RELATIONSHIP BETWEEN US/CANADA EXCHANGE RATE AND COMMODITY PRICES

ECONOMETRICS II: PROJECT II

NANA OSEI SARPONG
11371873

Contents

PART I

INTRODUCTION

Commodity currency is floating currency that exhibit co-movement with world prices of primary commodities due to a country's heavy dependence on commodity exports (Chen et al., 2010). Canada as a country boasts of a wide range of natural resources and depends on revenue from the exploitation and export of such resources. According to Statistics Canada, export commodities such as energy products, metal and non-metallic mineral products, metal ores and non-metallic minerals, forestry products, building and packaging materials, farm, fishing and intermediate food products contributed $ 443.5 billion in 2022, accounting for 56.92% of total exports. The large dependence on such well traded commodities on the world market characterises the Canadian economy making it feasible to regard the Canadian dollar as a commodity currency. In this paper, we set out to explore the empirical relationship between in Canadian exchange rate and commodity prices, particularly looking at total commodity prices, energy prices and non-energy commodity prices due to the importance of oil to the Canadian economy.

THEORETICAL FRAMEWORK

Chen et al.(2010) discuss the present value approach which posits a relationship between exchange rates and commodity prices.

$$s_t = \gamma \sum_{j}^{\infty} \varphi^j E_t(f_{t+j}|I_t)$$

Similar to mainstream finance and macro theoretical models, the paper posits that there is a relationship between the exchange rate (s), and the fundamentals in an economy (f) such that the exchange rate equals the discounted sum of the expected value of future fundamentals conditional on all information available at time t.

This thereby suggests there may be an underpinning relationship such that forecasts could be made about one variable based on the value of another. This suggests that one variable is a forerunner of the other, implying Granger causality. In our context, the fundamental of interest is the commodity price and this paper shall particularly examine how total commodity prices, energy prices and non-energy commodity prices relate to the US/Canada exchange rate.

Reverse causality or endogenous responses, however, may make it more difficult to see exchange rate Granger-causing basic movements when fundamentals are not fully exogenous. Exchange rate movements, for example, can seem to precede changes in interest rates or the money supply, although these correlations might be the product of underlying mechanisms or policy reactions. As a result, positive Granger-causality results between standard fundamentals and currency rates need to be read with caution, particularly if the fundamentals in question are not obviously exogenous to changes in exchange rates.

METHODOLOGY

In this study, we will employ various techniques to investigate the relationship between the Canadian exchange rate and commodity prices. Our approach will include descriptive statistics analysis, correlation analysis, unit root testing using the augmented Dickey-Fuller (ADF) test, cointegration testing with the Engel-Granger test, regression analysis using the Vector Autoregressive and Vector Error Correction Models, stability testing and Granger causality testing as well as some diagnostic tests. Through these analytical techniques, we aim to comprehensively understand the empirical dynamics between the exchange rate and commodity prices in Canada.

DATA DESCRIPTION

This study makes use of exchange rate and commodity price data obtained from Statistics Canada and US CPI obtained from the US Bureau of Labor Statistics. Contained in our data set are four monthly time series variables with 544 observations from January 1972 to April 2017. The exchange rate is measured as the United States dollar to the Canadian dollar, noon spot rate average, price indices are Fisher commodity price indices in terms of US dollars and the CPI is the US CPI, all urban consumers current series. Below is a table containing descriptive statistics of the log of exchange rate and the log of real price indices deflated using the US CPI.

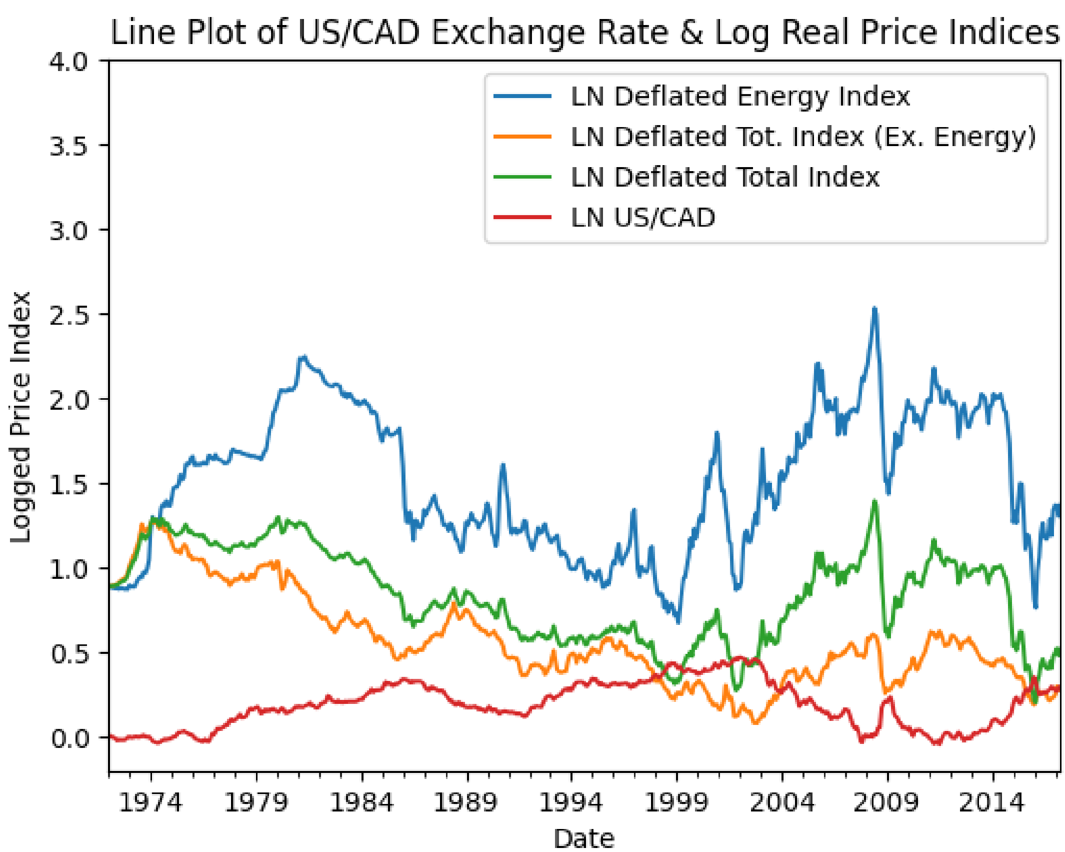
Table 1: Descriptive Statistics

Variable	**Minimum**	**Median**	**Maximum**	**Mean**	**Standard deviation**
LN Exchange rate (US/CAD)	-0.0457	0.1862	0.4702	0.1908	0.1362
LN Total Commodity Price Index	0.2045	0.8485	1.3932	0.8398	0.2701
LN Total Commodity Price Index (Excluding Energy)	0.0800	0.5111	1.2832	0.5704	0.2774
LN Energy Index	0.6726	1.5122	2.5332	1.5111	0.4143

Compared to most of the other variables, the log of the exchange rate has the least volatility with a standard deviation of a mere 0.1362 around an average of 0.1908. The log of the energy price index has both the highest maximum value, the largest mean, and the largest standard deviation among all three price indices.

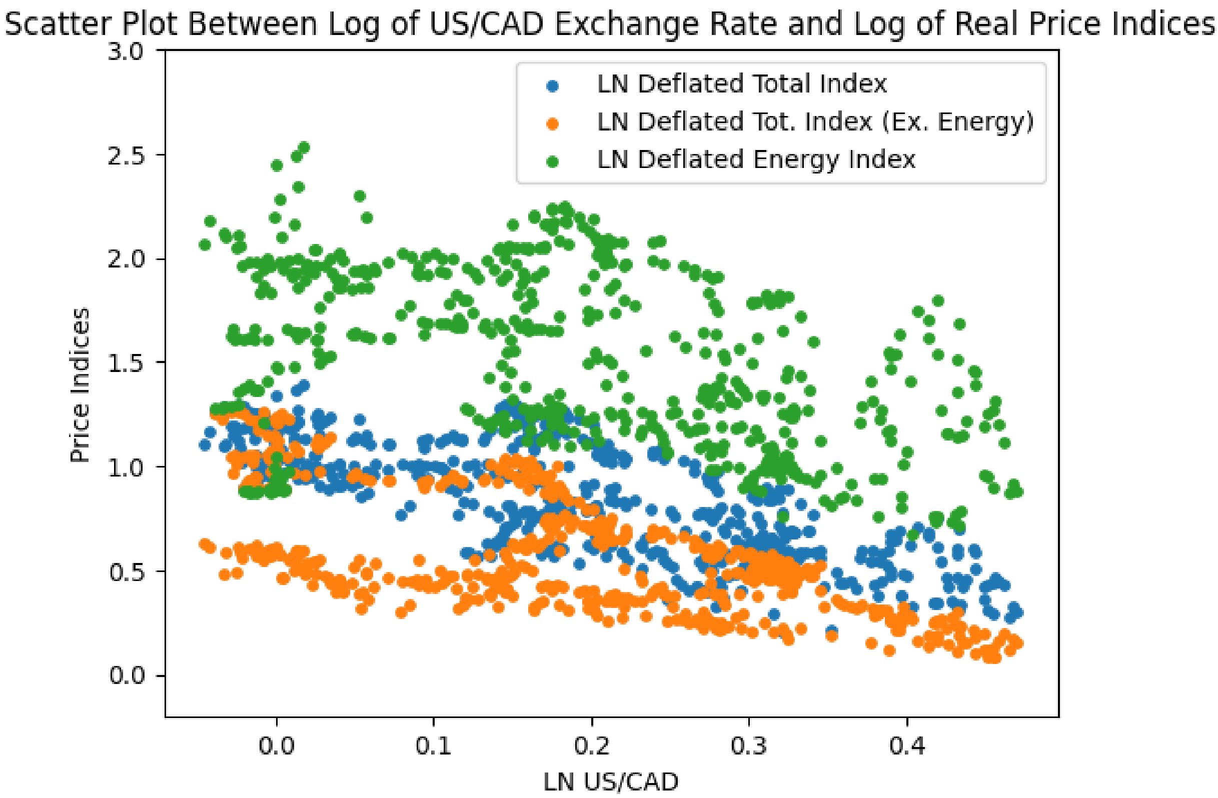
TREND ANALYSIS

A look at how the variables co-move over time reveals some interesting insights. From 1972 to 2003, price indices were falling as the exchange rate rose. An exception will be the energy price index which was initially rising alongside the exchange rates before finally taking a dive at 1982. The initial increases in the energy index is likely caused by the 1973 oil crisis which caused world oil prices to rise sharply due to an embargo by the Organization of Arab Petroleum Exporting countries against countries that supported Israel during the Fourth Arab-Israeli War. There is however a general divergence in the time plots in our graph afterwards as the exchange rate falls and all price indices rise and vice versa with some deviations along the line.



CORRELATION ANALYSIS

A relationship between the log of the exchange rate can easily be explored by looking at scatter plots and the correlation between the log of the exchange rate and the various logged real price indices.



The scatter plots show a weak negative relationship between the log of US/Canada exchange and the log of real price indices.

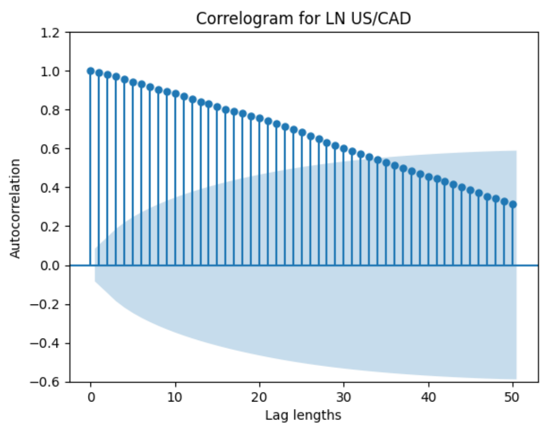
Table 2: Pearson correlation coefficients between log US/Canada exchange rates and the logged real price indices

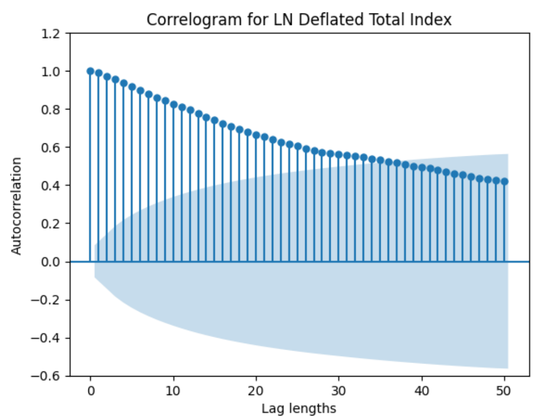
Index	Correlation coefficient	P-value
Total Commodity Price Index	-0.7553	0.0000
Total Commodity Price Index (Excluding Energy)	-0.6477	0.0000
Energy Index	-0.4233	0.0000

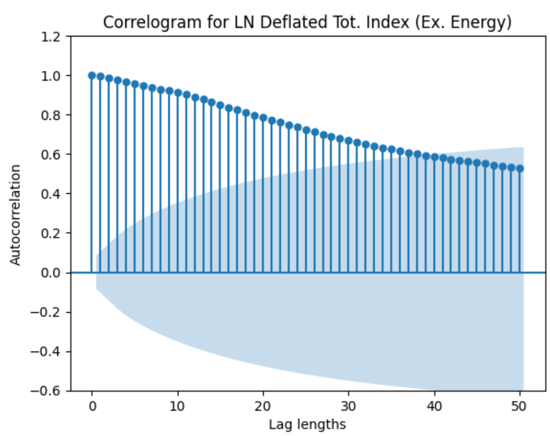
All indices show a negative relationship which appears statistically significant at the 5% level but these correlations are suspect since the presence of a unit root in such a time series will render them useless.

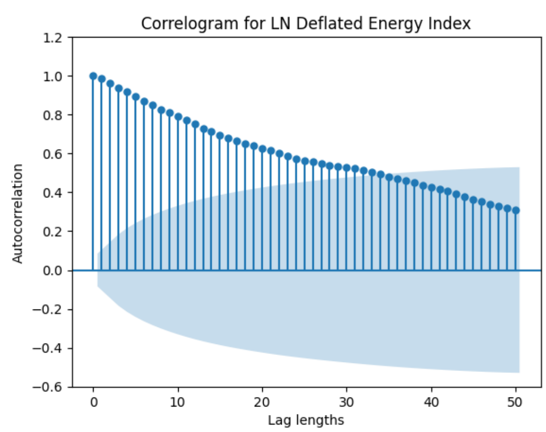
UNIT ROOT TEST

The following graphs display the respective autocorrelation functions shown with a correlogram.







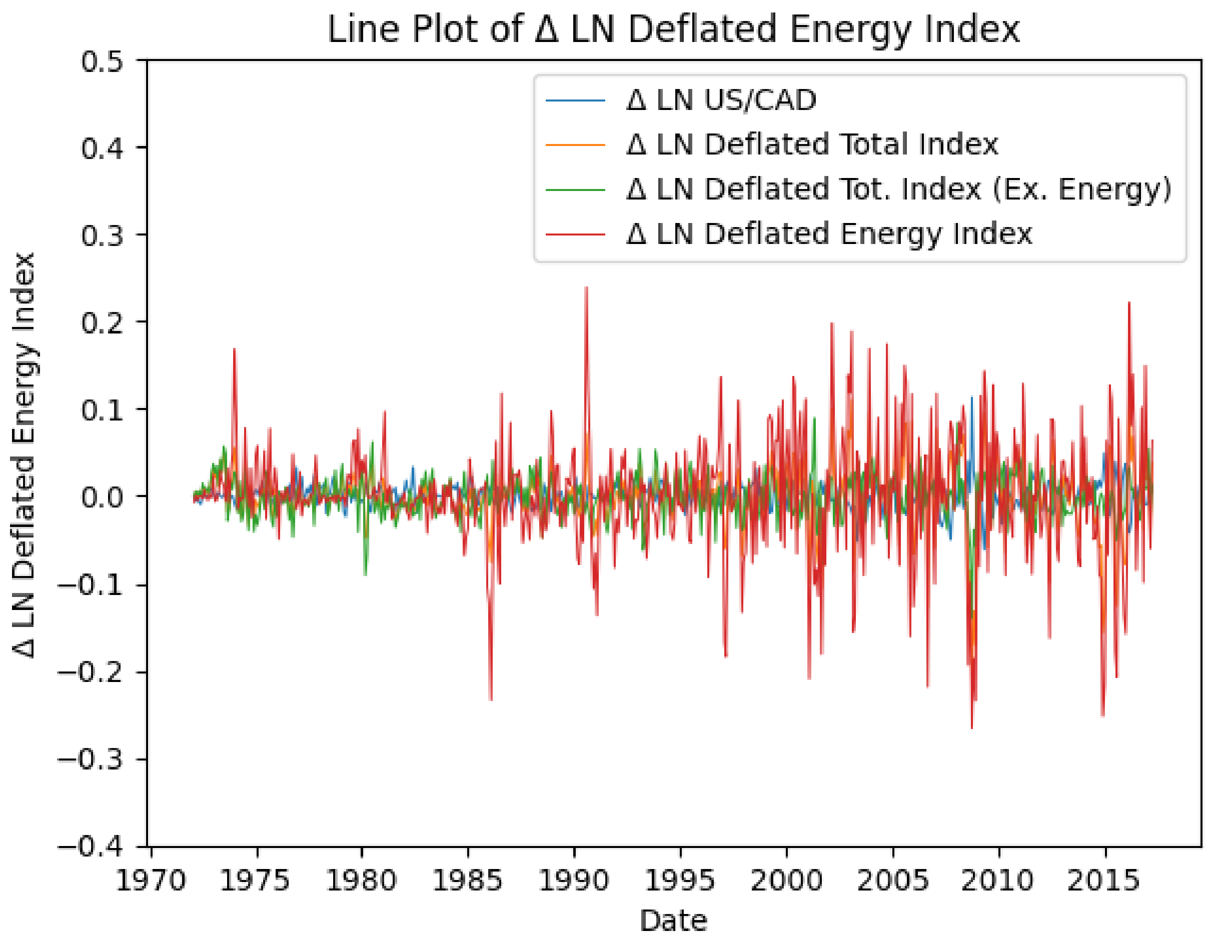


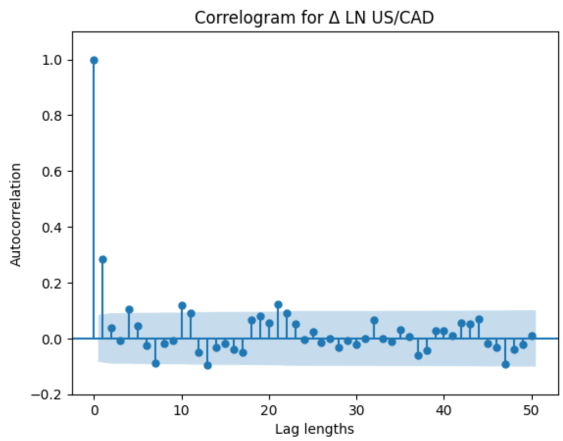
The graphs confirm suspicion of a possible unit root. The respective correlogram show that the variables are correlated with their past values and this correlation does not disappear even after 30 lags. We can objectively confirm this by conducting a unit root test. The test to be used is the augmented Dickey Fuller test with constant and trend. The table below shows the results of the test under the null hypothesis of the presence of a unit root. Lags are chosen for the test using the Bayesian Information Criterion (BIC).

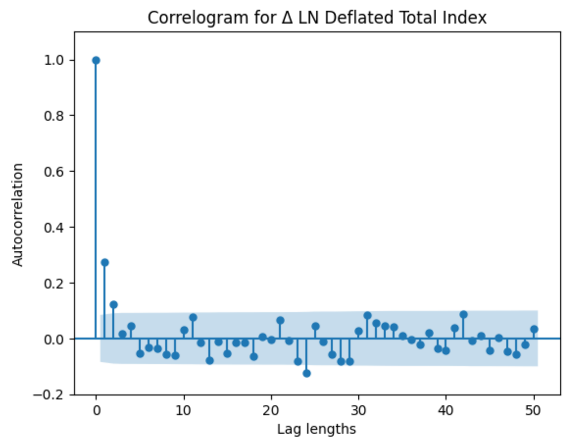
Table 3: Augmented Dickey Fuller Test at Levels

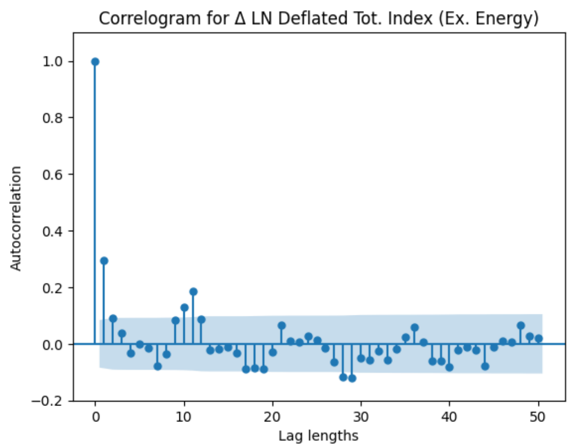
Variable	Test Statistic	P-Value	Optimal lags
Exchange rate (US/CAD)	-1.8355	0.6874	1
Total Commodity Price Index	-2.4207	0.3687	1
Total Commodity Price Index (Excluding Energy)	-2.1768	0.5029	1
Energy Index	-2.6345	0.2643	1

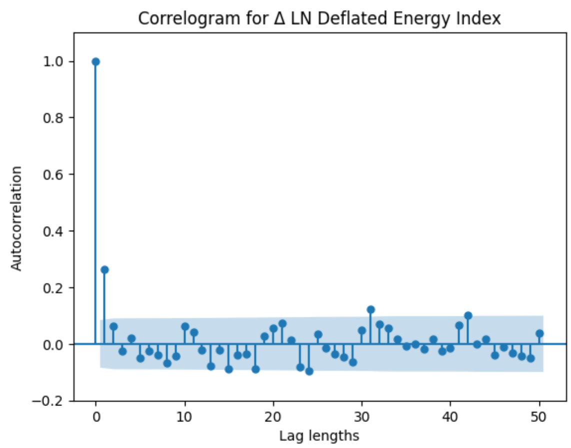
Based on the test, we fail to reject the null hypothesis at the 5% level of significance and conclude that a unit root is present in the variables. We can proceed with first differencing to rid the variables of the unit root issue.









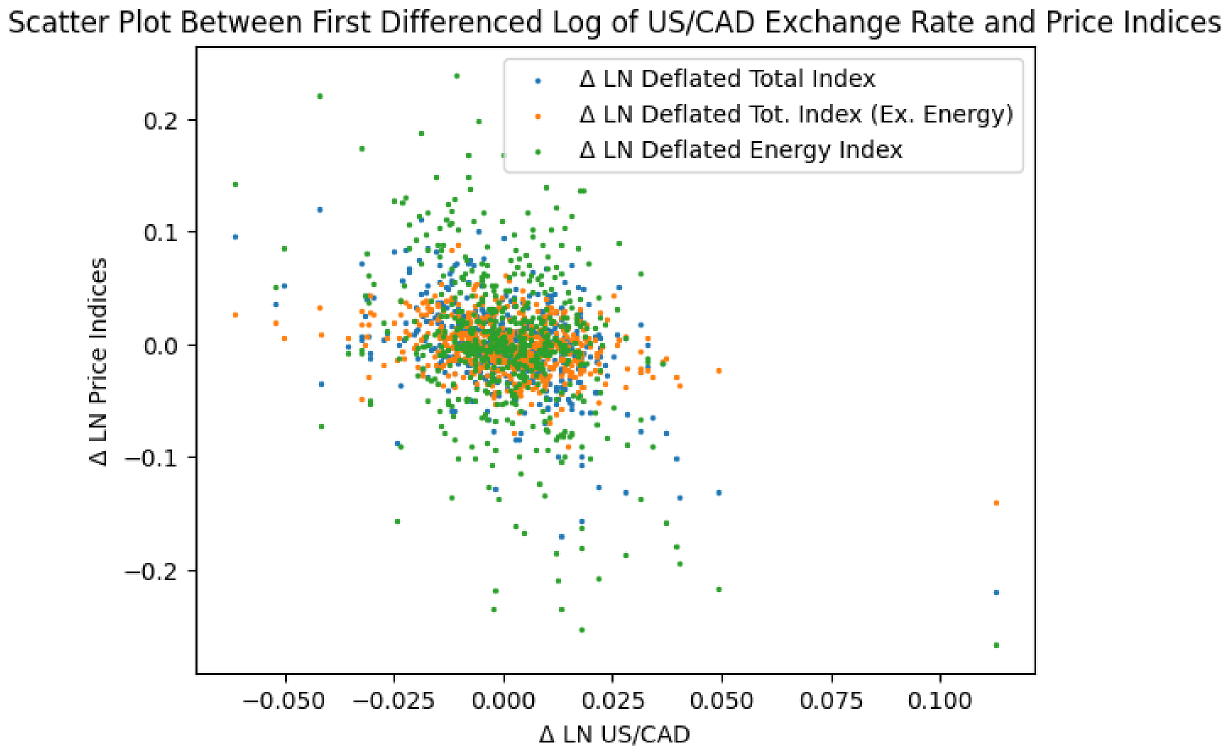


From the graphs above, it is observed that after first differencing, the discernible pattern is removed from the variables. The autocorrelation function shown by the correlogram shows that the relationship with lags of the variable disappears after a few lags. We conduct the augmented Dickey Fuller test once more to confirm that the unit root problem has been tackled.

Table 4: Augmented Dickey Fuller Test at first Differences

Variable	**Test Statistic**	**P-Value**	**Optimal lags**
Δ LN Exchange rate (US/CAD)	-17.3186	0.0000	0
Δ LN Real Total Commodity Price Index	-17.4837	0.0000	0
Δ LN Real Total Commodity Price Index (Excluding Energy)	-17.1401	0.0000	0
Δ Real Energy Index	-17.7468	0.0000	0

The p-value shows that the presence of the unit root disappears after first differencing. As such, our analysis will only make sense in the context of the first differenced variables unless cointegration exists.



The negative correlation however still exists although much milder. The correlation table for the variables in first differences can be found in the appendix. We turn to a cointegration test to confirm our decision to work with first differenced variables.

COINTEGRATION TEST

Although variables containing unit root could lead to a spurious regression when used in a model, there exists a possibility of cointegration, a phenomenon whereby a linear combination of the variables is stationary. The test used is the Engel-Granger cointegration test with the null hypothesis of no cointegration. The table below displays the results of the test.

Table 5: Engle-Granger Cointegration Test between Log US/Canada Exchange Rate and the Various Log Real Price Indices

Variable	Test Statistic	P-Value
LN Real Total Commodity Price Index	-3.5946	0.0797
LN Real Total Commodity Price Index (Excluding Energy)	-4.0088	0.0267
LN Real Energy Index	-2.9456	0.2880

Based on the test results, we fail to reject the null hypothesis at the 5% level and conclude that there exists no long-run relationship between the log exchange rate and both the log real total commodity price index and log real energy index. There however exists a long run relationship between the log exchange rate and the log non-energy price index. Due to this, we shall estimate an error correction model for non-energy price indices and a vector autoregressive model for the remaining variables which do not exhibit a long run relationship.

REGRESSION ANALYSIS

The tables below show the results of our regression estimates. We estimate the appropriate Vector Autoregressive and Vector Error Correction Models.

Table 6: VAR model for ∆ LN Exchange rate (US/CAD) & ∆ LN Deflated Total Index

Dependent variable: ∆ LN Exchange rate (US/CAD)		
	Co-efficient	P-Value
Constant	0.000381	0.524
L1.∆ LN US/CAD	0.243684	0.000
L1.∆ LN Deflated Total Index	-0.036608	0.042
Dependent variable: ∆ LN Deflated Total Index		
	Co-efficient	P-Value
Constant	-0.000414	0.787
L1.∆ LN US/CAD	-0.157715	0.180
L1.∆ LN Deflated Total Index	0.248064	0.000

Table 7: VAR model for ∆ LN Exchange rate (US/CAD) & ∆ LN Deflated Energy Index

Dependent variable: ∆ LN Exchange rate (US/CAD)		
	Co-efficient	P-Value
Constant	0.000408	0.496
L1.∆ LN US/CAD	0.264653	0.000
L1.∆ LN Deflated Energy Index	-0.012834	0.181
Dependent variable: ∆ LN Deflated Energy Index		
	Co-efficient	P-Value
Constant	0.000829	0.764
L1.∆ LN US/CAD	-0.230101	0.257
L1.∆ LN Deflated Energy Index	0.245019	0.000

Table 8: VECM for LN Exchange rate (US/CAD) & LN Deflated Total Commodity Price Index (Excluding Energy)

Dependent variable: Δ LN Exchange rate (US/CAD)		
	Co-efficient	P-Value
Error Correction Term	-0.0038	0.082
L1.Δ LN US/CAD	0.2635	0.000
L1.Δ LN Deflated Total Commodity Price Index (Excluding Energy)	-0.0410	0.118
Dependent variable: Δ LN Deflated Total Commodity Price Index (Excluding Energy)		
	Co-efficient	P-Value
Error Correction Term	0.0041	0.260
L1.Δ LN US/CAD	-0.0731	0.309
L1.Δ LN Deflated Total Commodity Price Index (Excluding Energy)	0.2821	0.000

The three models engender interesting insights. From table 6, we see that the short run dynamics suggest that total commodity prices have a negative significant impact on the exchange rate but not the other way round, the impact of the exchange rate on total commodity prices is insignificant.

Table 6 also shows that in the short run, energy prices do not impact the Canadian exchange rate and neither does the Canadian exchange rate have an impact on energy prices.

Table 7 reveals that although our cointegration tests suggested that a Vector Error Correction Model was more appropriate for examining the relationship between non-energy commodity prices and the Canadian exchange rate, the error correction term in both directions is insignificant. Further to this, the short-run dynamics show that neither does non-energy commodity price affect the Canadian exchange rate nor the exchange rate impact non-energy commodity prices.

The usefulness of the exchange rate for predicting future values of the various energy prices is examined by running a series of Granger causality tests.

GRANGER CAUSALITY TEST

Granger causality, as deceptive as the name may sound, simply is an examination of how a lagged variable enables us to make predictions of the future values of another variable beyond what can be achieved using lagged values of the variable itself. As such, I test whether or not there exists a relationship between the lagged values of the exchange rate and the various commodity prices. The null of the test presupposes no Granger causality.

Table 9: Granger-Causality Tests of US/Canada Exchange Rate on the Various Price Indices

	SSR Based F-test	P-Value
Total Commodity Price Index	4.1302	0.0426
Energy Index	1.7862	0.1819
Total Commodity Price Index (Excluding Energy)	0.0137	0.9069

At the 5% level of significance, the Granger-causality tests suggest that the US/Canada exchange rate has forecasting power in the predictions of the total commodity. This is in line with the findings of Chen et al. (2010). However, the same does not hold for energy and non-energy prices.

DIAGNOSTIC TESTING

A series of diagnostic tests were performed on our models to ensure robustness. The results of the test are shown in the appendix.

Stability testing was carried out and the results show that the absolute value of all eigenvalues is less than one. For all models, the Jarque-Bera normality tests lead to the rejection of the null hypothesis of normally distributed errors at the 5% level of significance. Errors are therefore non-normally distributed. The Ljun-Box Q test for autocorrelation was also run and the results show that there is no autocorrelation in all the models up to 10 lags at the 5% level of significance.

CONCLUSION

Our analysis sheds light on the empirical relationship between the Canadian exchange rate and commodity prices, focusing particularly on total commodity prices, energy prices, and non-energy commodity prices.

Firstly, we find evidence of a negative relationship between the Canadian exchange rate and total commodity prices in the short run, suggesting that fluctuations in commodity prices may impact the exchange rate. However, in the reverse, where the exchange rate influences commodity prices, the impact is not significant. Similarly, we observe no significant relationship between the exchange rate and energy prices or non-energy commodity prices.

Furthermore, our Granger causality tests indicate that the Canadian exchange rate has forecasting power for total commodity prices, but not for energy prices or non-energy commodity prices. These findings suggest that the exchange rate may serve as a useful indicator for predicting movements in total commodity prices in the Canadian economy.

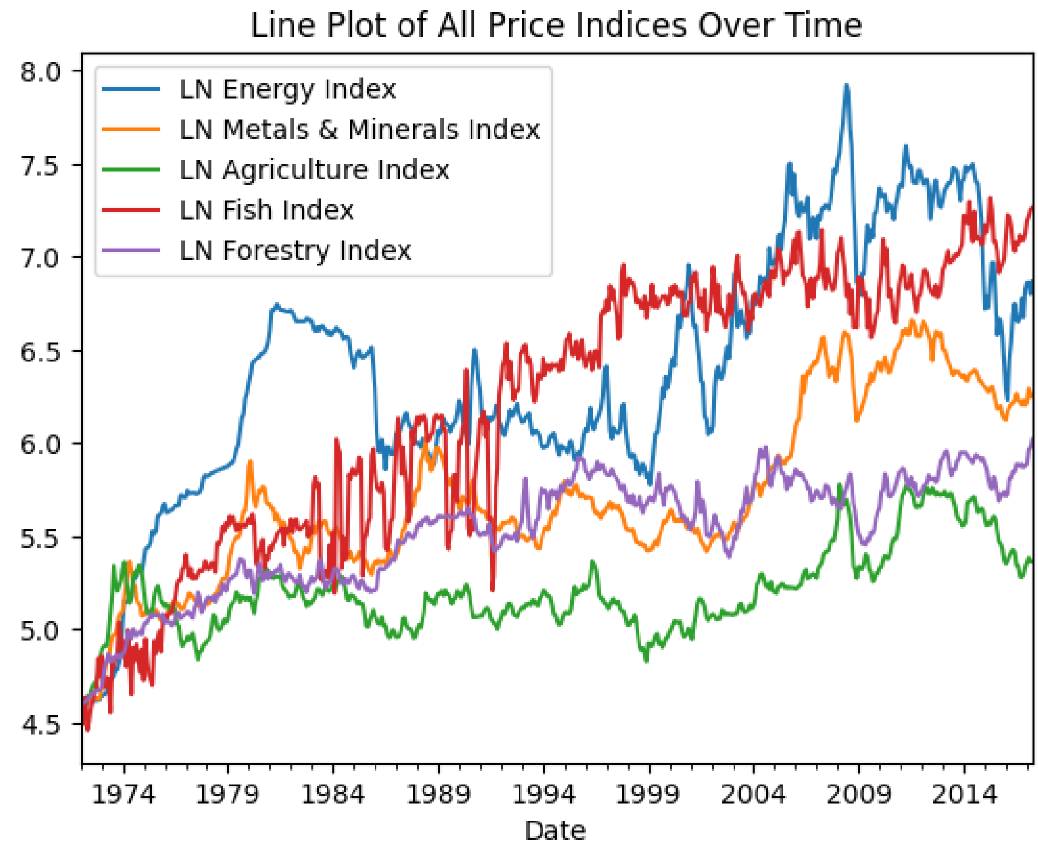
PART II

A short analysis is to be carried out in this part to have a look at the relationship between five key commodity prices, namely, agriculture, metals and minerals, energy, fish and forestry price indices. The purpose of this part is to simply test whether these price indices share the same permanent shocks.

DESCRIPTIVE STATISTICS

Table 10: Descriptive Statistics

Variable	Minimum	Median	Maximum	Mean	Standard deviation
LN Energy Index	4.603168	6.346162	7.921463	6.387978	0.4143
LN Metals & Minerals Index	4.605170	5.607448	6.659166	5.709536	0.467090
LN Agriculture Index	4.605170	5.170484	5.777343	5.205565	0.225740
LN Fish Index	4.457830	6.444527	7.315618	6.203912	0.742417
LN Forestry Index	4.605170	5.596383	6.018593	5.519041	0.313381



The descriptive statistics from table 10 above reveal that prices of fish are the most volatile however, and average energy prices were the largest. Further, the graph above shows how all prices indices evolve over the period. Aside a general increase in all price indices over the period, no discernible pattern is discovered. The agriculture index, on the other hand does not increase much, staying below almost all variables for most of the period.

We run a Dickey fuller test to know the order of integration before applying the Johansen test to see if the variables are affected by the same permanent shocks.

STATIONARITY TEST

Table 11: Augmented Dickey Fuller Test at Levels

Variable	Test Statistic	P-Value	Optimal lags
LN Energy Index	-2.791964	0.199912	1
LN Metals & Minerals Index	-2.564600	0.296452	1
LN Agriculture Index	-3.651722	0.025740	1
LN Fish Index	-2.893394	0.164361	14
LN Forestry Index	-3.902814	0.012025	1

The table above shows the results of the stationarity test. At the 5% level of significance, the agriculture and the forestry index do not contain a unit root. In order to carry on with our analysis, we shall first difference the remaining variables to make all variables of the same order and then proceed with the Johansen cointegration test.

Table 12: Augmented Dickey Fuller Test at First Differences

Variable	**Test Statistic**	**P-Value**	**Optimal lags**
LN Energy Index	-17.433031	0.000	0
LN Metals & Minerals Index	-17.371912	0.000	0
LN Fish Index	-6.427575	0.000	13

Table 12 confirms that after first differencing, all variables that have a unit root are now stationary. We can now proceed with the Johansen cointegration test with much confidence.

COINTEGRATION TEST

We make use of the Johansen test to examine if there exists cointegration among the different price indices. The null hypothesis of the test is that there are at most r number of cointegrating relationships. The table below presents the trace statistics for the test as well as the 5% critical values of the test.

Table 13: Test Results for the Johansen Cointegration Test

Number of cointegrating equations	Trace	5% Critical Values
1	629.88927991	69.8189
2	376.51651155	47.8545
3	184.85898849	29.7961
4	16.92448138	15.4943
5	5.9729954	3.8415

The trace exceeds the 5% critical value in all instances, hence, we conclude that all the price indices have a long-run relationship and thereby share the same permanent shocks.

REFERENCES

Chen, Y.-C., Rogoff, K.S., Rossi, B. (2010) "Can exchange rates forecast commodity prices" Quarterly Journal of Economics 125, pp. 1145 – 94

Statistics Canada, Table 12-10-0122-01, https://www.international.gc.ca/transparency-transparence/state-trade-commerce-international/2023.aspx?lang=eng

appendix

February 28, 2024

[1]:
```python
# Importing relevant libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm
from scipy.stats import pearsonr
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.stattools import coint
from statsmodels.tsa.stattools import grangercausalitytests
from statsmodels.tsa.api import VAR, VECM
from statsmodels.stats.diagnostic import het_white
from statsmodels.stats.stattools import jarque_bera
from statsmodels.tsa.stattools import acf
from statsmodels.tsa.vector_ar.vecm import select_order
import statsmodels.tsa.vector_ar
from statsmodels.tsa.vector_ar.vecm import coint_johansen
```

[2]:
```python
# Loading the first dataset
exchange_rates = pd.read_csv('/content/drive/MyDrive/Data sets/
↪StatsCanExchangeRates.csv')
exchange_rates.head()
```

[2]:
```
  REF_DATE     GEO  DGUID                                Type of currency  \
0  1950-10  Canada    NaN   United States dollar, noon spot rate, average
1  1950-10  Canada    NaN  United States dollar, 90-day forward noon rate
2  1950-10  Canada    NaN          Belgian franc, noon spot rate, average
3  1950-10  Canada    NaN           Danish krone, noon spot rate, average
4  1950-10  Canada    NaN           French franc, noon spot rate, average

       UOM  UOM_ID SCALAR_FACTOR  SCALAR_ID  VECTOR  COORDINATE     VALUE  \
0  Dollars      81         units          0  v37426        1.10  1.053333
1  Dollars      81         units          0  v37437        1.22  1.047313
2  Dollars      81         units          0  v37448        1.20  0.020928
3  Dollars      81         units          0  v37452        1.30  0.152562
4  Dollars      81         units          0  v37453        1.40  0.003014
```

```
  STATUS  SYMBOL TERMINATED  DECIMALS
0    NaN     NaN        NaN         8
1    NaN     NaN        NaN         8
2    NaN     NaN          t         8
3    NaN     NaN        NaN         8
4    NaN     NaN          t         8
```

[3]:
```python
# Filtering for only US/CAD related data
exchange_rates = exchange_rates[exchange_rates['Type of currency'] == 'United States dollar, noon spot rate, average']
exchange_rates.head()
```

[3]:
```
   REF_DATE     GEO  DGUID                              Type of currency  \
0   1950-10  Canada    NaN  United States dollar, noon spot rate, average
13  1950-11  Canada    NaN  United States dollar, noon spot rate, average
26  1950-12  Canada    NaN  United States dollar, noon spot rate, average
39  1951-01  Canada    NaN  United States dollar, noon spot rate, average
55  1951-02  Canada    NaN  United States dollar, noon spot rate, average

        UOM  UOM_ID SCALAR_FACTOR  SCALAR_ID  VECTOR  COORDINATE     VALUE  \
0   Dollars      81         units          0  v37426         1.1  1.053333
13  Dollars      81         units          0  v37426         1.1  1.040312
26  Dollars      81         units          0  v37426         1.1  1.053078
39  Dollars      81         units          0  v37426         1.1  1.051875
55  Dollars      81         units          0  v37426         1.1  1.049125

    STATUS  SYMBOL TERMINATED  DECIMALS
0      NaN     NaN        NaN         8
13     NaN     NaN        NaN         8
26     NaN     NaN        NaN         8
39     NaN     NaN        NaN         8
55     NaN     NaN        NaN         8
```

[4]:
```python
# Filtering for only relevant columns
filtered_ex_rate = exchange_rates[['REF_DATE', 'VALUE']]
filtered_ex_rate.columns = ['Date','US/CAD']

# Converting the date to a date type
filtered_ex_rate['Date'] = pd.to_datetime(filtered_ex_rate['Date'], format='%Y-%m')

filtered_ex_rate.head()
```

```
<ipython-input-4-fe1ea1171c26>:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  filtered_ex_rate['Date'] = pd.to_datetime(filtered_ex_rate['Date'],
format='%Y-%m')
```

[4]:

```
          Date    US/CAD
0   1950-10-01  1.053333
13  1950-11-01  1.040312
26  1950-12-01  1.053078
39  1951-01-01  1.051875
55  1951-02-01  1.049125
```

[5]:

```python
# Loading second dataset
price_indices = pd.read_csv('/content/drive/MyDrive/Data sets/
 ↪StatsCanPriceIndices.csv')
price_indices
```

[5]:

```
     REF_DATE     GEO          DGUID                 Commodity  \
0     1972-01  Canada  2016A000011124  Total, all commodities
1     1972-01  Canada  2016A000011124  Total excluding energy
2     1972-01  Canada  2016A000011124                  Energy
3     1972-01  Canada  2016A000011124     Metals and Minerals
4     1972-01  Canada  2016A000011124             Agriculture
…         …       …               …                       …
4363  2023-12  Canada  2016A000011124                  Energy
4364  2023-12  Canada  2016A000011124     Metals and Minerals
4365  2023-12  Canada  2016A000011124             Agriculture
4366  2023-12  Canada  2016A000011124                    Fish
4367  2023-12  Canada  2016A000011124                Forestry

                 UOM  UOM_ID SCALAR_FACTOR  SCALAR_ID     VECTOR  COORDINATE  \
0     Index, 1972=100     166        units          0  v52673496         1.1
1     Index, 1972=100     166        units          0  v52673497         1.2
2     Index, 1972=100     166        units          0  v52673498         1.3
3     Index, 1972=100     166        units          0  v52673499         1.4
4     Index, 1972=100     166        units          0  v52673500         1.5
…                  …       …            …          …          …           …
4363  Index, 1972=100     166         units         0  v52673498         1.3
4364  Index, 1972=100     166         units         0  v52673499         1.4
4365  Index, 1972=100     166         units         0  v52673500         1.5
4366  Index, 1972=100     166         units         0  v52673501         1.6
4367  Index, 1972=100     166         units         0  v52673502         1.7

      VALUE  STATUS  SYMBOL  TERMINATED  DECIMALS
0     100.0     NaN     NaN         NaN         1
1     100.0     NaN     NaN         NaN         1
2     100.0     NaN     NaN         NaN         1
```

```
3      100.0     NaN     NaN         NaN         1
4      100.0     NaN     NaN         NaN         1
…        …     …     …         …       …
4363   1285.1     NaN     NaN         NaN         1
4364    696.5     NaN     NaN         NaN         1
4365    285.6     NaN     NaN         NaN         1
4366   1634.8     NaN     NaN         NaN         1
4367    453.8     NaN     NaN         NaN         1

[4368 rows x 15 columns]
```

[6]:
```python
# Filtering data for relevant variables and placing them in different columns
filtered_price_indices = pd.DataFrame()
filtered_price_indices['Date'] = price_indices.loc[price_indices['Commodity']␣
 ↪== 'Total, all commodities', 'REF_DATE'].values
filtered_price_indices['Total Index'] = price_indices.
 ↪loc[price_indices['Commodity']=='Total, all commodities','VALUE'].values
filtered_price_indices['Tot. Index (Ex. Energy)'] = price_indices.
 ↪loc[price_indices['Commodity']=='Total excluding energy','VALUE'].values
filtered_price_indices['Energy Index'] = price_indices.
 ↪loc[price_indices['Commodity']=='Energy','VALUE'].values
filtered_price_indices['Metals & Minerals Index'] = price_indices.
 ↪loc[price_indices['Commodity']=='Metals and Minerals','VALUE'].values
filtered_price_indices['Agriculture Index'] = price_indices.
 ↪loc[price_indices['Commodity']=='Agriculture','VALUE'].values
filtered_price_indices['Fish Index'] = price_indices.
 ↪loc[price_indices['Commodity']=='Fish','VALUE'].values
filtered_price_indices['Forestry Index'] = price_indices.
 ↪loc[price_indices['Commodity']=='Forestry','VALUE'].values

# Converting the date to a date type
filtered_price_indices['Date'] = pd.to_datetime(filtered_price_indices['Date'],␣
 ↪format='%Y-%m')

filtered_price_indices
```

[6]:
```
          Date  Total Index  Tot. Index (Ex. Energy)  Energy Index  \
0   1972-01-01        100.0                    100.0         100.0
1   1972-02-01        100.4                    100.5          99.8
2   1972-03-01        101.1                    101.3         100.1
3   1972-04-01        101.2                    101.5          99.8
4   1972-05-01        101.9                    102.3         100.0
..         …            …                        …             …
619 2023-08-01        625.8                    436.4        1483.6
620 2023-09-01        649.5                    425.5        1611.3
621 2023-10-01        620.5                    416.1        1513.3
```

```
622 2023-11-01        578.3                    418.5        1334.6
623 2023-12-01        565.4                    417.7        1285.1

     Metals & Minerals Index  Agriculture Index  Fish Index  Forestry Index
0                      100.0              100.0       100.0           100.0
1                      100.7              101.2        88.9           100.1
2                      101.4              102.5        99.0           100.2
3                      101.2              102.1       103.1           100.9
4                      101.3              103.5        86.3           102.3
..                       …                  …          …               …
619                    713.6              322.2      1595.0           436.2
620                    712.7              304.4      1603.6           424.9
621                    700.3              291.6      1628.2           424.7
622                    702.1              288.7      1591.1           443.3
623                    696.5              285.6      1634.8           453.8

[624 rows x 8 columns]
```

[7]:
```python
# Loading third dataset
cpi_data = pd.read_excel('/content/drive/MyDrive/Data sets/cpidata.xlsx')
cpi_data
```

[7]:
```
     Year      Jan      Feb      Mar      Apr      May      Jun      Jul  \
0    1913    9.800    9.800    9.800    9.800    9.700    9.800    9.900
1    1914   10.000    9.900    9.900    9.800    9.900    9.900   10.000
2    1915   10.100   10.000    9.900   10.000   10.100   10.100   10.100
3    1916   10.400   10.400   10.500   10.600   10.700   10.800   10.800
4    1917   11.700   12.000   12.000   12.600   12.800   13.000   12.800
..    …      …        …        …        …        …        …        …
107  2020  257.971  258.678  258.115  256.389  256.394  257.797  259.101
108  2021  261.582  263.014  264.877  267.054  269.195  271.696  273.003
109  2022  281.148  283.716  287.504  289.109  292.296  296.311  296.276
110  2023  299.170  300.840  301.836  303.363  304.127  305.109  305.691
111  2024  308.417      NaN      NaN      NaN      NaN      NaN      NaN

         Aug      Sep      Oct      Nov      Dec
0      9.900   10.000   10.000   10.100   10.000
1     10.200   10.200   10.100   10.200   10.100
2     10.100   10.100   10.200   10.300   10.300
3     10.900   11.100   11.300   11.500   11.600
4     13.000   13.300   13.500   13.500   13.700
..      …        …        …        …        …
107  259.918  260.280  260.388  260.229  260.474
108  273.567  274.310  276.589  277.948  278.802
109  296.171  296.808  298.012  297.711  296.797
110  307.026  307.789  307.671  307.051  306.746
111      NaN      NaN      NaN      NaN      NaN
```

```
[112 rows x 13 columns]
```

[8]:
```python
# Retaining only relevant columns
cpi_data = cpi_data.iloc[:,1:]

# Re-arranging cpi values into a single column in a new data frame
cpi_data_new = pd.DataFrame()
data_list = []
for i in range(len(cpi_data)):
    x = list(cpi_data.iloc[i])
    data_list += x

cpi_data_new['CPI']=data_list

# Adding a date column
start_date = pd.to_datetime('1913-01')
cpi_data_new['Date'] = pd.
 ↪date_range(start=start_date,freq='MS',periods=len(cpi_data_new))

cpi_data_new
```

[8]:
```
      CPI       Date
0     9.8 1913-01-01
1     9.8 1913-02-01
2     9.8 1913-03-01
3     9.8 1913-04-01
4     9.7 1913-05-01
…     …          …
1339  NaN 2024-08-01
1340  NaN 2024-09-01
1341  NaN 2024-10-01
1342  NaN 2024-11-01
1343  NaN 2024-12-01

[1344 rows x 2 columns]
```

[9]:
```python
# Merging all three data sets
merged_data = pd.merge(filtered_ex_rate, filtered_price_indices,on='Date').
 ↪dropna()
merged_data = pd.merge(cpi_data_new, merged_data, on='Date').dropna()
merged_data.set_index('Date',inplace=True)
merged_data.tail()
```

[9]:
```
                CPI    US/CAD  Total Index  Tot. Index (Ex. Energy)  \
Date
2016-12-01  241.432  1.332935        388.8                    304.7
```

```
2017-01-01  242.839  1.319090       398.4                        310.0
2017-02-01  243.603  1.310989       409.9                        328.2
2017-03-01  243.801  1.338752       393.5                        321.2
2017-04-01  244.524  1.344395       410.0                        326.4

            Energy Index  Metals & Minerals Index  Agriculture Index  \
Date
2016-12-01         919.9                    494.9              207.4
2017-01-01         953.6                    500.9              212.4
2017-02-01         953.8                    540.1              217.7
2017-03-01         898.4                    516.3              213.7
2017-04-01         959.9                    524.1              213.7

            Fish Index  Forestry Index
Date
2016-12-01      1239.7           357.1
2017-01-01      1329.9           360.5
2017-02-01      1361.2           389.7
2017-03-01      1413.0           393.5
2017-04-01      1424.7           411.0
```

[10]:
```
# Deflating the data by the US CPI
deflated_data = merged_data.iloc[:,1:5].copy()
for col in deflated_data.columns:
  if col != 'US/CAD':
    deflated_data[col]= deflated_data[col]/merged_data['CPI']

deflated_data.columns= [f'Deflated {col}' if col!='US/CAD' else col for col in␣
 ↪deflated_data.columns]
deflated_data
```

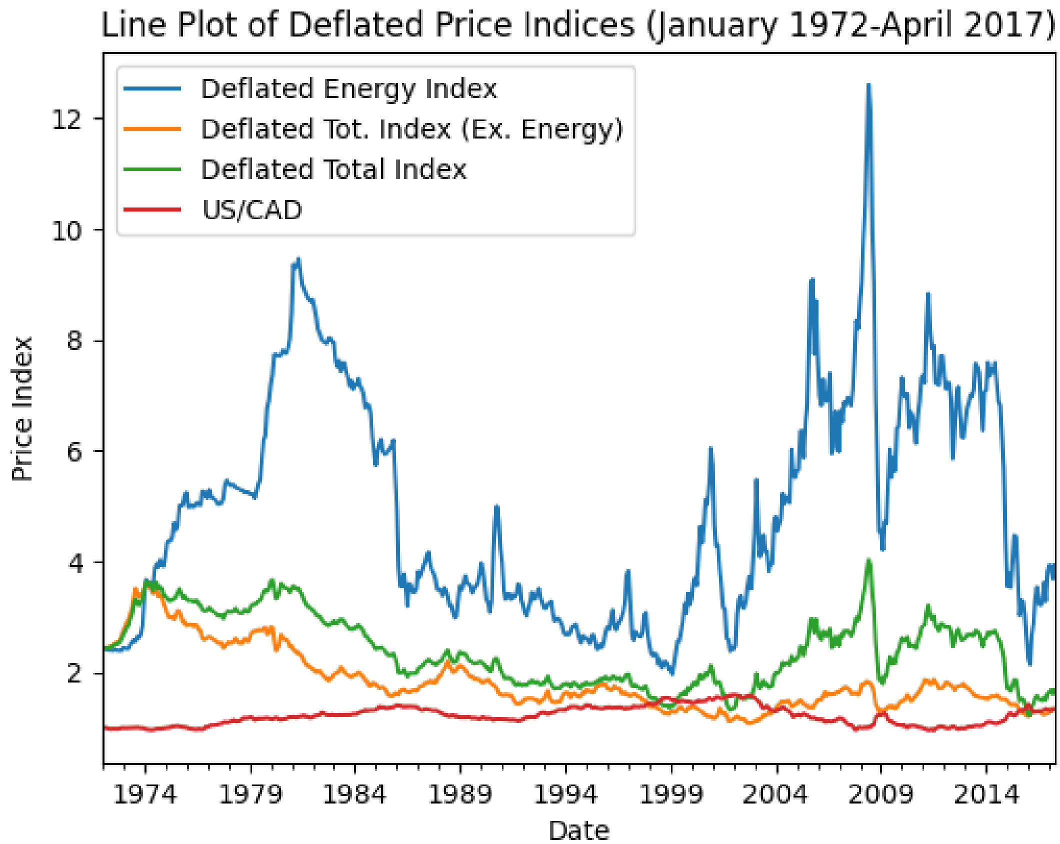
[10]:
```
              US/CAD  Deflated Total Index  Deflated Tot. Index (Ex. Energy)  \
Date
1972-01-01  1.005922              2.433090                          2.433090
1972-02-01  1.004583              2.430993                          2.433414
1972-03-01  0.998395              2.442029                          2.446860
1972-04-01  0.995594              2.438554                          2.445783
1972-05-01  0.988665              2.449519                          2.459135
…                …                   …                                …
2016-12-01  1.332935              1.610391                          1.262053
2017-01-01  1.319090              1.640593                          1.276566
2017-02-01  1.310989              1.682656                          1.347274
2017-03-01  1.338752              1.614021                          1.317468
2017-04-01  1.344395              1.676727                          1.334838

            Deflated Energy Index
Date
```

```
1972-01-01                2.433090
1972-02-01                2.416465
1972-03-01                2.417874
1972-04-01                2.404819
1972-05-01                2.403846
…                            …
2016-12-01                3.810183
2017-01-01                3.926882
2017-02-01                3.915387
2017-03-01                3.684973
2017-04-01                3.925586

[544 rows x 4 columns]
```

[11]:
```
# Summary Statistics
deflated_data.describe()
```

[11]:
```
           US/CAD  Deflated Total Index  Deflated Tot. Index (Ex. Energy)  \
count  544.000000            544.000000                        544.000000
mean     1.221514              2.400493                          1.842344
std      0.166675              0.636366                          0.561874
min      0.955300              1.226852                          1.083287
25%      1.074648              1.825524                          1.462654
50%      1.204636              2.336201                          1.667160
75%      1.352579              2.886606                          2.023810
max      1.600286              4.027603                          3.608051

       Deflated Energy Index
count             544.000000
mean                4.932531
std                 2.035377
min                 1.959271
25%                 3.257409
50%                 4.536728
75%                 6.750231
max                12.594201
```

[12]:
```
# Data plot of deflated price indices
deflated_data[['Deflated Energy Index','Deflated Tot. Index (Ex.␣
 ↪Energy)','Deflated Total Index','US/CAD']].plot()
plt.ylabel('Price Index')
plt.title('Line Plot of Deflated Price Indices (January 1972-April 2017)')
plt.show()
```



[13]:
```
logged_deflated_data = np.log(deflated_data.copy())
logged_deflated_data.columns = [f'LN {col}' for col in logged_deflated_data.
 ↪columns]
logged_deflated_data
```

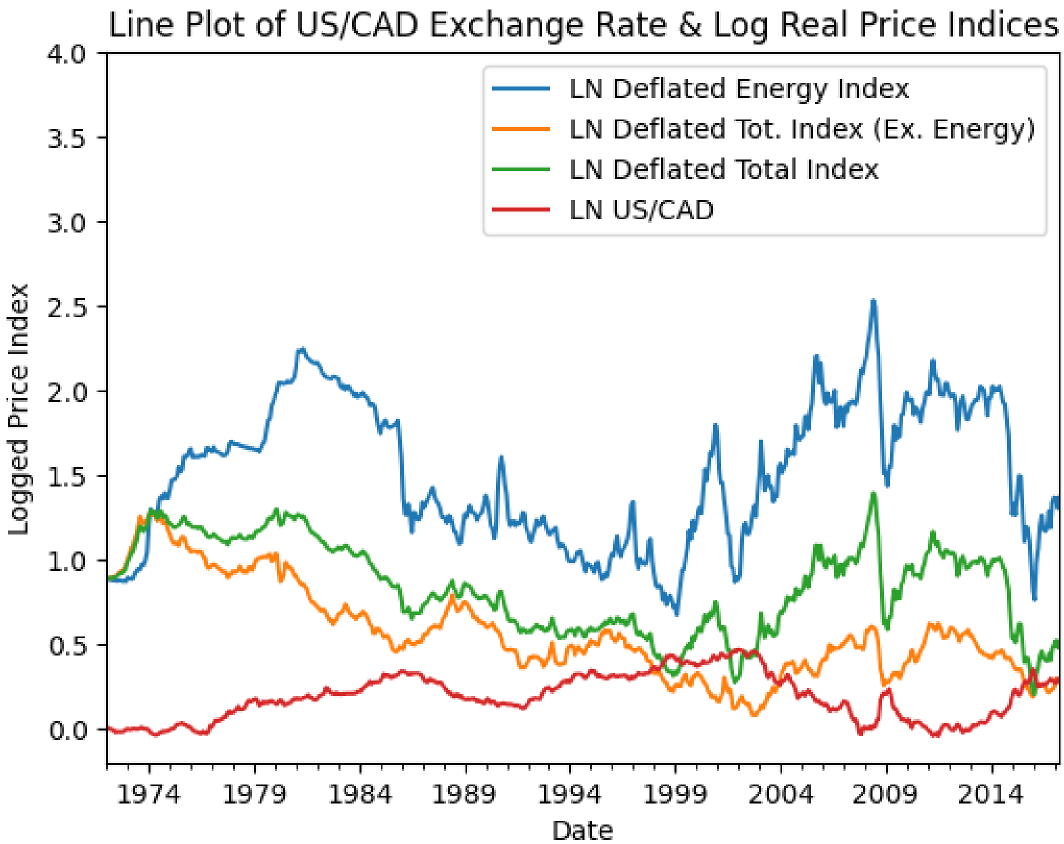
[13]:
```
            LN US/CAD  LN Deflated Total Index  \
Date
1972-01-01   0.005904                 0.889162
1972-02-01   0.004573                 0.888300
1972-03-01  -0.001606                 0.892829
1972-04-01  -0.004416                 0.891405
1972-05-01  -0.011400                 0.895892
…                 …                      …
2016-12-01   0.287383                 0.476477
2017-01-01   0.276942                 0.495058
2017-02-01   0.270782                 0.520373
2017-03-01   0.291738                 0.478729
2017-04-01   0.295944                 0.516844

            LN Deflated Tot. Index (Ex. Energy)  LN Deflated Energy Index
```

```
Date
1972-01-01                                       0.889162                    0.889162
1972-02-01                                       0.889295                    0.882306
1972-03-01                                       0.894806                    0.882889
1972-04-01                                       0.894365                    0.877475
1972-05-01                                       0.899810                    0.877070
…                                                   …                           …
2016-12-01                                       0.232740                    1.337677
2017-01-01                                       0.244174                    1.367846
2017-02-01                                       0.298083                    1.364914
2017-03-01                                       0.275712                    1.304263
2017-04-01                                       0.288810                    1.367516

[544 rows x 4 columns]
```

[14]:
```
#Summary Statistics
logged_deflated_data.describe()
```

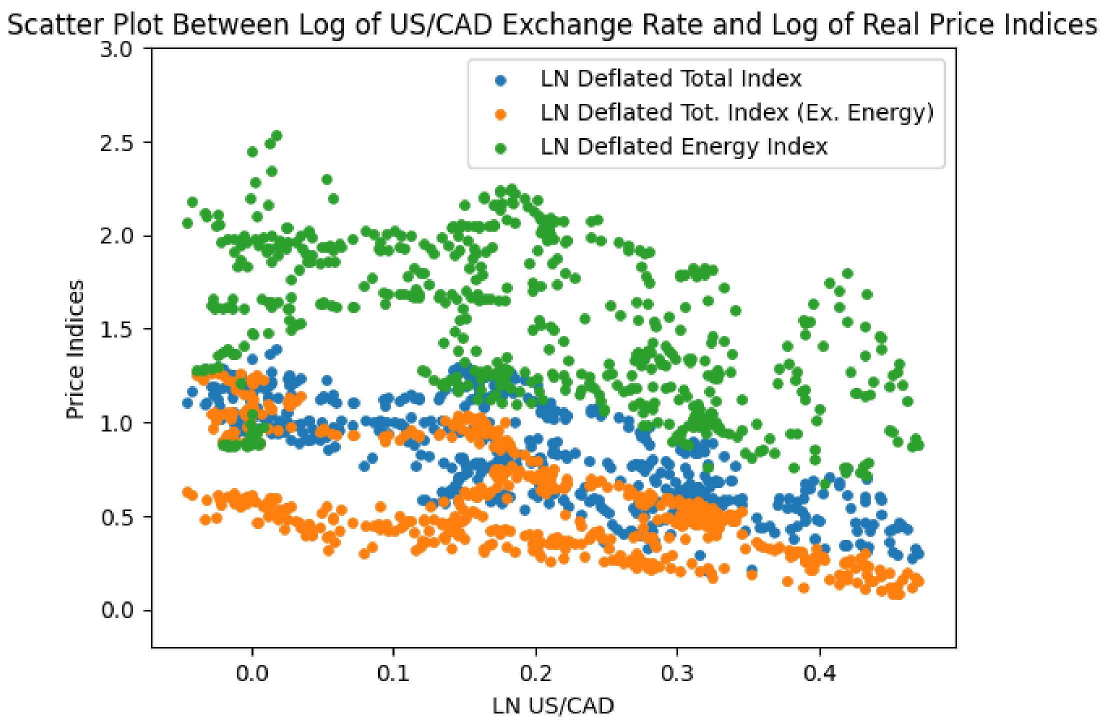
[14]:
```
        LN US/CAD  LN Deflated Total Index  \
count  544.000000               544.000000
mean     0.190835                 0.839843
std      0.136161                 0.270047
min     -0.045730                 0.204451
25%      0.071993                 0.601867
50%      0.186178                 0.848526
75%      0.302013                 1.060079
max      0.470183                 1.393171

       LN Deflated Tot. Index (Ex. Energy)  LN Deflated Energy Index
count                           544.000000                544.000000
mean                              0.570374                  1.511074
std                               0.277362                  0.414298
min                               0.080000                  0.672572
25%                               0.380252                  1.180932
50%                               0.511121                  1.512206
75%                               0.704982                  1.909576
max                               1.283168                  2.533236
```

[15]:
```
# Data plot of logged deflated price indices
logged_deflated_data[['LN Deflated Energy Index','LN Deflated Tot. Index (Ex.␣
↪Energy)','LN Deflated Total Index','LN US/CAD']].plot()
plt.ylabel('Logged Price Index')
plt.title('Line Plot of US/CAD Exchange Rate & Log Real Price Indices')
plt.ylim(-0.2,4)
plt.show()
```



[16]:

```
# Scatter plot Between log US/CAD exchange rate and the various price indices
for col in logged_deflated_data:
    if col != 'LN US/CAD':
        plt.scatter(logged_deflated_data['LN US/
 ↪CAD'],logged_deflated_data[col], marker='o',s=15,label=col)

plt.xlabel('LN US/CAD')
plt.ylabel('Price Indices')
plt.legend()
plt.ylim(-0.2,3)
plt.title('Scatter Plot Between Log of US/CAD Exchange Rate and Log of Real␣
 ↪Price Indices')
plt.show()
```

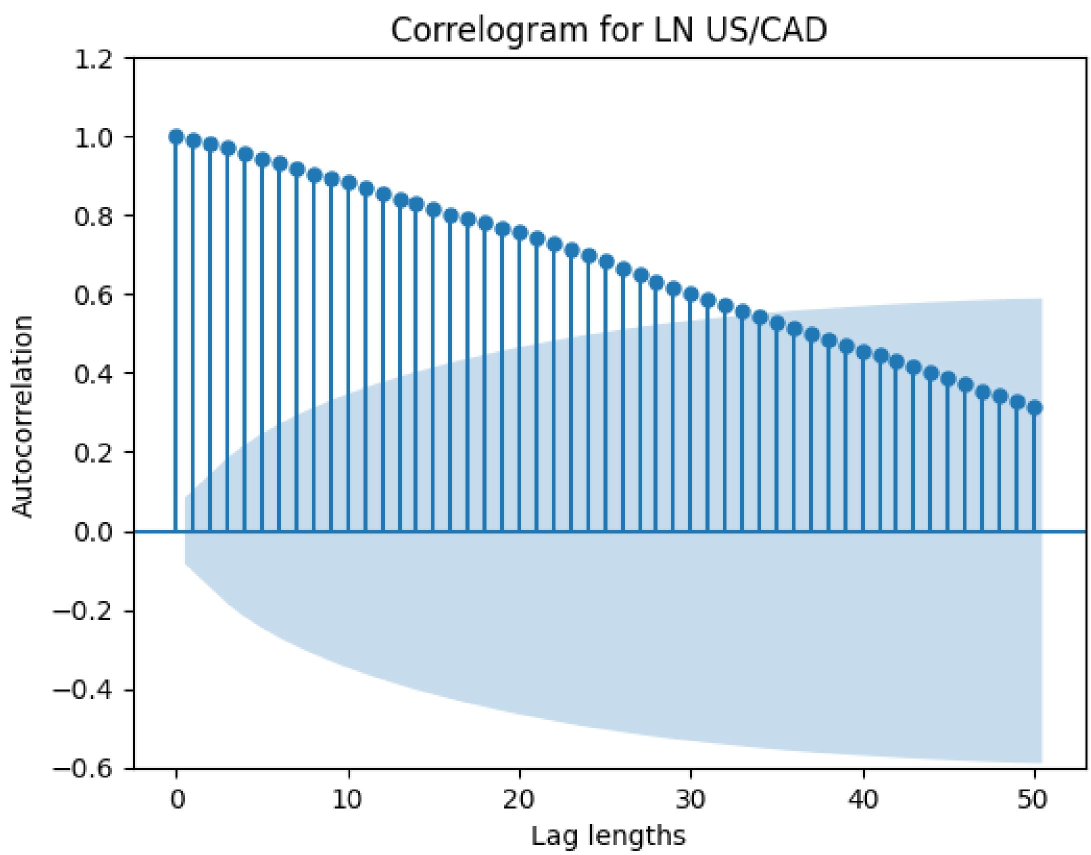


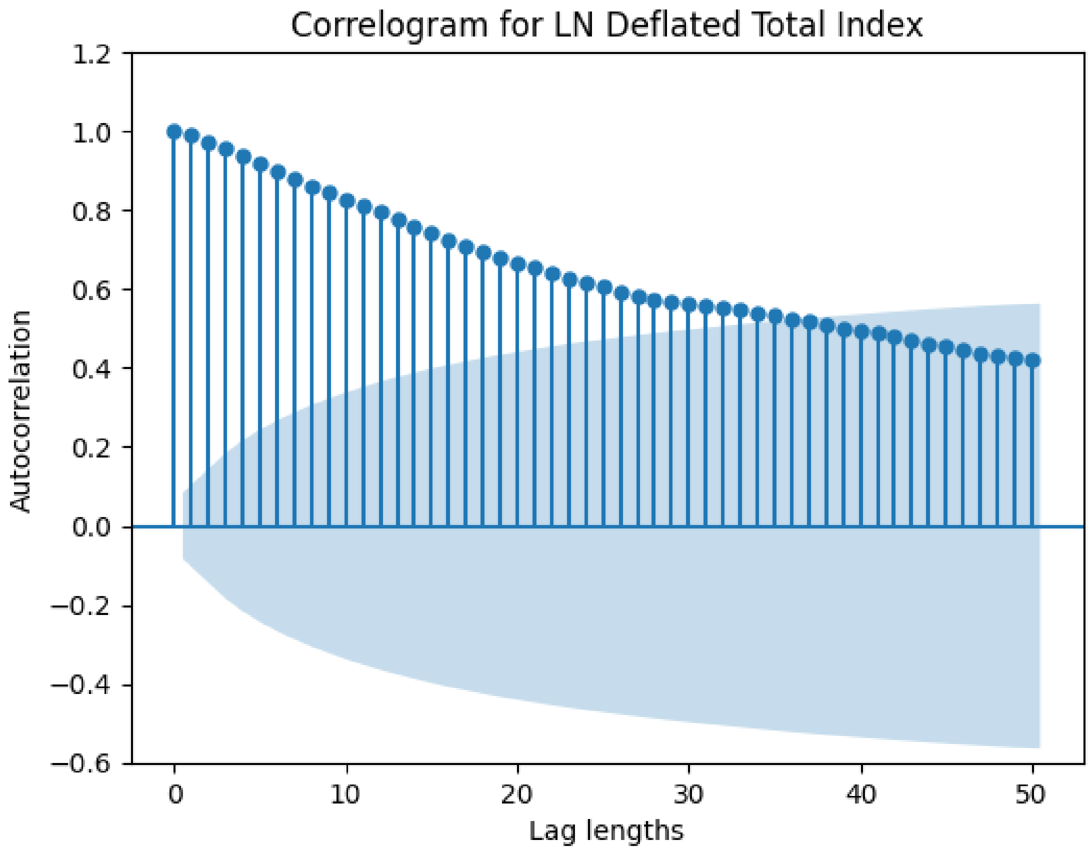
[17]:
```python
# Table of correlation between Exchange rate and Price Indices
correlations = []
p_values = []

for col in logged_deflated_data.columns:
    if col != 'LN US/CAD':
        corr, p_value =␣
 ↪pearsonr(logged_deflated_data[col],logged_deflated_data['LN US/CAD'])
        correlations.append(corr)
        p_values.append(p_value)

correlation_table = pd.DataFrame({'Correlation':correlations,'P-Value':
 ↪p_values},
                                 index=['LN Deflated Total Index','LN Deflated␣
 ↪Tot. Index (Ex. Energy)','LN Deflated Energy Index'])
correlation_table
```

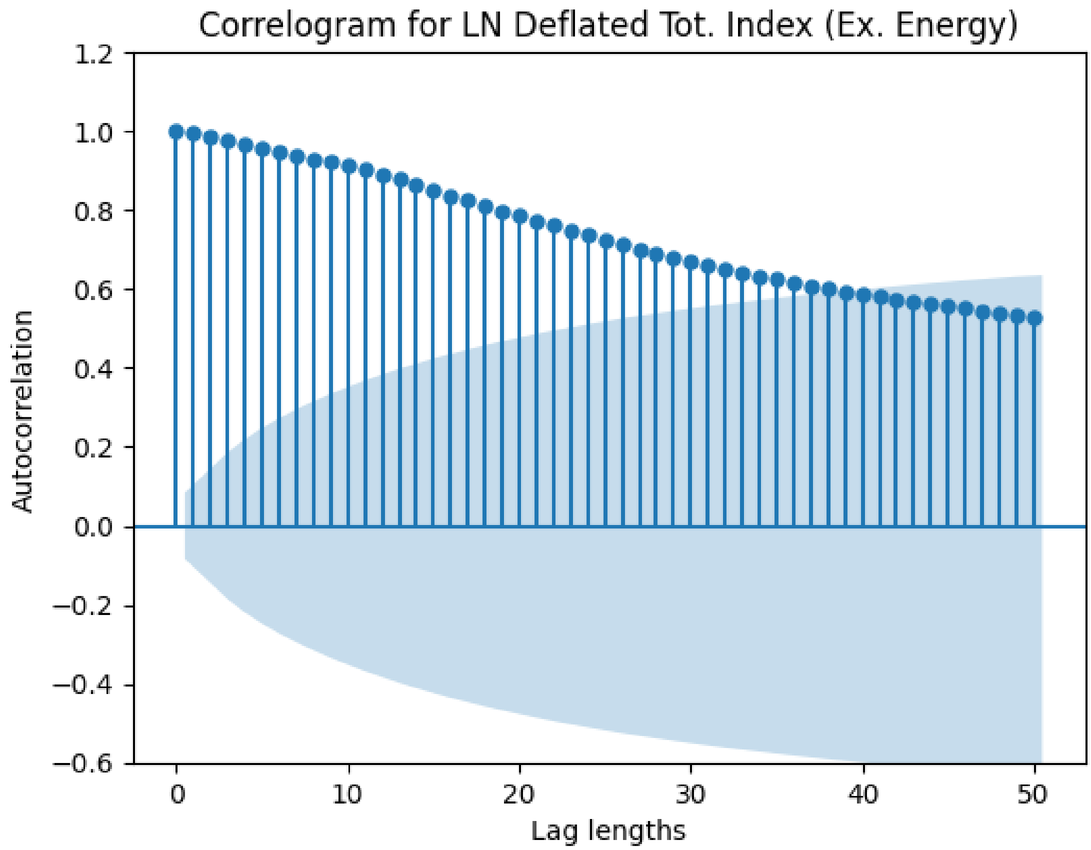
[17]:

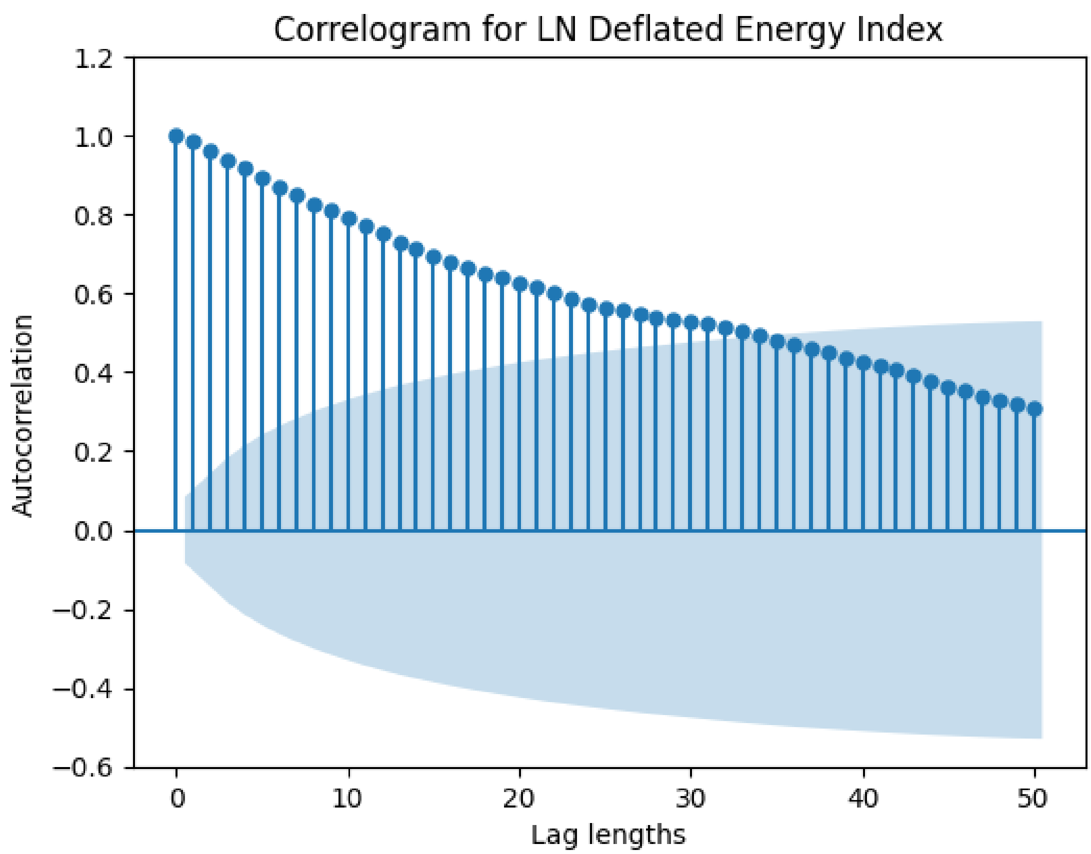
	Correlation	P-Value
LN Deflated Total Index	-0.755309	1.554005e-101
LN Deflated Tot. Index (Ex. Energy)	-0.647738	5.007718e-66
LN Deflated Energy Index	-0.423291	4.638060e-25

[18]:
```
# Graph of autocorrelation function for all variables
for col in logged_deflated_data.columns:
  plot_acf(logged_deflated_data[col], lags=50)
  plt.title(f'Correlogram for {col}')
  plt.xlabel('Lag lengths')
  plt.ylabel('Autocorrelation')
  plt.ylim(-0.6,1.2)
  plt.show()
```




Correlogram for LN Deflated Total Index
Autocorrelation
Lag lengths
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
10
20
30
40
50


Correlogram for LN Deflated Tot. Index (Ex. Energy)
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
Autocorrelation
0
10
20
30
40
50
Lag lengths



[19]:
```
# Augmented Dickey Fuller test for all variables
test_statistic= []
p_value = []
lag_order = []
for col in logged_deflated_data.columns:
    test_result = adfuller(logged_deflated_data[col], regression='ct', autolag
 ↪= 'BIC')
    test_stat, p_val, lag = test_result[:3]
    test_statistic.append(test_stat)
    p_value.append(p_val)
    lag_order.append(lag)

adf_table = pd.DataFrame({'Test Statistic': test_statistic,
                                     'P-value': p_value,
                                     'Optimal Lags': lag_order},
                                    index = ['LN US/CAD','LN Deflated Total
 ↪Index','LN Deflated Tot. Index (Ex. Energy)','LN Deflated Energy Index'])

adf_table
```

[19]:

```
                                       Test Statistic   P-value  Optimal Lags
LN US/CAD                                   -1.835534  0.687377             1
LN Deflated Total Index                     -2.420684  0.368682             1
LN Deflated Tot. Index (Ex. Energy)         -2.176795  0.502918             1
LN Deflated Energy Index                    -2.634461  0.264288             1
```

[20]:

```python
# Creating a dataframe for variables in first differences
differenced_data = logged_deflated_data.copy().diff().dropna()
differenced_data.columns = [f'\u0394 {col}' for col in differenced_data.columns]

differenced_data
```

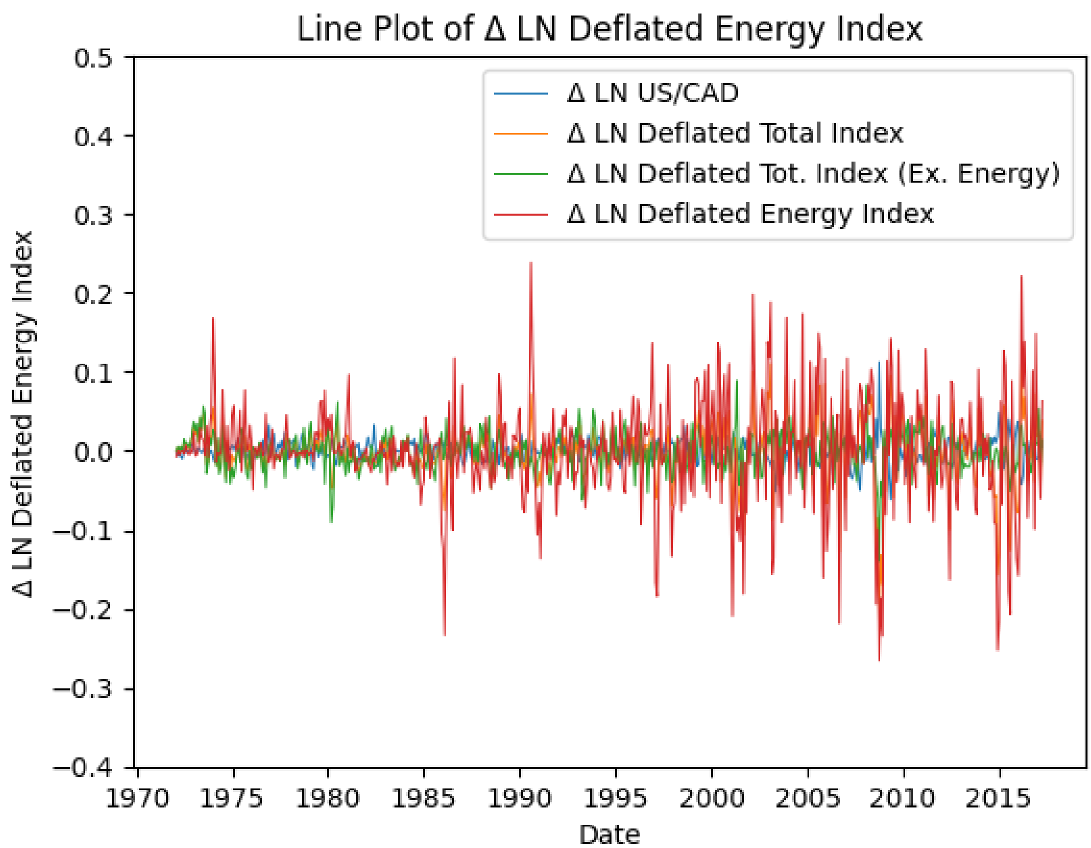
[20]:

```
            Δ LN US/CAD  Δ LN Deflated Total Index  \
Date
1972-02-01    -0.001332                  -0.000862
1972-03-01    -0.006179                   0.004530
1972-04-01    -0.002810                  -0.001424
1972-05-01    -0.006984                   0.004486
1972-06-01    -0.009441                  -0.000440
…                   …                          …
2016-12-01    -0.008118                   0.071923
2017-01-01    -0.010441                   0.018581
2017-02-01    -0.006160                   0.025316
2017-03-01     0.020956                  -0.041645
2017-04-01     0.004206                   0.038115

            Δ LN Deflated Tot. Index (Ex. Energy)  Δ LN Deflated Energy Index
Date
1972-02-01                               0.000133                   -0.006856
1972-03-01                               0.005510                    0.000583
1972-04-01                              -0.000440                   -0.005414
1972-05-01                               0.005444                   -0.000405
1972-06-01                              -0.000448                   -0.000403
…                                          …                           …
2016-12-01                               0.006919                    0.148871
2017-01-01                               0.011434                    0.030169
2017-02-01                               0.053910                   -0.002931
2017-03-01                              -0.022372                   -0.060651
2017-04-01                               0.013098                    0.063253

[543 rows x 4 columns]
```

[21]:

```python
# Data plots for differenced variables
for col in differenced_data.columns:
    plt.plot(differenced_data[col], linewidth=0.7, label=col)

plt.xlabel('Date')
```

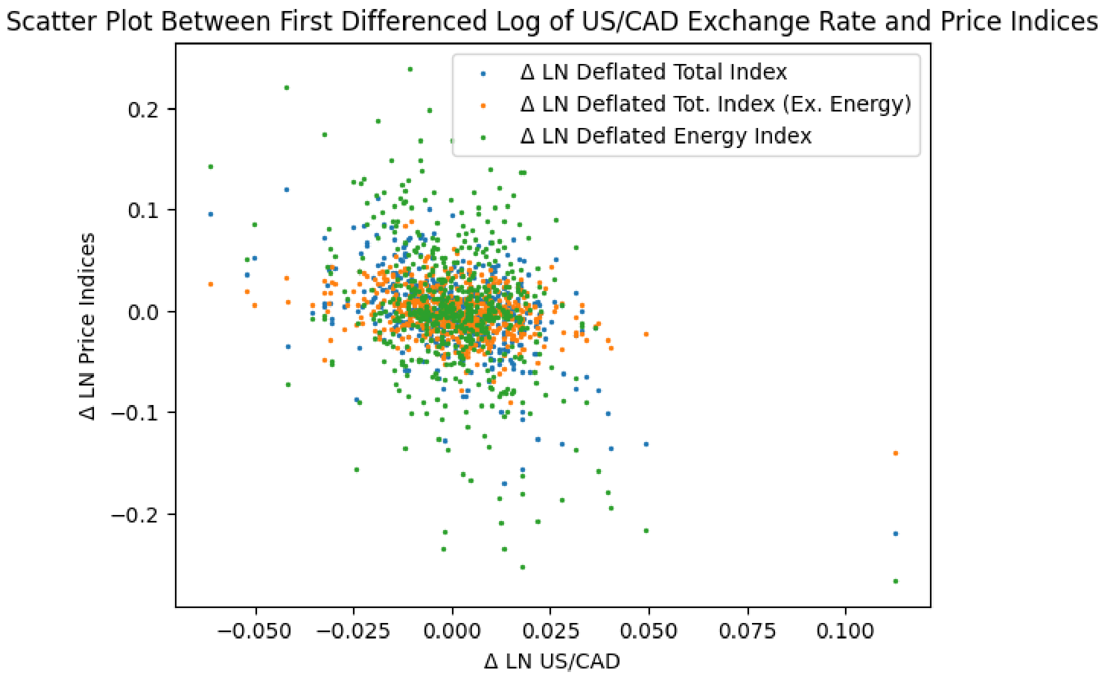
```
plt.ylabel(f'{col}')
plt.title(f'Line Plot of {col}')
plt.legend()
plt.ylim(-0.4,0.5)
plt.show()
```



[22]:

```
# Scatter plot between first differenced log of exchange rate and the various␣
↪price indices
for col in differenced_data:
    if col != 'Δ LN US/CAD':
        plt.scatter(differenced_data['Δ LN US/CAD'],differenced_data[col],␣
↪marker='o',s=2,label=col)

plt.xlabel('Δ LN US/CAD')
plt.ylabel('Δ LN Price Indices')
plt.legend()

plt.title('Scatter Plot Between First Differenced Log of US/CAD Exchange Rate␣
↪and Price Indices')
```

```
plt.show()
```

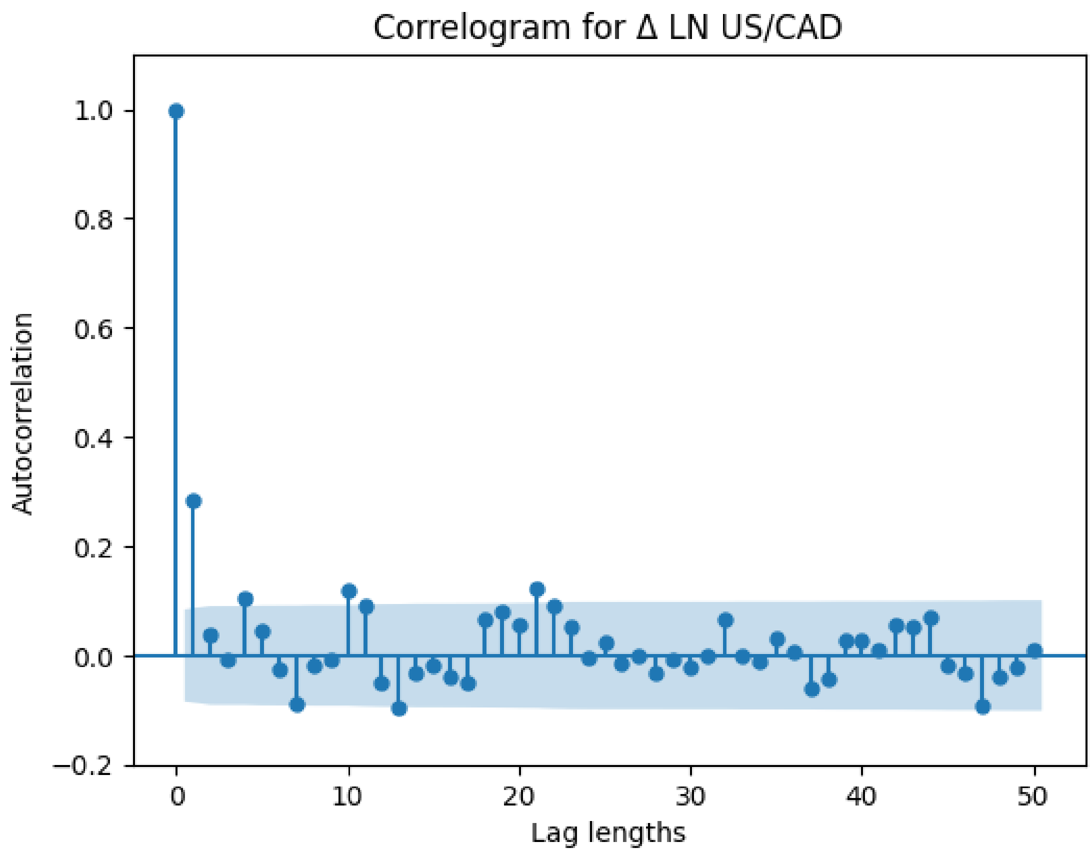


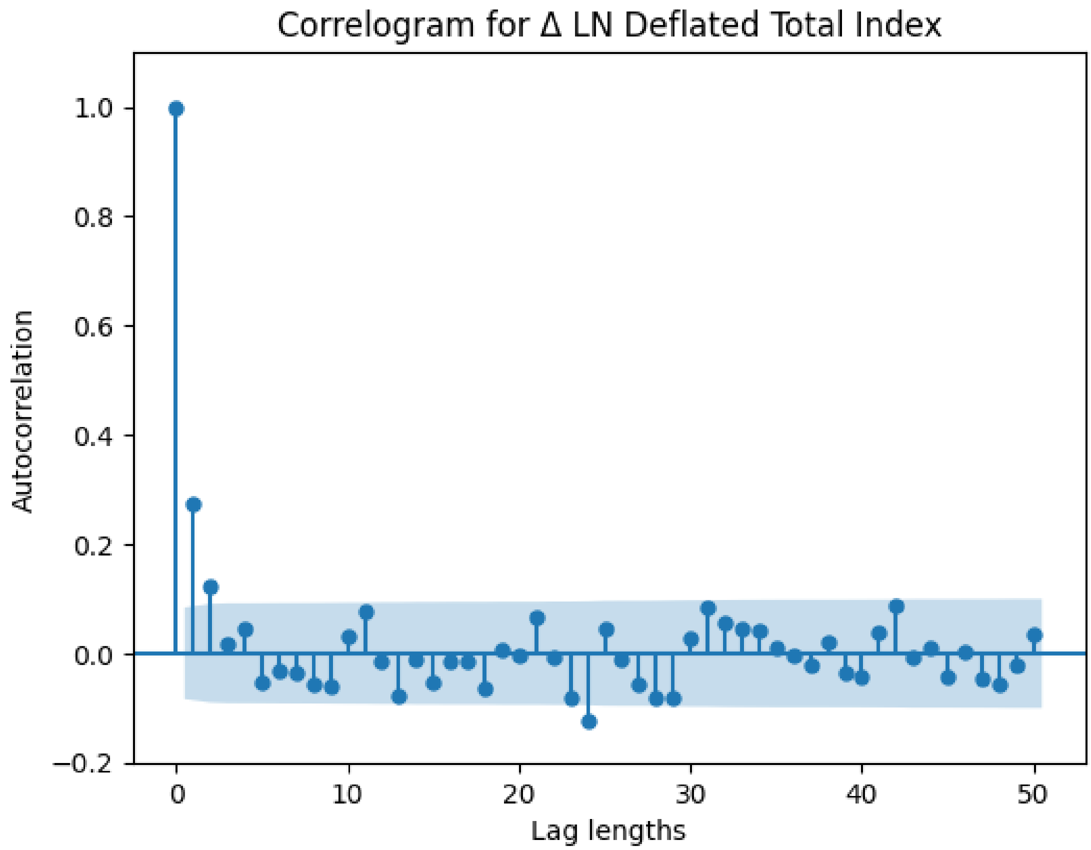
[23]:
```
# Correlation between first differenced log of exchange rate and the various␣
↪price indices
correlations = []
p_values = []

for col in differenced_data.columns:
    if col != 'Δ LN US/CAD':
        corr, p_value = pearsonr(differenced_data[col],differenced_data['Δ LN␣
↪US/CAD'])
        correlations.append(corr)
        p_values.append(p_value)

correlation_table2 = pd.DataFrame({'Correlation':correlations,'P-Value':
↪p_values},
                                 index=['Δ LN Deflated Total Index','Δ LN␣
↪Deflated Tot. Index (Ex. Energy)','Δ LN Deflated Energy Index'])
correlation_table2
```

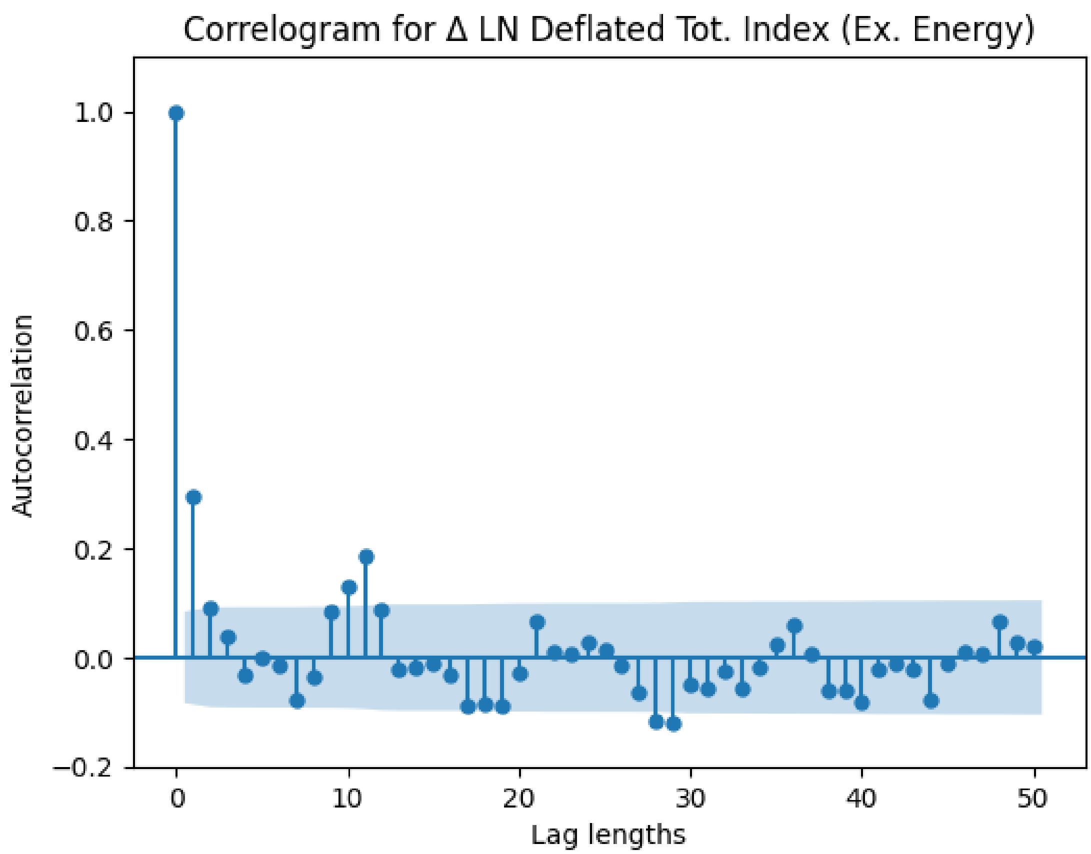
[23]:

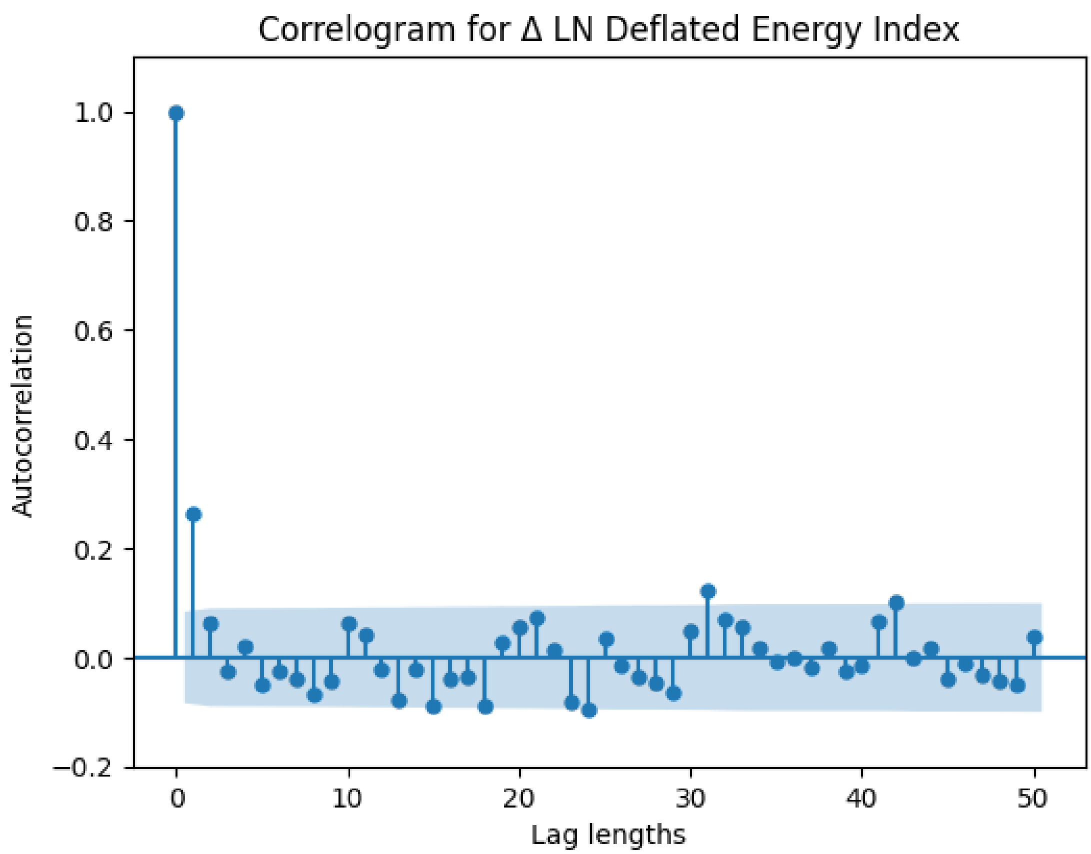
	Correlation	P-Value
Δ LN Deflated Total Index	-0.443436	1.458803e-27
Δ LN Deflated Tot. Index (Ex. Energy)	-0.330451	2.664313e-15
Δ LN Deflated Energy Index	-0.347970	6.696740e-17

[24]:
```
# Autocorrelation function for first differenced log of exchange rate and the
↪various price indices
for col in differenced_data.columns:
  plot_acf(differenced_data[col], lags=50)
  plt.title(f'Correlogram for {col}')
  plt.xlabel('Lag lengths')
  plt.ylabel('Autocorrelation')
  plt.ylim(-0.2,1.1)
  plt.show()
```




Correlogram for Δ LN Deflated Total Index
Autocorrelation
Lag lengths
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
0
10
20
30
40
50


Correlogram for Δ LN Deflated Tot. Index (Ex. Energy)
Autocorrelation
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
0
10
20
30
40
50
Lag lengths



[25]:

```
# Augmented Dickey-Fuller Test for first differenced log of exchange rate and␣
↪the various price indices
test_statistic= []
p_value = []
lag_order = []
for col in differenced_data.columns:
    test_result = adfuller(differenced_data[col], regression='ct', autolag =␣
↪'BIC')
    test_stat, p_val, lag = test_result[:3]
    test_statistic.append(test_stat)
    p_value.append(p_val)
    lag_order.append(lag)

adf_table2 = pd.DataFrame({'Test Statistic': test_statistic,
                                     'P-value': p_value,
                                     'Optimal Lags': lag_order},
                                    index = ['Δ LN US/CAD','Δ LN Deflated Total␣
↪Index','Δ LN Deflated Tot. Index (Ex. Energy)','Δ LN Deflated Energy Index'])
```

```
adf_table2
```

[25]:

	Test Statistic	P-value	Optimal Lags
Δ LN US/CAD	-17.318583	0.0	0
Δ LN Deflated Total Index	-17.483749	0.0	0
Δ LN Deflated Tot. Index (Ex. Energy)	-17.140076	0.0	0
Δ LN Deflated Energy Index	-17.746822	0.0	0

[26]:

```
# Engel-Granger cointegration test
test_statistics = []
p_values = []


for col in logged_deflated_data.columns:
    if col != 'LN US/CAD':
        test_stat,p_val,crit_val =␣
↪coint(logged_deflated_data[col],logged_deflated_data['LN US/
↪CAD'],autolag='AIC',trend='ct' )
        test_statistics.append(test_stat)
        p_values.append(p_val)

coint_table = pd.DataFrame({'Test Statistic': test_statistics,
                            'P-value': p_values,},
                            index=['LN Deflated Total Index','LN␣
↪Deflated Tot. Index (Ex. Energy)','LN Deflated Energy Index'])

coint_table
```

[26]:

	Test Statistic	P-value
LN Deflated Total Index	-3.594558	0.079727
LN Deflated Tot. Index (Ex. Energy)	-4.008835	0.026749
LN Deflated Energy Index	-2.945591	0.288001

[27]:

```
# VAR model1 optimal lags
model1 = VAR(differenced_data[['Δ LN US/CAD','Δ LN Deflated Total Index']])
x1 = model1.select_order(maxlags=5)
x1.summary()
```

```
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473:
ValueWarning: No frequency information was provided, so inferred frequency MS
will be used.
  self._init_dates(dates, freq)
```

[27]:

	AIC	BIC	FPE	HQIC
0	-15.25	-15.24	2.376e-07	-15.25
1	-15.38*	-15.33*	2.098e-07*	-15.36*
2	-15.38	-15.30	2.100e-07	-15.35
3	-15.37	-15.26	2.120e-07	-15.32
4	-15.37	-15.23	2.116e-07	-15.31
5	-15.36	-15.19	2.130e-07	-15.29

[28]:
```
# VAR model2 optimal lags
model2 = VAR(differenced_data[['Δ LN US/CAD','Δ LN Deflated Energy Index']])
x2 = model2.select_order(maxlags=5)
x2.summary()
```

```
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473:
ValueWarning: No frequency information was provided, so inferred frequency MS
will be used.
  self._init_dates(dates, freq)
```

[28]:

	AIC	BIC	FPE	HQIC
0	-13.99	-13.98	8.387e-07	-13.99
1	-14.12*	-14.07*	7.390e-07*	-14.10*
2	-14.12	-14.04	7.405e-07	-14.08
3	-14.11	-13.99	7.486e-07	-14.06
4	-14.11	-13.96	7.465e-07	-14.05
5	-14.10	-13.92	7.533e-07	-14.03

[29]:
```
# VECM model optimal lags
model3 = VECM(logged_deflated_data[['LN US/CAD','LN Deflated Tot. Index (Ex.␣
↪Energy)']])
x3 = select_order(logged_deflated_data[['LN US/CAD','LN Deflated Tot. Index (Ex.
↪ Energy)']], maxlags=5)
x3.summary()
```

```
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473:
ValueWarning: No frequency information was provided, so inferred frequency MS
will be used.
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473:
ValueWarning: No frequency information was provided, so inferred frequency MS
will be used.
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473:
ValueWarning: No frequency information was provided, so inferred frequency MS
will be used.
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473:
ValueWarning: No frequency information was provided, so inferred frequency MS
```

```
will be used.
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473:
ValueWarning: No frequency information was provided, so inferred frequency MS
will be used.
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473:
ValueWarning: No frequency information was provided, so inferred frequency MS
will be used.
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473:
ValueWarning: No frequency information was provided, so inferred frequency MS
will be used.
  self._init_dates(dates, freq)
```

[29]:

	AIC	BIC	FPE	HQIC
0	-16.01	-15.96	1.114e-07	-15.99
1	-16.16*	-16.08*	9.592e-08*	-16.13*
2	-16.15	-16.04	9.714e-08	-16.10
3	-16.13	-15.99	9.841e-08	-16.08
4	-16.15	-15.98	9.654e-08	-16.08
5	-16.14	-15.93	9.781e-08	-16.06

[30]:

```python
# Regression results for VAR model1
fitted_model1 = model1.fit(1)
fitted_model1.summary()
```

[30]:

```
  Summary of Regression Results
==================================
Model:                         VAR
Method:                        OLS
Date:           Wed, 28, Feb, 2024
Time:                     13:30:51
--------------------------------------------------------------------
No. of Equations:         2.00000    BIC:                   -15.3430
Nobs:                     542.000    HQIC:                  -15.3720
Log likelihood:           2638.71    FPE:                2.06996e-07
AIC:                     -15.3906    Det(Omega_mle):     2.04724e-07
--------------------------------------------------------------------
Results for equation Δ LN US/CAD
================================================================================
===============
                                  coefficient       std. error           t-stat
prob
--------------------------------------------------------------------------------
---------------
const                                0.000381         0.000598            0.638
```

```
0.524
L1.Δ LN US/CAD                          0.243684         0.045912            5.308
0.000
L1.Δ LN Deflated Total Index           -0.036608         0.018013           -2.032
0.042
================================================================================
===============

Results for equation Δ LN Deflated Total Index
================================================================================
===============
                                     coefficient       std. error           t-stat
prob
--------------------------------------------------------------------------------
---------------
const                                  -0.000414         0.001533           -0.270
0.787
L1.Δ LN US/CAD                         -0.157715         0.117741           -1.340
0.180
L1.Δ LN Deflated Total Index            0.248064         0.046194            5.370
0.000
================================================================================
===============

Correlation matrix of residuals
                             Δ LN US/CAD  Δ LN Deflated Total Index
Δ LN US/CAD                     1.000000                  -0.410141
Δ LN Deflated Total Index      -0.410141                   1.000000
```

[31]:

```
# Tests for model1
jb_test = jarque_bera(fitted_model1.resid)
print("Jarque-Bera test results:")
print("Statistic:", jb_test[0])
print("p-value:", jb_test[1])

granger_test = grangercausalitytests(differenced_data[['Δ LN US/CAD','Δ LN␣
 ↪Deflated Total Index']],1)
print('\nGranger Causality Test')
print(granger_test)

print('\nModel stability test')
print(fitted_model1.is_stable(verbose=True))

ljun_box = fitted_model1.test_whiteness()
print('\n',ljun_box)
```

```
Jarque-Bera test results:
Statistic: [1226.17371851  292.57850503]
p-value: [5.49237395e-267 2.93349175e-064]

Granger Causality
number of lags (no zero) 1
ssr based F test:         F=4.1302  , p=0.0426  , df_denom=539, df_num=1
ssr based chi2 test:   chi2=4.1532  , p=0.0416  , df=1
likelihood ratio test: chi2=4.1374  , p=0.0419  , df=1
parameter F test:         F=4.1302  , p=0.0426  , df_denom=539, df_num=1

Granger Causality Test
{1: ({'ssr_ftest': (4.130224392021003, 0.04261353422801978, 539.0, 1),
'ssr_chi2test': (4.153212653943198, 0.0415556094973795, 1), 'lrtest':
(4.13738095539793, 0.0419460203454895, 1), 'params_ftest': (4.1302243920209305,
0.04261353422801978, 539.0, 1.0)},
[<statsmodels.regression.linear_model.RegressionResultsWrapper object at
0x7e74a3f1ace0>, <statsmodels.regression.linear_model.RegressionResultsWrapper
object at 0x7e74a3f1add0>, array([[0., 1., 0.]])])}

Model stability test
Eigenvalues of VAR(1) rep
0.16985788026909565
0.321889722760335
True

 <statsmodels.tsa.vector_ar.hypothesis_test_results.WhitenessTestResults object.
H_0: residual autocorrelation up to lag 10 is zero: reject at 5% significance
level. Test statistic: 60.394, critical value: 50.998>, p-value: 0.007>
```

[32]:
```python
# Regression results for VAR model2
fitted_model2 = model2.fit(1)
fitted_model2.summary()
```

[32]:
```
  Summary of Regression Results
==================================
Model:                         VAR
Method:                        OLS
Date:           Wed, 28, Feb, 2024
Time:                     13:30:51
--------------------------------------------------------------------
No. of Equations:         2.00000    BIC:                   -14.0836
Nobs:                     542.000    HQIC:                  -14.1126
Log likelihood:           2297.42    FPE:                7.29291e-07
AIC:                     -14.1312    Det(Omega_mle):     7.21284e-07
--------------------------------------------------------------------
Results for equation Δ LN US/CAD
```

```
================================================================================
================
                                  coefficient       std. error             t-stat
prob
--------------------------------------------------------------------------------
----------------
const                                0.000408         0.000599              0.681
0.496
L1.Δ LN US/CAD                       0.264653         0.043979              6.018
0.000
L1.Δ LN Deflated Energy Index       -0.012834         0.009603             -1.337
0.181
================================================================================
================

Results for equation Δ LN Deflated Energy Index
================================================================================
================
                                  coefficient       std. error             t-stat
prob
--------------------------------------------------------------------------------
----------------
const                                0.000829         0.002767              0.300
0.764
L1.Δ LN US/CAD                      -0.230101         0.203032             -1.133
0.257
L1.Δ LN Deflated Energy Index        0.245019         0.044331              5.527
0.000
================================================================================
================

Correlation matrix of residuals
                             Δ LN US/CAD  Δ LN Deflated Energy Index
Δ LN US/CAD                     1.000000                   -0.321709
Δ LN Deflated Energy Index     -0.321709                    1.000000
```

[33]:

```python
# Tests for model2
jb_test = jarque_bera(fitted_model2.resid)
print("Jarque-Bera test results:")
print("Statistic:", jb_test[0])
print("p-value:", jb_test[1])

granger_test = grangercausalitytests(differenced_data[['Δ LN US/CAD','Δ LN␣
↪Deflated Energy Index']],1)
print('\nGranger Causality Test')
```

```python
print(granger_test)

print('\nModel stability test')
print(fitted_model2.is_stable(verbose=True))

ljung_box = fitted_model2.test_whiteness()
print('\n',ljung_box)
```

```
Jarque-Bera test results:
Statistic: [1344.27447435   94.84760443]
p-value: [1.24310214e-292 2.53573805e-021]

Granger Causality
number of lags (no zero) 1
ssr based F test:         F=1.7862  , p=0.1819  , df_denom=539, df_num=1
ssr based chi2 test:   chi2=1.7962  , p=0.1802  , df=1
likelihood ratio test: chi2=1.7932  , p=0.1805  , df=1
parameter F test:         F=1.7862  , p=0.1819  , df_denom=539, df_num=1

Granger Causality Test
{1: ({'ssr_ftest': (1.7862474772699735, 0.18194793197867135, 539.0, 1),
'ssr_chi2test': (1.7961894854922553, 0.18017385100004263, 1), 'lrtest':
(1.7932197563713999, 0.18053435939115428, 1), 'params_ftest':
(1.7862474772700203, 0.18194793197867135, 539.0, 1.0)},
[<statsmodels.regression.linear_model.RegressionResultsWrapper object at
0x7e74a80e9e40>, <statsmodels.regression.linear_model.RegressionResultsWrapper
object at 0x7e74a80a60b0>, array([[0., 1., 0.]])])}

Model stability test
Eigenvalues of VAR(1) rep
0.31005823798332793
0.19961408057675423
True

 <statsmodels.tsa.vector_ar.hypothesis_test_results.WhitenessTestResults object.
H_0: residual autocorrelation up to lag 10 is zero: reject at 5% significance
level. Test statistic: 55.858, critical value: 50.998>, p-value: 0.018>
```

[34]:
```python
# Regression results for VECM model
fitted_model3 = model3.fit()
fitted_model3.summary()
```

[34]:

	coef	std err	z	P
L1.LN US/CAD	0.2635	0.043	6.068	
L1.LN Deflated Tot. Index (Ex. Energy)	-0.0410	0.026	-1.563	
	coef	**std err**	**z**	**P**
L1.LN US/CAD.LN Deflated Tot. Index (Ex	-0.0731	0.072	-1.018	
L1.LN Deflated Tot. Index (Ex. Energy).LN Deflated Tot. Index (Ex	0.2821	0.043	6.495	
	coef	**std err**	**z**	**P**
ec1	-0.0038	0.002	-1.739	
	coef	**std err**	**z**	**P**
ec1.LN Deflated Tot. Index (Ex	0.0041	0.004	1.127	
	coef	**std err**	**z**	**P**
beta.1	1.0000	0	0	
beta.2	-0.5095	0.231	-2.210	

[35]:
```
# Tests for model3
jb_test = jarque_bera(fitted_model3.resid)
print("Jarque-Bera test results:")
print("Statistic:", jb_test[0])
print("p-value:", jb_test[1])

granger_test = grangercausalitytests(logged_deflated_data[['LN US/CAD','LN␣
 ↪Deflated Tot. Index (Ex. Energy)']],1)
print('Granger Causality Test')
print(granger_test)

ljung_box = fitted_model3.test_whiteness()
print('\n',ljung_box)
```

```
Jarque-Bera test results:
Statistic: [1229.73021182   77.19655659]
p-value: [9.27849181e-268 1.72576130e-017]

Granger Causality
number of lags (no zero) 1
ssr based F test:         F=0.0137  , p=0.9069  , df_denom=540, df_num=1
ssr based chi2 test:   chi2=0.0138  , p=0.9066  , df=1
likelihood ratio test: chi2=0.0138  , p=0.9066  , df=1
parameter F test:         F=0.0137  , p=0.9069  , df_denom=540, df_num=1
Granger Causality Test
{1: ({'ssr_ftest': (0.013676005090602884, 0.9069475518095418, 540.0, 1),
'ssr_chi2test': (0.013751982896661791, 0.9066470097385542, 1), 'lrtest':
(0.0137518087594799, 0.9066475980863145, 1), 'params_ftest':
(0.013676005090585162, 0.9069475518095418, 540.0, 1.0)},
[<statsmodels.regression.linear_model.RegressionResultsWrapper object at
0x7e74a3f6e560>, <statsmodels.regression.linear_model.RegressionResultsWrapper
```

```
object at 0x7e74a3f6f040>, array([[0., 1., 0.]])])}

 <statsmodels.tsa.vector_ar.hypothesis_test_results.WhitenessTestResults object.
H_0: residual autocorrelation up to lag 10 is zero: reject at 5% significance
level. Test statistic: 56.112, critical value: 48.602>, p-value: 0.010>
```

[36]:
```
# Data frame of commodity prices of interest
prices = merged_data.iloc[:,4:].copy()
logged_prices = np.log(prices.copy())
logged_prices.columns = [f'LN {col}' for col in logged_prices.columns]
logged_prices
```

[36]:
```
            LN Energy Index  LN Metals & Minerals Index  LN Agriculture Index  \
Date
1972-01-01         4.605170                    4.605170              4.605170
1972-02-01         4.603168                    4.612146              4.617099
1972-03-01         4.606170                    4.619073              4.629863
1972-04-01         4.603168                    4.617099              4.625953
1972-05-01         4.605170                    4.618086              4.639572
…                       …                          …                     …
2016-12-01         6.824265                    6.204356              5.334649
2017-01-01         6.860244                    6.216406              5.358471
2017-02-01         6.860454                    6.291754              5.383118
2017-03-01         6.800615                    6.246688              5.364573
2017-04-01         6.866829                    6.261683              5.364573

            LN Fish Index  LN Forestry Index
Date
1972-01-01       4.605170           4.605170
1972-02-01       4.487512           4.606170
1972-03-01       4.595120           4.607168
1972-04-01       4.635699           4.614130
1972-05-01       4.457830           4.627910
…                     …                  …
2016-12-01       7.122625           5.878016
2017-01-01       7.192859           5.887492
2017-02-01       7.216122           5.965377
2017-03-01       7.253470           5.975081
2017-04-01       7.261717           6.018593

[544 rows x 5 columns]
```

[37]:
```
logged_prices.describe()
```

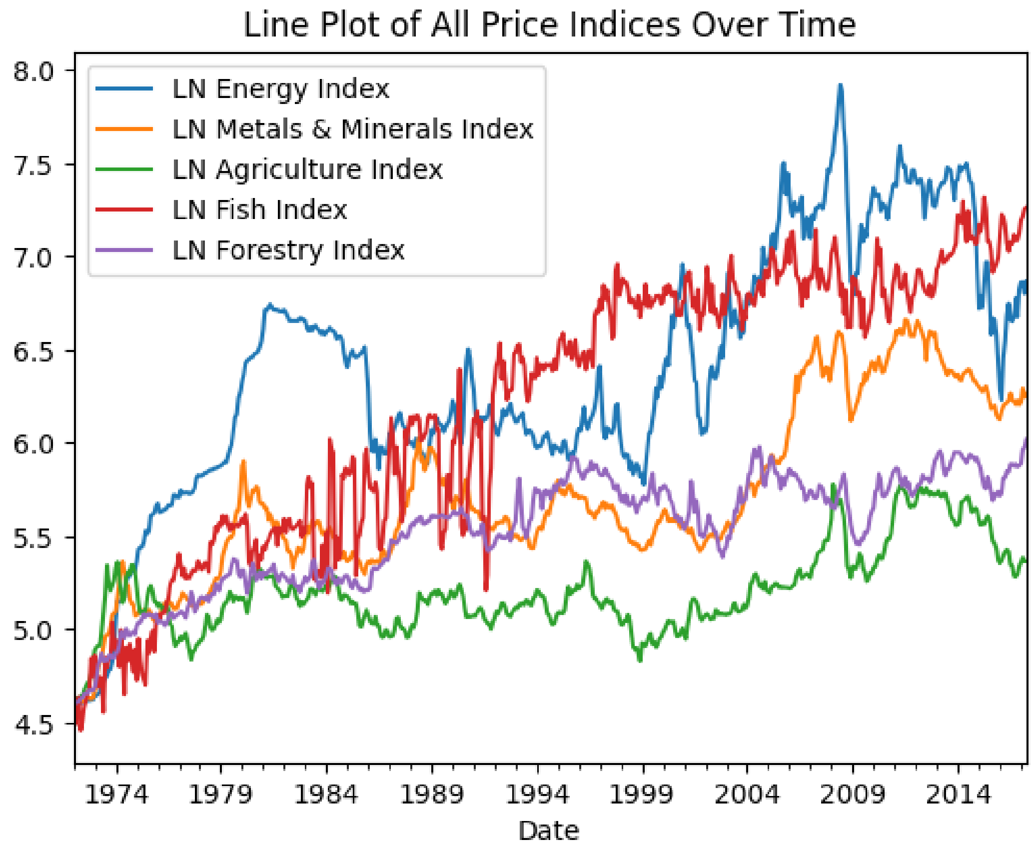
[37]:
```
       LN Energy Index  LN Metals & Minerals Index  LN Agriculture Index  \
count       544.000000                  544.000000            544.000000
mean          6.387978                    5.709536              5.205565
```

```
std          0.685754                    0.467090                0.225740
min          4.603168                    4.605170                4.605170
25%          5.984692                    5.437426                5.070475
50%          6.346162                    5.607448                5.170484
75%          6.848954                    6.046223                5.295062
max          7.921463                    6.659166                5.777343

       LN Fish Index  LN Forestry Index
count     544.000000         544.000000
mean        6.203912           5.519041
std         0.742417           0.313381
min         4.457830           4.605170
25%         5.543320           5.285989
50%         6.444527           5.596383
75%         6.828035           5.770662
max         7.315618           6.018593
```

[38]:
```python
logged_prices.corr()
```

[38]:
```
                            LN Energy Index  LN Metals & Minerals Index  \
LN Energy Index                    1.000000                    0.867618
LN Metals & Minerals Index         0.867618                    1.000000
LN Agriculture Index               0.728980                    0.822192
LN Fish Index                      0.720196                    0.761381
LN Forestry Index                  0.695765                    0.776815

                            LN Agriculture Index  LN Fish Index  \
LN Energy Index                         0.728980       0.720196
LN Metals & Minerals Index              0.822192       0.761381
LN Agriculture Index                    1.000000       0.493036
LN Fish Index                           0.493036       1.000000
LN Forestry Index                       0.522904       0.910223

                            LN Forestry Index
LN Energy Index                      0.695765
LN Metals & Minerals Index           0.776815
LN Agriculture Index                 0.522904
LN Fish Index                        0.910223
LN Forestry Index                    1.000000
```

[39]:
```python
# Line plot of all variables
logged_prices[['LN Energy Index', 'LN Metals & Minerals Index','LN Agriculture␣
↪Index','LN Fish Index', 'LN Forestry Index']].plot()
plt.title('Line Plot of All Price Indices Over Time')
plt.show()
```



[40]:
```python
# Augmented Dickey Fuller test for all variables
test_statistic= []
p_value = []
lag_order = []
for col in logged_prices.columns:
    test_result = adfuller(logged_prices[col], regression='ct', autolag = 'BIC')
    test_stat, p_val, lag = test_result[:3]
    test_statistic.append(test_stat)
    p_value.append(p_val)
    lag_order.append(lag)

adf_table3 = pd.DataFrame({'Test Statistic': test_statistic,
                                     'P-value': p_value,
                                     'Optimal Lags': lag_order},
                                    index = ['LN Energy Index', 'LN Metals &␣
↪Minerals Index','LN Agriculture Index','LN Fish Index', 'LN Forestry Index'])

adf_table3
```

[40]:

```
                            Test Statistic   P-value  Optimal Lags
LN Energy Index                  -2.791964  0.199912             1
LN Metals & Minerals Index       -2.564600  0.296452             1
LN Agriculture Index             -3.651722  0.025740             1
LN Fish Index                    -2.893394  0.164361            14
LN Forestry Index                -3.902814  0.012025             1
```

[41]:

```
logged_prices2 = logged_prices.copy().diff().dropna()
logged_prices2['LN Agriculture Index'] = logged_prices['LN Agriculture Index'].
 ↪copy().iloc[1:]
logged_prices2['LN Forestry Index'] = logged_prices['LN Forestry Index'].copy().
 ↪iloc[1:]
logged_prices2.columns = [f'Δ {col}' if col not in ['LN Agriculture Index','LN␣
 ↪Forestry Index'] else col for col in logged_prices2.columns]
logged_prices2
```

[41]:

```
            Δ LN Energy Index  Δ LN Metals & Minerals Index  \
Date
1972-02-01          -0.002002                      0.006976
1972-03-01           0.003002                      0.006927
1972-04-01          -0.003002                     -0.001974
1972-05-01           0.002002                      0.000988
1972-06-01           0.001998                     -0.003956
…                         …                           …
2016-12-01           0.149199                     -0.024155
2017-01-01           0.035979                      0.012051
2017-02-01           0.000210                      0.075348
2017-03-01          -0.059839                     -0.045066
2017-04-01           0.066214                      0.014995

            LN Agriculture Index  Δ LN Fish Index  LN Forestry Index
Date
1972-02-01              4.617099        -0.117658           4.606170
1972-03-01              4.629863         0.107608           4.607168
1972-04-01              4.625953         0.040580           4.614130
1972-05-01              4.639572        -0.177870           4.627910
1972-06-01              4.640537         0.049728           4.633758
…                            …               …                  …
2016-12-01              5.334649         0.034967           5.878016
2017-01-01              5.358471         0.070234           5.887492
2017-02-01              5.383118         0.023263           5.965377
2017-03-01              5.364573         0.037348           5.975081
2017-04-01              5.364573         0.008246           6.018593

[543 rows x 5 columns]
```

[42]:
```python
# Augmented Dickey Fuller test for all variables
test_statistic= []
p_value = []
lag_order = []
for col in logged_prices2.columns:
    test_result = adfuller(logged_prices2[col], regression='ct', autolag =␣
↪'BIC')
    test_stat, p_val, lag = test_result[:3]
    test_statistic.append(test_stat)
    p_value.append(p_val)
    lag_order.append(lag)

adf_table4 = pd.DataFrame({'Test Statistic': test_statistic,
                                     'P-value': p_value,
                                     'Optimal Lags': lag_order},
                                    index = ['Δ LN Energy Index', 'Δ LN Metals␣
↪& Minerals Index','LN Agriculture Index','Δ LN Fish Index', 'LN Forestry␣
↪Index'])

adf_table4
```

[42]:

	Test Statistic	P-value	Optimal Lags
Δ LN Energy Index	-17.433031	0.000000e+00	0
Δ LN Metals & Minerals Index	-17.371912	0.000000e+00	0
LN Agriculture Index	-3.645423	2.621236e-02	1
Δ LN Fish Index	-6.427575	2.659068e-07	13
LN Forestry Index	-3.942903	1.058167e-02	1

[43]:
```python
# Johansen cointegration test
data = logged_prices2.values

result = coint_johansen(data, det_order=0, k_ar_diff=1)
critical_values = result.cvt[:, 1]

eigenvalues = result.eig
eigenvectors = result.evec
trace = result.lr1
# Print the results
print("Eigenvalues:", eigenvalues)
print('Trace Statistics', trace)
print("Critical Values:", critical_values)
print("Eigenvectors:", eigenvectors)
```

```
Eigenvalues: [0.37396032 0.29831121 0.26685735 0.02003953 0.01097993]
Trace Statistics [629.88927991 376.51651155 184.85898849  16.92448138
5.9729954 ]
Critical Values: [69.8189 47.8545 29.7961 15.4943  3.8415]
```

```
Eigenvectors: [[ 1.32820739e+00  1.50571991e+01  1.19907535e+01  3.64970263e-01
  -3.43892417e-01]
 [-7.42549437e+00 -2.96897364e+01  2.17801358e+01 -3.18383939e-01
  -4.87826359e-01]
 [-1.50946054e-01 -1.99599609e-01  3.04741352e-01 -4.71163184e+00
   2.26463723e+00]
 [-1.22830533e+01  2.71929416e+00 -1.49795528e+00 -4.22220082e-01
  -3.77229179e-02]
 [ 1.88475315e-02  9.19652027e-02  4.73057257e-02  3.32926219e-01
  -3.76965780e+00]]
```