pred - (est_closing - real_closing) # Shift
```

Listing 6: Estimation using Linear Regression, Wavelet Signal Denoiser and Shifting

We apply these three steps on all the new input after 12:00, so that we can get the estimation path as following.

In here, for applying on the real-time trading situation, we as well calculate the time spent of both models, only to make sure this methodology is capable with handling the real-time tick data.

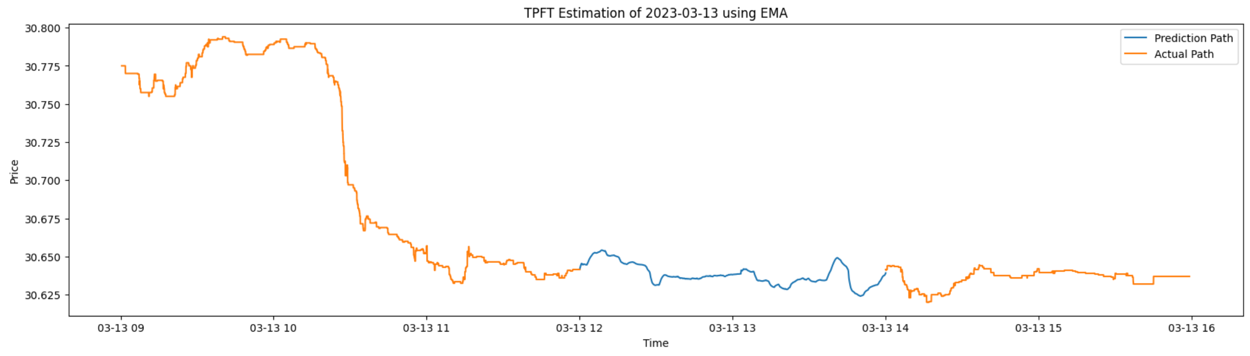

Figure 7: TPFT Estimation path with EMA Denoising.

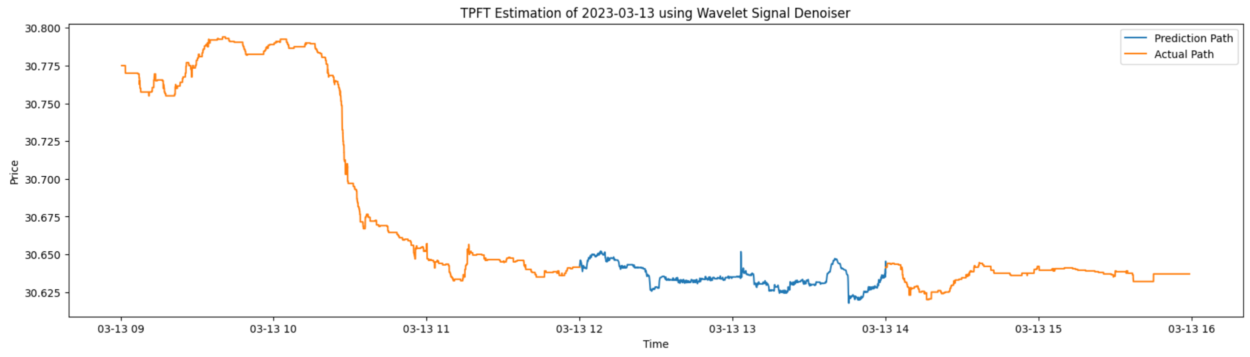

Figure 8: TPFT Estimation path with Wavelet Signal Denoising.

We can see that, for each new input, most of the time spent of the calculation are ranged from 0.015 to 0.035 second, which showing the high efficiency and capability of handling the high-frequency price input data.

5 Numerical Results

We compare and evaluate the performance of two models utilizing different denoising methods, which are EMA and Wavelet Signal Denoiser, by their performance in predicting the 15:00 TPFT price. We as well tested on different timestamp every 15 minutes between 12:00 and 14:00, to see in which timestamp, our models can give out the most accurate prediction of 15:00 TPFT price. The performance of each model was measured based on the Absolute value of the difference between Actual TPFT 15:00 price each day and the estimated price of every 15 minutes between 12:00 and 14:00. The results of this study suggest that the estimation at 13:15 using EMA as the denoiser can provide the best performance in predicting the 15:00 TPFT price. However, the selection of the optimal technique highly depends on the type of data and the application domain. We will as well compare to previously used model that simply applying EMA on CMPN price as the estimation, too see if our model can give out the better results.

The testing results shown as below:

Table 4: Numerical Results (smaller is better).

Time	Linear Regr., EMA and Shift	Linear Regr., Wavelet Denoiser and Shift	EMA Smoothing
12:15	0.015785	0.017466	0.017060
12:30	0.016061	0.017492	0.017721
12:45	0.015523	0.016999	0.017410
13:00	0.015296	0.017000	0.017457
13:15	0.014977	0.016760	0.017297
13:30	0.015118	0.016599	0.017819
13:45	0.015109	0.016712	0.017319

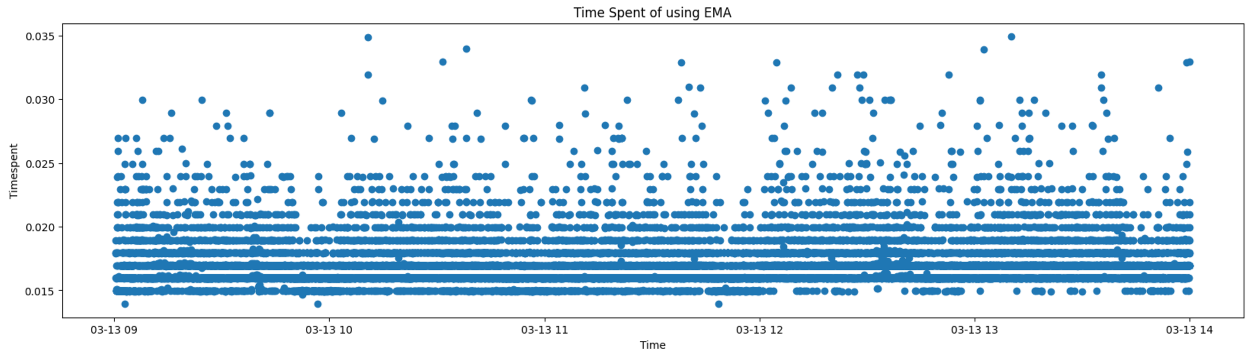

Figure 9: TPFT Estimation time of each new input with EMA Denoising.

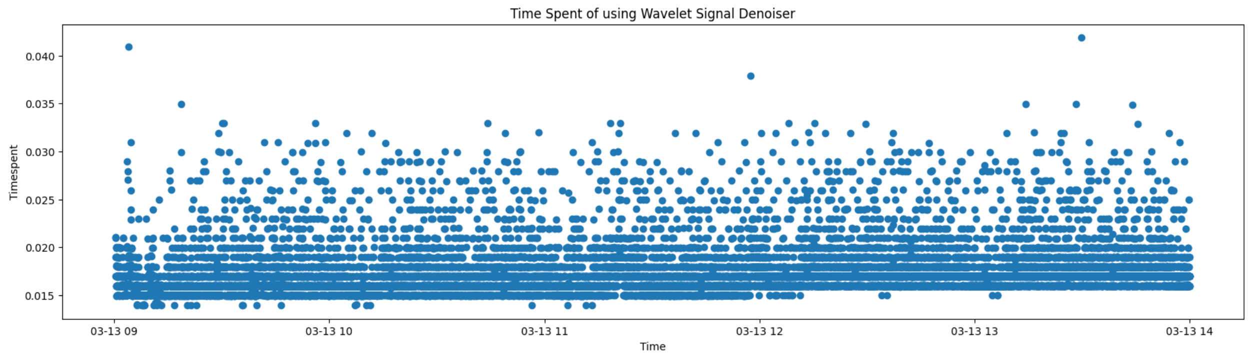

Figure 10: TPFT Estimation time of each new input with Wavelet Signal Denoising.

6 Conclusions

In conclusion, in most of the timestamp, both of our model can generate the better result compared to simply taking EMA value of the CMPN price. We compared the results of these three different models based on the absolute difference between each estimation of each timestamp and actual 15:00 price. Our results showed that the best estimation, in average, is the one at 13:15 that generated by the model implementing Linear Regression, EMA and Shifting, which can reduce the average difference of all 133 tested days down to 0.014977 per day. However, since we are using the average difference everyday, it is not necessary that our model will perform the best in all the timestamp in the real-time. Besides, the result, though there is nuance, is not significantly different from model to another. The better way, therefore, is to compare three different estimations that generate from three different models and generate the conclusion based on these three different methods. This result, however, can be the standard for further research to improve the performance and accuracy of the TPFT speculation technique.