1 Achieving the Paris Agreement Goals: Projecting Energy Production Sources in the United States With Time Series Modeling

- Student name: Greg Osborne
- Student pace: self paced / part time
- Scheduled project review date/time: 4/28/23, 2:45 PM
- Instructor name: Morgan Jones
- Blog post URL: https://medium.com/@gregosborne (https://medium.com/@gregosborne)

2 Business Understanding

2.1 Stakeholder: Environmental Protection Agency (EPA) (https://www.epa.gov/)

2.2 Overview

The United States needs to achieve Paris Agreement greenhouse gas reduction goals in the energy production sector by 2050. The nation must understand current trends in the sector to see if EPA intervention is required to meet the targets. This project projects current energy production trends in the United States and recommends changes to achieve the Paris Agreement reduction goals.

2.3 Background:

On December 12, 2015, the historic Paris Agreement (https://apnews.com/article/climate-climate-change-john-kerry-paris-archive-81dabae32cb8463b86bd85d762da9e6d) legally bound the United States and other nations of the world to limit global warming to well below 2 °C (3.6 °F) over preindustrial levels.

The Paris Agreement also commissioned the Intergovernmental Panl no Climate Change (IPCC (https://www.ipcc.ch/)) to review the effects of climate change at 1.5 °C (2.7 °F). The IPCC released its report on the matter in 2018 (https://www.ipcc.ch/sr15/chapter/spm/), detailing the pestilence of a warmer world.

To limit climate change to no greater than 1.5 °C over pre-industrial levels, the IPCC's 2018 special report concluded that greenhouse gas (GHG) emissions must "decline by about 45% from 2010 levels by 2030 (40–60% interquartile range), reaching net zero around 2050." (https://www.ipcc.ch/sr15/chapter/spm/#article-spm-c) Today, in 2023, we are already at 1.1 °C (2 °F) (https://earthobservatory.nasa.gov/world-of-change/global-temperatures). With GHG emissions as they are, scientists anticipate this average to grow by about 0.2 °C per decade unless GHG emissions are cut.

2.4 Buisness Problem:

The United States needs to ensure our energy production sector achieves the Paris Agreement goals.

GHG emissions occur in multiple sectors of the economy. Electricity generation releases about 25% of GHG emissions in the United States (https://www.epa.gov/ghgemissions/sources-greenhouse-gas-emissions), and about 40% globally (https://www.iea.org/data-and-statistics/charts/global-energy-related-co2-emissions-by-sector). The EPA commissioned Greg Osborne to project current energy production trends and recommend policies to help the United States reach the GHG reduction targets set by the IPCC.

Electricity generation refers to power plants creating electricity to distribute across power lines to residential and business locations. Electricity sources include fossil fuel methods, burning coal, oil and natural gas which emit GHG, and sources that do not emit GHG, nuclear, wind turbines, solar panels and hydroelectrical. Reducing America's GHG emissions and exporting that technology to our allies will result in greater reduction of GHG emissions than any other sector.

2.5 Time-Series Modeling

The time series analysis requires projecting two factors into the future:

1. Construction/Reduction Rate of GHG-Emitting Power Generation

Unfortunately, not only are fossil fuels still powering our lifestyles, but power companies in the US still plan to open natural gas power plants in future (https://www.eia.gov/todayinenergy/detail.php?id=50436). Construction of new coal plants has stalled in the United States (https://www.scientificamerican.com/article/will-the-u-s-ever-build-another-big-coal-plant/), but worldwide, several coal-fired power plants are scheduled to be built in the future (https://www.reuters.com/business/energy/cop26-aims-banish-coal-asia-is-building-hundreds-power-plants-burn-it-2021-10-29/). The time series analysis must consider the growth and reduction of fossil-fuel power plants.

2. Construction Rate of Clean Power Generation

Contrary to what the crisis needs, we cannot build a billion wind turbines overnight. We can only build clean energy sources as materials, labor availability, construction time, and politics allow. Fortunately, construction of new renewable and nuclear power production is underway. The time series analysis will model current trends of construction to see if they can meet the The United State's electricity needs as fossil fuels diminish.

I will then compare this time series analysis with two other factors that are not projected with a time series model:

1. Increase in Electricity Demand

America's demand for electricity will rise as we cut emissions across all sectors. The poster-child example of this is the impending fuel source change for automobiles. Car manufacturers representing a quarter of global sales have pledged to cease production of internal-combustion-engine cars (https://www.caranddriver.com/news/a38213848/automakers-pledge-end-gas-sales-2040/) within the IPCC's timeline. The most likely power source for their new cars is electricity, creating greater demand. The data used to estimate the increase in Electricity demand is provided by the International Energy Agency.

2. The Paris Agreement GHG Emissions reduction targets

As part of the Paris Agreement, the IPCC has determined that GHG emissions must "decline by about 45% from 2010 levels by 2030 (40–60% interquartile range), reaching net zero around 2050" (https://www.ipcc.ch/sr15/chapter/spm/#article-spm-c). To simulate this in the electricity production industry, I have defined this reduction as an overall reduction in electricity power generation for each fossil fuel energy production source (coal, oil and natural gas). These reduction targets are visible in the graphs produced in this report.

With all these factors, time-series modeling can assist with projection needs. This project will show the United State's energy production trends, and make recommendations for how to achieve the Paris Agreement Goals.

2.6 Methodology

The planned steps I will follow for this project include:

1. Create a time series models that project energy production trends for each fuel source in the United States to the year 2050.
2. Plot all energy production trends against the projected energy demands of the United States.
3. Using interpolation, determine what reductions and construction increases are necessary to achieve the Paris Agreement goals. This will not use time series, as it's a determination of what trends must accomplish, rather than where the trends will go if left unchecked.

3 Table of Contents

Project 5 – What would it take to Meet the Paris Agreement Energy Production Targets?

4 Python Libraries

In [1]:
```
# DataFrames and computation
import pandas as pd
import numpy as np
import os
import statsmodels.api as sm

# Graphing
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

# To supress warnings
import warnings
warnings.filterwarnings("ignore")
from statsmodels.tools.sm_exceptions import ValueWarning
from statsmodels.tools.sm_exceptions import ConvergenceWarning
warnings.simplefilter('ignore', ValueWarning)
warnings.simplefilter('ignore', ConvergenceWarning)

# For distribution calculations.
import scipy.stats as stats

# For datetime conversions
import datetime as dt

# Decompose to check for seasonality
from statsmodels.tsa.seasonal import seasonal_decompose

# Check for Stationarity
from statsmodels.tsa.stattools import adfuller

# Plot ACF / PACF
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

# Plot with Arima
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.statespace.sarimax import SARIMAX

# For testing and metrics
import sklearn as sk
import statsmodels.api as sm
from sklearn import metrics
from statsmodels.tsa.stattools import adfuller

# For creating iterations
import itertools
# Setting DataFrame Display Settings
pd.set_option("display.max_columns", None)
pd.set_option("display.max_rows", 6)
# Setting Pandas Series Display Settings

# Formatting decimals to show three numbers after the point.
pd.options.display.float_format = '{:,.2f}'.format

# Plotly for a tree map
```

```
import plotly.express as px
```

5 Data Loading From Source, EDA, and Cleaning

For details on the origins of this project's data, Exploratory Data Analysis, and Cleaning, see the Jupyter Notebook file: data_import_cleaning_export.ipynb (https://github.com/FunkyTable/Project-5/blob/main/data_import_cleaning_export.ipynb).

6 Initial Modeling

6.1 Loading the Data

6.1.1 Power Generation Model Data

In [2]:
```
# Loading the DataFrame
df_hist_elec = pd.read_csv(
    "Data/Export/df_hist_elec_yr_reg.csv",
    encoding='ANSI')

# Setting the index to the year and DateTime format
df_hist_elec['Year'] = pd.to_datetime(df_hist_elec.Year, format='%Y')

df_hist_elec.index = df_hist_elec['Year']
df_hist_elec = df_hist_elec.iloc[:,1:]

df_hist_elec
```

Out[2]:

Year	Earth Other Renewables (TWh)	Earth Bioenergy (TWh)	Earth Solar (TWh)	Earth Wind (TWh)	Earth Hydro (TWh)	Earth Nuclear (TWh)	Earth Oil (TWh)	Earth Gas (TWh)	Eart Coa (TWh
2000-01-01	52.37	148.41	1.08	31.16	2,621.47	2,507.43	1,173.34	2,717.70	5,718.9
2001-01-01	52.60	142.90	1.35	38.16	2,561.15	2,573.71	1,157.32	2,867.51	5,801.7
2002-01-01	54.08	156.22	1.69	52.04	2,601.49	2,601.89	1,135.84	3,071.37	6,058.6
...	...	...	...	...	...	...	...	...	.
2019-01-01	91.39	577.97	701.19	1,419.53	4,220.50	2,724.08	738.53	6,208.60	9,684.4
2020-01-01	94.28	605.08	852.10	1,586.92	4,340.61	2,634.69	710.57	6,153.20	9,297.7
2021-01-01	95.65	666.28	1,040.50	1,848.26	4,234.35	2,739.32	764.52	6,337.96	10,085.9

22 rows × 144 columns

6.1.2 Energy Demand Model Data

In [3]:
```
# Loading the Historical energy demand DataFrame
df_hist_dem = pd.read_csv(
    "Data/Export/df_hist_dem_yr_reg.csv",
    encoding='ANSI')

# Setting the index to the year and DateTime format
df_hist_dem['Year'] = pd.to_datetime(df_hist_dem.Year, format='%Y')
df_hist_dem.index = df_hist_dem['Year']
df_hist_dem = df_hist_dem.iloc[:,1:]

df_hist_dem
```

Out[3]:

Year	Earth Historic Demand (TWh)	Asia Pacific Historic Demand (TWh)	North America Historic Demand (TWh)	Europe Historic Demand (TWh)	Eurasia Historic Demand (TWh)	Central & South America Historic Demand (TWh)	Middle East Historic Demand (TWh)	Africa Historic Demand (TWh)	United States Historic Demand (TWh)	D
2000-01-01	14,971.90	3,753.73	4,583.44	1,014.49	987.19	778.62	430.47	423.01	3,835.86	1
2001-01-01	15,196.46	3,922.35	4,500.90	1,022.65	996.68	768.05	455.59	438.41	3,749.60	1
2002-01-01	15,733.31	4,176.54	4,632.77	1,031.26	1,014.12	792.29	487.66	463.28	3,865.21	1
...	...	...	...	...	...	...	...	...	...	
2019-01-01	26,366.21	11,517.82	5,097.23	1,197.71	1,303.09	1,293.64	1,149.14	827.63	4,197.42	7
2020-01-01	26,275.23	11,708.52	4,986.52	1,178.12	1,295.58	1,291.84	1,142.57	806.57	4,090.49	7
2021-01-01	27,812.74	12,667.08	5,118.78	1,232.56	1,364.76	1,362.57	1,211.87	839.32	4,191.53	8

22 rows × 16 columns

In [4]:
```
# Projected energy demand, Stated Policies
df_proj_dem_stat = pd.read_csv(
    "Data/Export/df_proj_dem_stat_yr.csv",
    encoding='ANSI')

# Setting the index to the year and DateTime format
df_proj_dem_stat['Year'] = pd.to_datetime(df_proj_dem_stat.Year, format='%Y')
df_proj_dem_stat.index = df_proj_dem_stat['Year']
df_proj_dem_stat = df_proj_dem_stat.iloc[:,2:]

# Dropping the 2010 data and adding the historic demand for 2019 for graphing
# purposes.
df_proj_dem_stat.drop(index = df_proj_dem_stat.index[0:3],inplace = True)

jnk = df_hist_dem.iloc[-1:,:]
jnk.columns = df_proj_dem_stat.columns
df_proj_dem_stat = pd.concat([jnk, df_proj_dem_stat], axis = 0)

df_proj_dem_stat
```

Out[4]:

Year	World Projected Demand (TWh)	Asia Pacific Projected Demand (TWh)	North America Projected Demand (TWh)	Europe Projected Demand (TWh)	Eurasia Projected Demand (TWh)	Central and South America Projected Demand (TWh)	Middle East Projected Demand (TWh)	Africa Projected Demand (TWh)	Pro D
2021-01-01	27,812.74	12,667.08	5,118.78	1,232.56	1,364.76	1,362.57	1,211.87	839.32	4,
2030-01-01	34,833.60	18,370.70	5,771.40	4,691.24	1,539.56	1,605.23	1,651.34	1,204.15	4,
2050-01-01	49,844.90	26,573.20	7,815.80	5,703.31	1,937.40	2,591.74	2,886.16	2,337.29	6,

6.1.3 Pruning the Data to Just USA

In [5]:
```
# I originally thought I would perform this analysis on the entire world, but
# I settled for just the United States. I left this here in case I ever return
# to the broader scope of this project.
# Creating lists of column names to easily call on different types of data.

# Energy Types including all regions and countries.
oth = []
bio = []
sol = []
wnd = []
hyd = []
nuc = []
oil = []
gas = []
coal = []

# Energy Types including all regions.
ear = []
apc = []
nam = []
erp = []
eur = []
csa = []
mde = []
afc = []

# Energy Types including all countries.
usa = []
chn = []
eu =  []
jpn = []
rus = []
ind = []
sea = []
bra = []
```

In [6]:
```
# Filling the lists with the correct column names to easily call on
# different organizations of data.
for col in df_hist_elec.columns:
    # Organized by Fuel Type
    if 'Other' in col:
        oth.append(col)
    elif 'Bioenergy' in col:
        bio.append(col)
    elif 'Solar' in col:
        sol.append(col)
    elif 'Wind' in col:
        wnd.append(col)
    elif 'Hydro' in col:
        hyd.append(col)
    elif 'Nuclear' in col:
        nuc.append(col)
    elif 'Oil' in col:
        oil.append(col)
    elif 'Gas' in col:
        gas.append(col)
    elif 'Coal' in col:
        coal.append(col)
    else:
        print(col, 'not found.')

    # Regions lists
    if 'Earth' in col:
        ear.append(col)
    elif 'Asia Pacific' in col:
        apc.append(col)
    elif 'North America' in col:
        nam.append(col)
    elif 'Europe ' in col:
        erp.append(col)
    elif 'Eurasia' in col:
        eur.append(col)
    elif 'Central & South America' in col:
        csa.append(col)
    elif 'Middle East' in col:
        mde.append(col)
    elif 'Africa' in col:
        afc.append(col)

    # Country lists
    elif 'United States' in col:
        usa.append(col)
    elif 'China' in col:
        chn.append(col)
    elif 'European Union' in col:
        eu.append(col)
    elif 'Japan' in col:
        jpn.append(col)
    elif 'Russia' in col:
        rus.append(col)
    elif 'India' in col:
        ind.append(col)
    elif 'SouthEast Asia' in col:
```

```
        sea.append(col)
    elif 'Brazil' in col:
        bra.append(col)
    else:
        print(col, 'not found.')
```

In [7]:
```
# Creating a DataFrame of America's electricity production fuel sources
model_data = df_hist_elec[usa].copy()
model_data = model_data.iloc[:,[8,7,6,5,4,1,3,2,0]]
model_data
```

Out[7]:

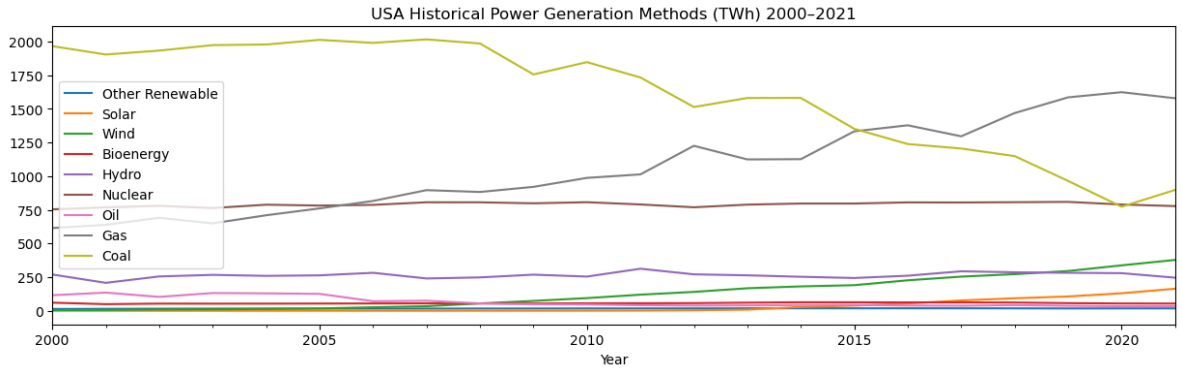
Year	United States Coal (TWh)	United States Gas (TWh)	United States Oil (TWh)	United States Nuclear (TWh)	United States Hydro (TWh)	United States Bioenergy (TWh)	United States Wind (TWh)	United States Solar (TWh)	United States Other Renewables (TWh)
2000-01-01	1,966.27	614.99	116.02	753.89	270.03	60.73	5.59	0.49	14.09
2001-01-01	1,903.96	639.13	135.59	768.83	208.14	49.75	6.74	0.54	14.89
2002-01-01	1,933.13	691.01	103.69	780.06	255.59	53.71	10.35	0.55	16.14
...	...	...	...	...	...	...	...	...	...
2019-01-01	964.96	1,585.81	37.44	809.41	282.61	57.51	295.88	106.89	17.87
2020-01-01	773.39	1,624.17	34.34	789.88	279.95	54.70	337.94	130.72	18.09
2021-01-01	897.89	1,579.36	35.20	778.19	246.47	54.25	378.20	164.42	18.24

22 rows × 9 columns

6.2 Visualizations

I will create a line chart showing the trends of electricity generation in America.

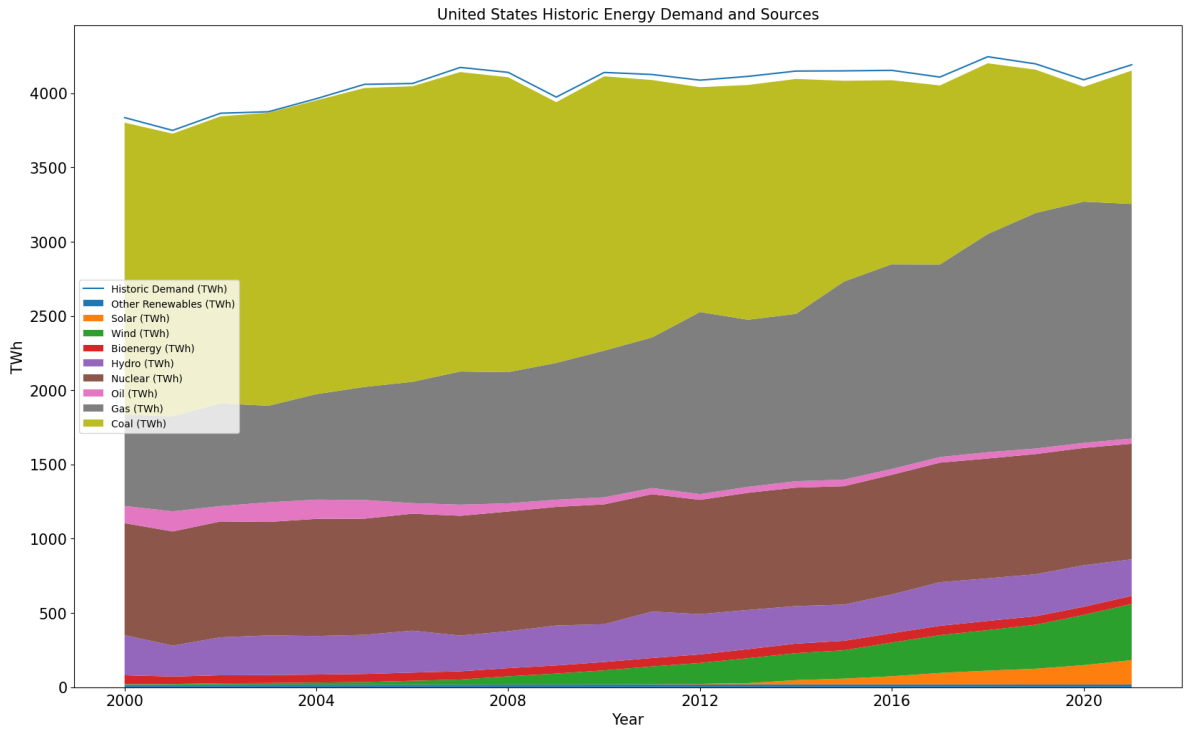
In [8]:
```
line8 = model_data.iloc[:,8].plot(label = 'Other Renewable')
line7 = model_data.iloc[:,7].plot(label = 'Solar')
line6 = model_data.iloc[:,6].plot(label = 'Wind')
line5 = model_data.iloc[:,5].plot(label = 'Bioenergy')
line4 = model_data.iloc[:,4].plot(label = 'Hydro')
line3 = model_data.iloc[:,3].plot(label = 'Nuclear')
line2 = model_data.iloc[:,2].plot(label = 'Oil')
line1 = model_data.iloc[:,1].plot(label = 'Gas')
line0 = model_data.iloc[:,0].plot(figsize=(15,4),label = 'Coal')


plt.legend(loc = 6)
plt.title('USA Historical Power Generation Methods (TWh) 2000-2021');
```



I will create a stacked graph for the same data.

In [9]:
```
def stacked_energy_hist(his_en, his_dem, title, s=0):

    # Creating the labels for the graph
    df_names = list(his_en.columns)
    if s>0:
        for i in range(len(df_names)):
            df_names[i] = df_names[i][s:]
    his_dem.name = his_dem.iloc[:,0:1].columns[0][s:]

    fig = plt.figure(figsize = (20,12))
    plt.title(title, size = 15)

    # Plotting the demand line
    plt.plot(his_dem.index, his_dem.iloc[:,0],
             label = his_dem.name)

    # Plotting the electricity generation numbers.
    his_en_lst = []
    for i in range(len(his_en.columns)):
        his_en_lst.append(his_en.iloc[:,i])

    # Plotting the stacked energy production
    plt.gca().set_prop_cycle(None)
    plt.stackplot(his_en.index, his_en_lst, labels = df_names)

    # Formatting
    plt.xlabel('Year', size = 15)
    plt.ylabel('TWh', size = 15)
    plt.xticks(size=15)
    plt.yticks(size=15)
    plt.legend(loc=6)
    # show the graph
    plt.show()
```

In [10]:
```
stacked_energy_hist(model_data.iloc[:,::-1], df_hist_dem.iloc[:,8:9],
                    'United States Historic Energy Demand and Sources', 14)
```



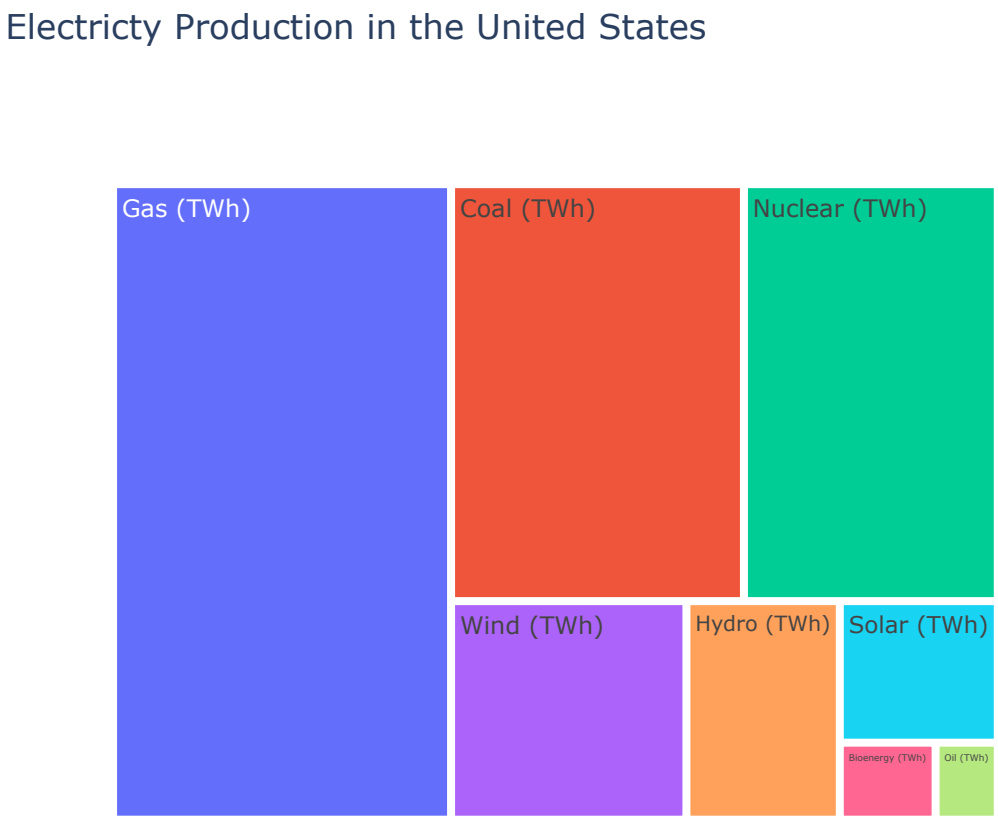
I will create a Treemap of current electricity production percentages in America.

In [11]:
```
pd.set_option('display.max_rows', 9)

# Setting up a DataFrame specifically for the TreeMap.
figure_data = pd.DataFrame(model_data.iloc[21:22,0:8].transpose().copy(),
                           columns = ['Electricity Production'])

figure_data['KWh'] = model_data.iloc[21:22,0:8].transpose().values

total = np.sum(figure_data['KWh'])

for idx in figure_data.index:
    figure_data.loc[idx, 'Electricity Production'] = idx[14:]

    #Calculating percentage for informational purposes.
    figure_data.loc[idx, 'Percentage'] = figure_data.loc[idx, 'KWh']/total

figure_data
```

Out[11]:

	Electricity Production	KWh	Percentage
United States Coal (TWh)	Coal (TWh)	897.89	0.22
United States Gas (TWh)	Gas (TWh)	1,579.36	0.38
United States Oil (TWh)	Oil (TWh)	35.20	0.01
United States Nuclear (TWh)	Nuclear (TWh)	778.19	0.19
United States Hydro (TWh)	Hydro (TWh)	246.47	0.06
United States Bioenergy (TWh)	Bioenergy (TWh)	54.25	0.01
United States Wind (TWh)	Wind (TWh)	378.20	0.09
United States Solar (TWh)	Solar (TWh)	164.42	0.04

In [12]:
```
fig = px.treemap(figure_data, path = ['Electricity Production'],
                 values = 'KWh',
                 title = 'Electricty Production in the United States')
fig.show()
```



7 Modeling Energy Fuel Trends

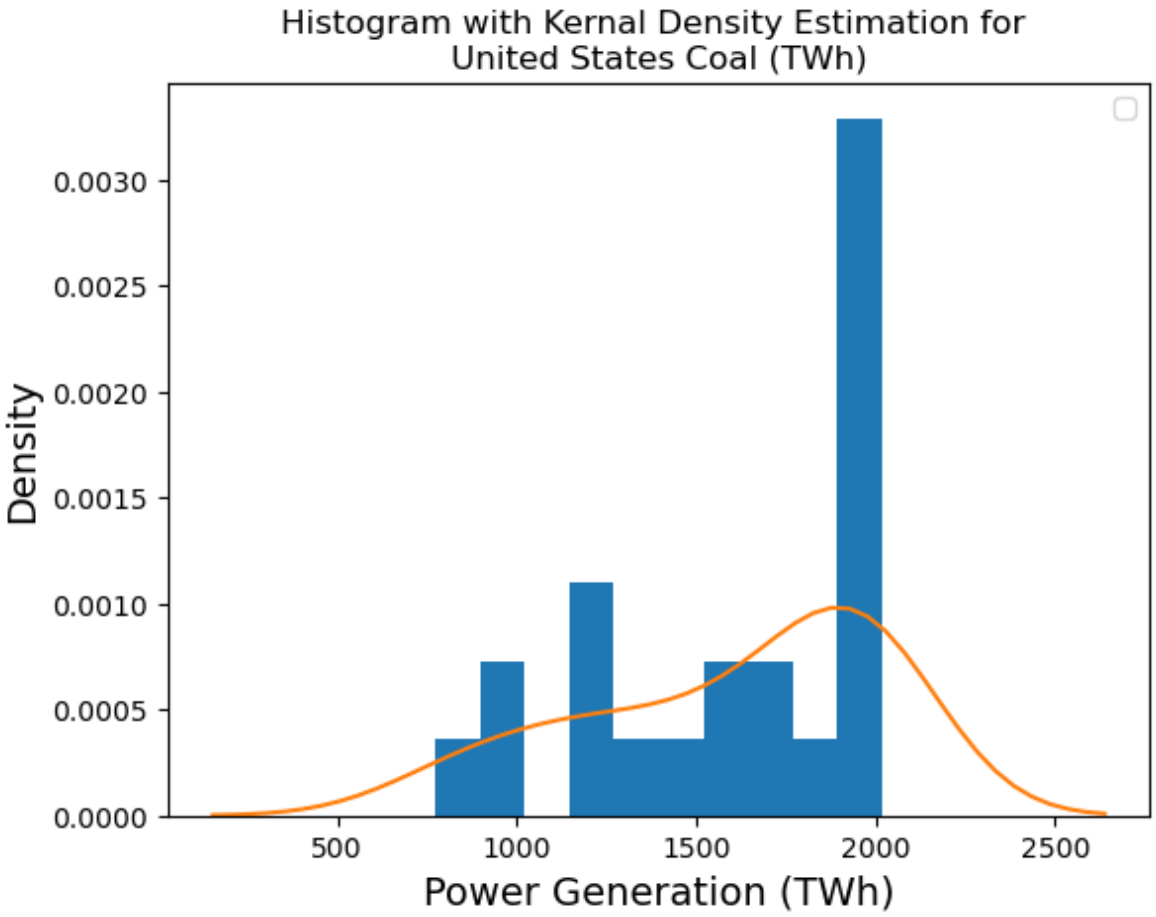
7.1 Coal

7.1.1 Distribution Investigation

In [13]:
```
# Plots a histogram with the kernal density estimation visible.
def hist(df):
    fig, ax = plt.subplots(nrows=1, ncols=1)
    df.plot.hist(ax=ax, density = True, grid=False, label = 'Histogram');
    df.plot.kde(ax=ax, ind=50, label = 'Kernal Density Estimation')
    plt.title('Histogram with Kernal Density Estimation for \n'+df.columns[0])
    plt.ylabel('Density', fontsize=14)
    plt.xlabel('Power Generation (TWh)', fontsize=14)
    plt.legend([])
```

In [14]:
```
# Creates a statistics DataFrame that displays, and collects, model details.
def s_block(df, compare = pd.DataFrame()):
    # Creates a new DataFrame and calculates the associated Statistics.
    block = pd.DataFrame([[df.columns[0], np.mean(df)[0], np.median(df),
                    np.std(df)[0], stats.skew(df)[0], stats.kurtosis(df)[0]]],
                         columns = ['Dataset', 'Mean', 'Median',
                                    'Standard Deviation', 'Skew',
                                    'Fisher Kurtosis'])

    # Concatenates the block with a passed block.
    if ~compare.empty:
        block = pd.concat([compare, block], axis=0)
        block.reset_index(drop=True, inplace=True)

    return block
```

In [15]:
```
hist(model_data.iloc[:,0:1])
```



In [16]:
```
stats_block = s_block(model_data.iloc[:,0:1])
stats_block
```

Out[16]:

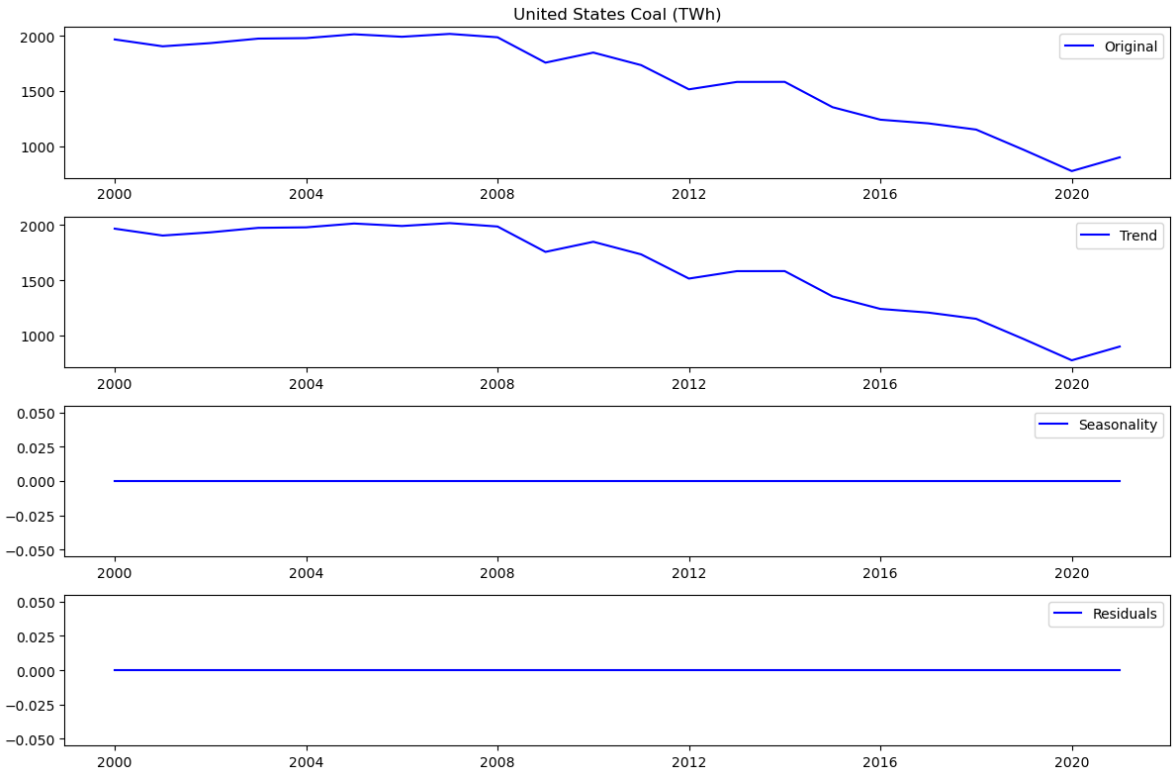
	Dataset	Mean	Median	Standard Deviation	Skew	Fisher Kurtosis
0	United States Coal (TWh)	1,607.17	1,744.66	399.15	-0.68	-0.90

The data is skewed negatively with flat tails on either side.

7.1.2 Decomposing The Data

In [17]:
```
def decomp_graph(df, mod = 'additive'):

    # Activating seasonal_decompose
    decomposition = seasonal_decompose(df, mod)

    # Gather the trend, seasonality, and residuals
    trend = decomposition.trend
    seasonal = decomposition.seasonal
    residual = decomposition.resid

    # Plot gathered statistics
    plt.figure(figsize=(12,8))
    plt.subplot(411)
    plt.title(df.columns[0])
    plt.plot(df, label='Original', color='blue')
    plt.legend(loc='best')
    plt.subplot(412)
    plt.plot(trend, label='Trend', color='blue')
    plt.legend(loc='best')
    plt.subplot(413)
    plt.plot(seasonal,label='Seasonality', color='blue')
    plt.legend(loc='best')
    plt.subplot(414)
    plt.plot(residual, label='Residuals', color='blue')
    plt.legend(loc='best')
    plt.tight_layout()
```

In [18]:
```
decomp_graph(model_data.iloc[:,0:1])
```



No seasonality or residuals. I'll check for stationarity.

7.1.3 Checking for Stationarity and Flattening

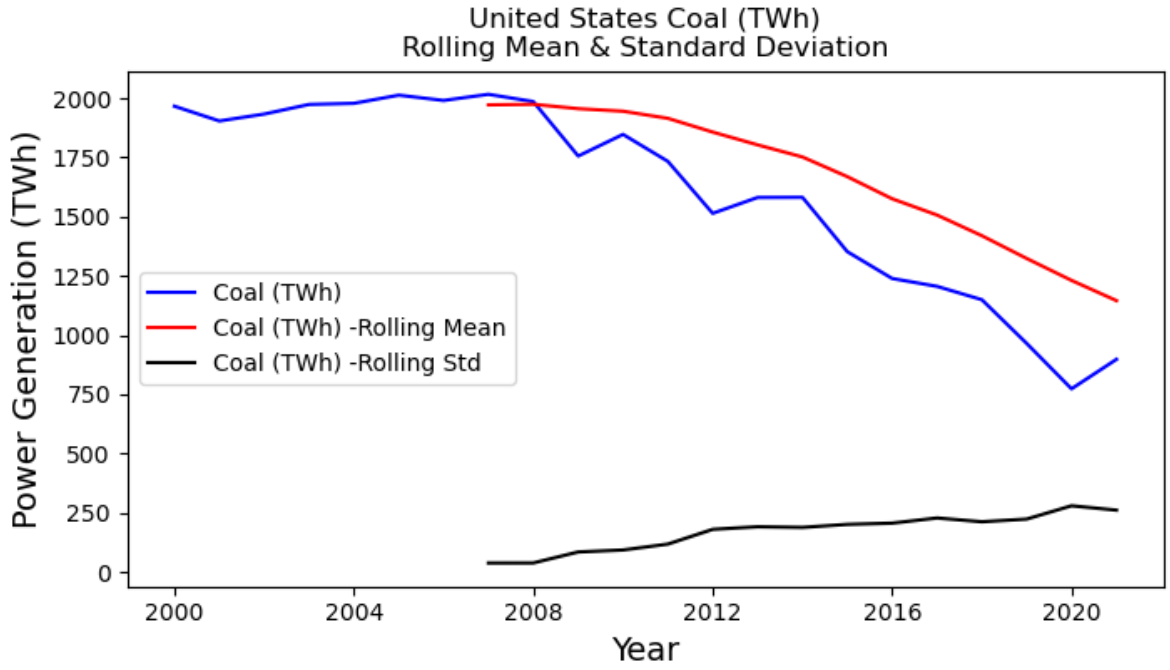
In [19]:
```
def df_test(df):
    # Activate the Augmented Dickey-Fuller. Dropping any NaNs if necessary.
    dftest = adfuller(df.dropna())

    # Extract and display test results in a reader friendly manner
    dfoutput = pd.Series(dftest[0:4], index=['Test Statistic', 'p-value',
                                             '#Lags Used',
                                             'Number of Observations Used'])

    # Add critical values to the output.
    for key,value in dftest[4].items():
        dfoutput['Critical Value (%s)'%key] = value

    # Print the results.
    print (df.columns[0] + '\nResults of Dickey-Fuller test: \n')
    print(dfoutput)
```

In [20]:
```
# Plots the data, standard defiation and rolling mean of the passed data.
def mean_std(df, title='Rolling Mean & Standard Deviation', s=0):
    # The parameter "s" is used repeatedly in this noebook. It is a value
    # meant to cut off the repeated beginning of the legend. It's how I
    # Get rid of "United States" at the beginning of each label in the legend.

    # Calculate values lines
    roll_mean = df.rolling(window=8, center=False).mean()
    roll_std = df.rolling(window=8, center=False).std()

    # Plot the figure.
    fig = plt.figure(figsize=(8,4))
    plt.plot(df, color='blue', label=str(df.columns[0])[s:])
    plt.plot(roll_mean, color='red', label=str(
                                        df.columns[0])[s:]+' -Rolling Mean')
    plt.plot(roll_std, color='black', label=str(
                                        df.columns[0])[s:]+' -Rolling Std')
    plt.legend(loc='best')
    plt.title(df.columns[0] + '\n' + title)
    plt.ylabel('Power Generation (TWh)', fontsize=14)
    plt.xlabel('Year', fontsize=14)
    plt.show(block=False)
```

In [21]:
```
# I put the Dickey Fuller and the associated plot together into one function.
def stationarity_check(df, t = 'Rolling Mean & Standard Deviation', s=0):
    df_test(df)
    mean_std(df, title=t, s=s)
```

In [22]:
```
pd.set_option('display.max_rows', 30)
stationarity_check(model_data.iloc[:,0:1], s=14)
```

```
United States Coal (TWh)
Results of Dickey-Fuller test:

Test Statistic                  0.75
p-value                         0.99
#Lags Used                      3.00
Number of Observations Used    18.00
Critical Value (1%)            -3.86
Critical Value (5%)            -3.04
Critical Value (10%)           -2.66
dtype: float64
```



This is not close to stationary. After playing around with transformations, I found that if I took the difference and then subtracted the rolling average of seven values, I can get it stationary.

In [23]:
```
# This function subtracts the rolling mean, adds identifying text to the colum
# name and returns it. n is the number of values averaged together.
def subtract_roll_mean(df, n=0):
    roll_mean = df.rolling(window=n, center=False).mean()
    sub_roll_mean = df - roll_mean
    sub_roll_mean = sub_roll_mean
    sub_roll_mean.columns = [
        'Subtract Rolling Mean of ' + str(sub_roll_mean.columns[0])]

    return sub_roll_mean
```

In [24]:
```
# This function takes the difference of the data and returns a the difference
# column.
def difference(df):
    diff_df = df.diff()
    diff_df.columns = ['Annual Change of ' + str(df.columns[0])]
    return diff_df
```

In [25]:
```
# Create a DataFrame that includes both the original data and the
# transformation. I'll use this DataFrame throughout the entire Notebook.
model_data_t = model_data.copy()

t = subtract_roll_mean(difference(model_data.iloc[:,0:1]), n=7)
model_data_t.insert(1,t.columns[0],t)

model_data_t.iloc[:,1:2]
```

Out[25]:

Year	Subtract Rolling Mean of Annual Change of United States Coal (TWh)
2000-01-01	NaN
2001-01-01	NaN
2002-01-01	NaN
2003-01-01	NaN
2004-01-01	NaN
2005-01-01	NaN
2006-01-01	NaN
2007-01-01	18.78
2008-01-01	-42.35
2009-01-01	-204.58
2010-01-01	109.45
2011-01-01	-78.88
2012-01-01	-148.13
2013-01-01	125.56
2014-01-01	62.71
2015-01-01	-138.82
2016-01-01	-39.43
2017-01-01	58.33
2018-01-01	27.07
2019-01-01	-106.09
2020-01-01	-76.18
2021-01-01	222.19

Before I move forward, I want to make sure I can transform this data back after modeling.

In [26]:
```
# This column reverses the subtract rolling mean operation and returns the
# DataFrame. The input must include both the original data and the transformed
# data. The user can stack predicted values on the right column on top of the
# transformed orginal data andleave the orginal data column as NaNs next to
# the predicted values.
def inv_subtract_roll_mean(dfo, n=0, ex_copy = 0):
    df = dfo.copy()

    # Personal note: Why did I have to figure out how to do this myself?
    # Why isn't there a standard proess for reversing the subtract rolling
    # mean operation since it's so common in Time Series analysis.
    roll_sum = df.iloc[:,0:1].copy()
    for i in range(n-1+ex_copy, len(df.iloc[:,0:1])):
        roll_sum.iloc[i,0] = (n*df.iloc[i,1:3]+
                              np.sum(np.sum(roll_sum.iloc[i-n+1:i,0])))/(n-1)

    roll_sum.columns = ['Back Transformation of '+df.columns[1]]

    return roll_sum
```

In [27]:
```
# Inverses taking the difference of the data. Insert a double column dataframe
# with the original values on the left and the differenced values on the right
def inv_difference(df):
    data = df.copy()
    data['Back Transformation of ' + str(df.columns[1])] = np.nan
    data.iloc[0,2:3] = data.iloc[0,0:1]
    for i in range(1,len(df)):
        data.iloc[i,2:3] = data.iloc[i-1,2:3].values[0]+ data.iloc[
                                                          i,1:2].values[0]
    return data.iloc[:,2:3]
```

In [28]:
```
# This combines the reversal of subtract rolling mean and difference functions
# since passing the data through both requires some extra steps and dataframe
# management.
def inv_diff_inv_sub(df, n=0):
    # DataFrame, df, has the correct values on the left, and the transformed
    # values on the right. The projected values are stacked on top of the
    # transformed values, also on the right.

    test = df.copy()

    # Recreate the original differenced column to pass to the rolling mean
    # function.
    t = difference(test.iloc[:,0:1])
    test[str(t.columns[0])] = t
    test = test.iloc[:,[0,2,1]]

    # Pass this to the inverse subtract rolling mean function
    t = inv_subtract_roll_mean(test.iloc[:,1:], n=n, ex_copy=1)
    test[str(t.columns[0])] = t

    # Pass it back through the inverse difference function and return the
    # column.
    t = inv_difference(test.iloc[:,[0,3]])
    test[str(t.columns[0])] = t
    test.rename(columns={test.iloc[:,4:5].columns[0]:
        'Back Transformation of ' + df.iloc[:,1:2].columns[0]},inplace=1)

    return test.iloc[:,4:5]
```

In [29]:
```
# Testing to see if it worked.
test = model_data_t.iloc[:,0:2].copy()
test.iloc[:,1:2] = inv_diff_inv_sub(test, n=7)
test
```

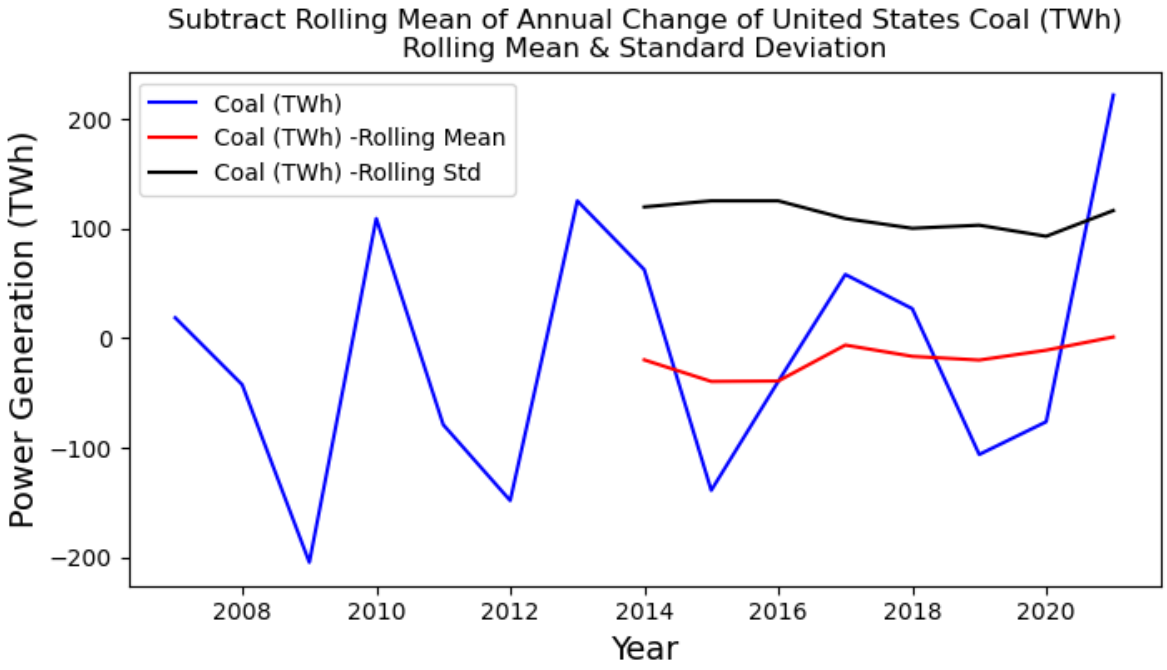
Out[29]:

Year	United States Coal (TWh)	Subtract Rolling Mean of Annual Change of United States Coal (TWh)
2000-01-01	1,966.27	1,966.27
2001-01-01	1,903.96	1,903.96
2002-01-01	1,933.13	1,933.13
2003-01-01	1,973.74	1,973.74
2004-01-01	1,978.30	1,978.30
2005-01-01	2,012.87	2,012.87
2006-01-01	1,990.51	1,990.51
2007-01-01	2,016.46	2,016.46
2008-01-01	1,985.80	1,985.80
2009-01-01	1,755.90	1,755.90
2010-01-01	1,847.29	1,847.29
2011-01-01	1,733.43	1,733.43
2012-01-01	1,514.04	1,514.04
2013-01-01	1,581.11	1,581.11
2014-01-01	1,581.71	1,581.71
2015-01-01	1,352.40	1,352.40
2016-01-01	1,239.15	1,239.15
2017-01-01	1,205.84	1,205.84
2018-01-01	1,149.49	1,149.49
2019-01-01	964.96	964.96
2020-01-01	773.39	773.39
2021-01-01	897.89	897.89

Transferring back is a possibility. Therefore, we can move forward. I'll again check for stationarity.

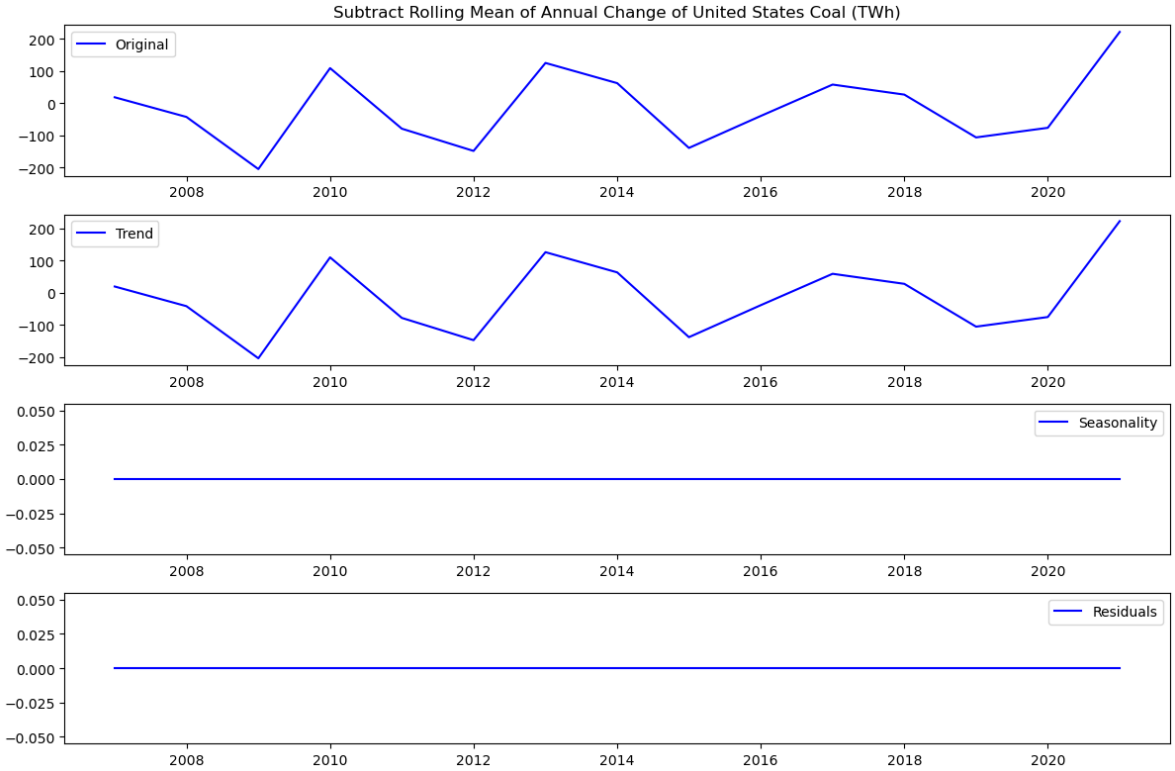
In [30]:
```
stationarity_check(model_data_t.iloc[:,1:2], s=56)
```

```
Subtract Rolling Mean of Annual Change of United States Coal (TWh)
Results of Dickey-Fuller test:

Test Statistic                -2.87
p-value                        0.05
#Lags Used                     2.00
Number of Observations Used   12.00
Critical Value (1%)           -4.14
Critical Value (5%)           -3.15
Critical Value (10%)          -2.71
dtype: float64
```



My p-value is .05. Stationarity achieved. I'll look at the decomposition graph again.

In [31]:
```
decomp_graph(model_data_t.iloc[:,1:2].dropna())
```



Looks like there's still a trend line, but it does pass stationarity, so I'm moving forward to the baseline random walk model.

7.1.4 Random Walk Model

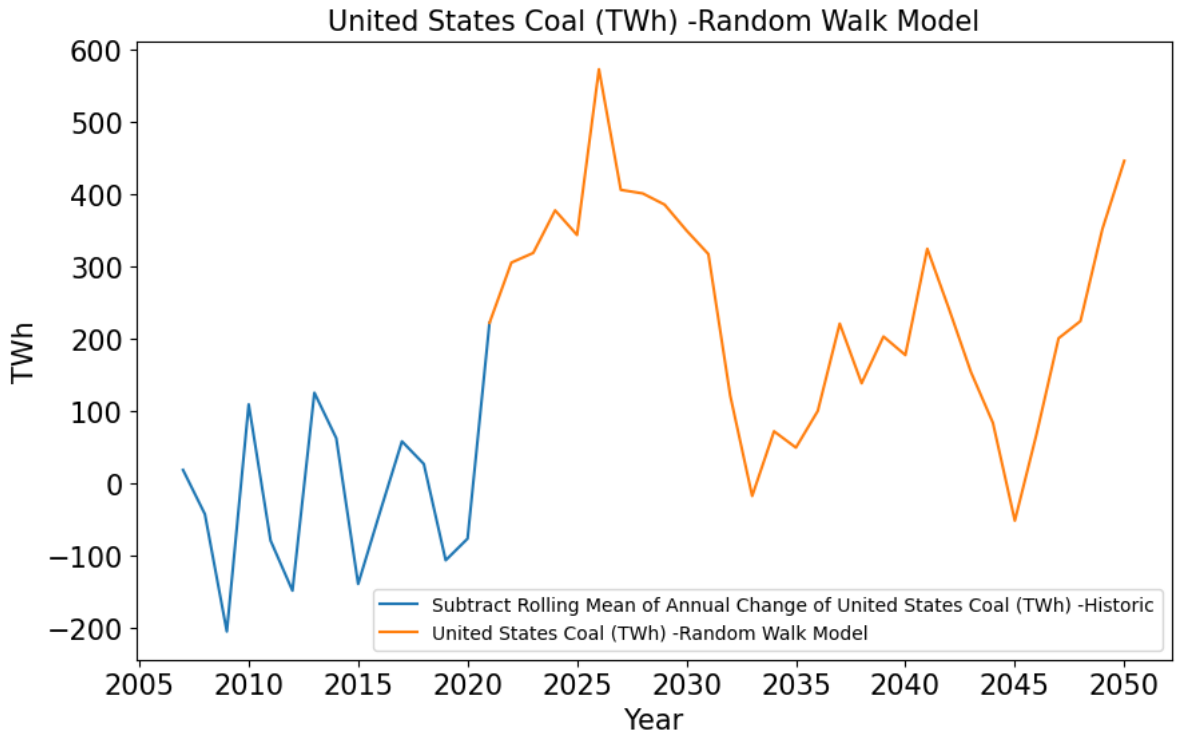
In [32]:
```
def random_walk(df):

    # Create a series with the specified dates
    # Need to drop any NaN terms.
    first = df.dropna().copy()
    skipped = len(df) - len(first)
    dates_first = str(first.iloc[0:1,0:1].index[0])[:10]
    dates_50 = pd.date_range("2022-01-01", "2051-01-01", freq="AS",
                         inclusive="left")

    # White noise error term
    # For standard devitaion, I took the standard deviation of the transformed
    # data.
    diff = df.diff().copy()
    stnd_dev = np.std(df)[0]
    error = np.random.normal(0, stnd_dev, len(dates_50))

    # Printing the calculated standard deviation.
    print('Random Walk with Standard Deviation',
          'of Transformed Data set: {:.2f}'.format(stnd_dev))

    #Starting point is the last transformed value.
    Y_0 = df.iloc[-1,0:1].values[0]

    # Creating the plot data
    cum_error = np.cumsum(error)
    values = pd.DataFrame(cum_error + Y_0, index=dates_50)

    series = pd.DataFrame(values, index=dates_50)
    series.columns = [df.columns[0] + ' -Random Walk Model']

    return series
```

In [33]:
```
# I used this same code for multiple graphs, so I decided to make a function
# that passed all these instructions to any graph I need this data in.

def graph_formatting(df, y_hat, s):
    # Showing IPCC targets lines if what is printed is Fossil Fuels
    fossil = [' Gas ', ' Oil ', ' Coal ']
    if (any([fuel in str(df.columns[0]) for fuel in fossil])) & (
        any([str(df.columns[0]) in col for col in df_hist_elec.columns])):

        goal = pd.concat([df,y_hat],axis=1).iloc[:,0:0]
        goal['IPCC Emissions 2030 Reduction Goal'] = df.iloc[10,0]*.45
        goal['IPCC Emissions 2050 Reduction Goal'] = 0
        plt.plot(goal.iloc[10:31].index, goal.iloc[10:31,0:1].values, '--',
                                                    label = goal.columns[0])
        plt.plot(goal.iloc[30:51].index, goal.iloc[30:51,1:2].values, '--',
                                                    label = goal.columns[1])

    # Plotting both historic and projected energy demand if the values are
    # high enough to need the comparison.

    if np.max([float(df.max()[0]), float(y_hat.max()[0])]) > 4000:

        if df.columns[0][:6] == 'Europe':
            c=7
        else:
            c=6
        for i in range(len(df_proj_dem_stat.columns)):
            if df_proj_dem_stat.columns[i][:c] in df.columns[0]:
                dc = i
                plt.plot(df_hist_dem.iloc[:,dc].index, df_hist_dem.iloc[:,dc],
                   '--', label=str(df_hist_dem.iloc[:,dc:dc+1].columns[0])[s:]
                plt.plot(df_proj_dem_stat.iloc[:,dc].index,
                    df_proj_dem_stat.iloc[:,dc],
                    '--', label=str(
                    df_proj_dem_stat.iloc[:,dc:dc+1].columns[0])[s:])
                break

    # Final Graph Formatting
    plt.xlabel('Year', size = 15)
    plt.ylabel('TWh', size = 15)
    plt.xticks(size=15)
    plt.yticks(size=15)

#    return goal, df_hist_dem, df_proj_dem_stat,
```

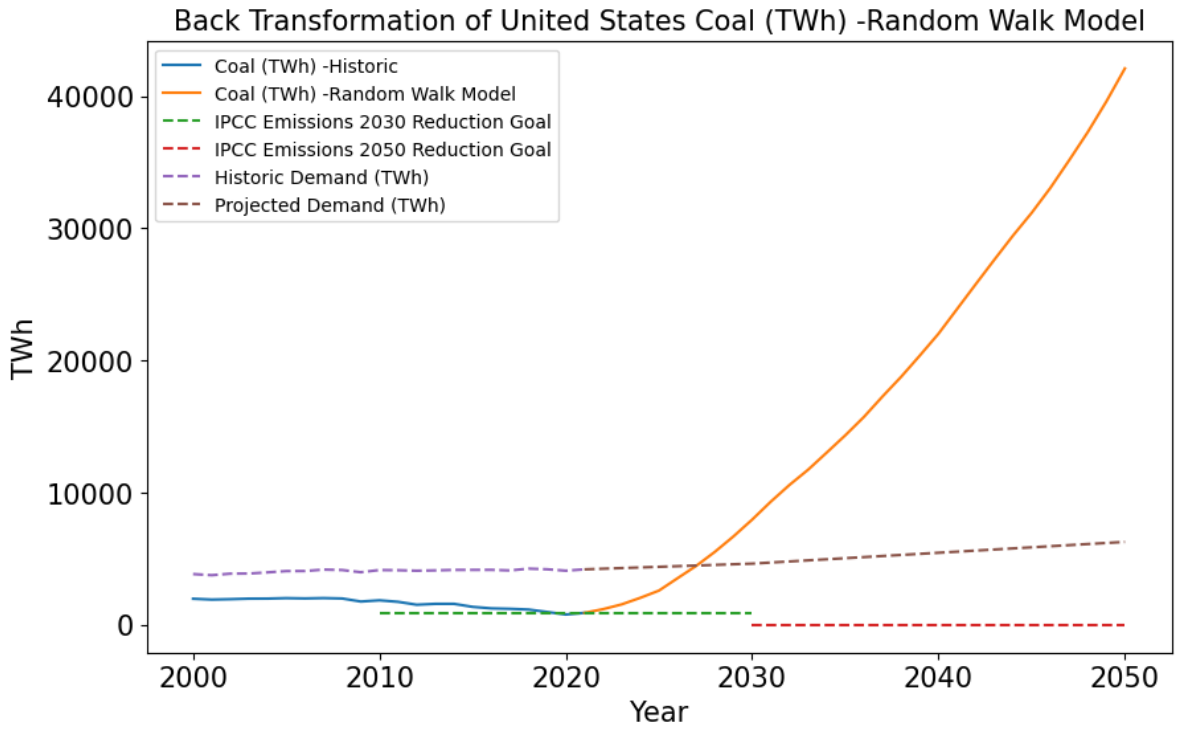
In [34]:
```
# Plots the historic and projected data.
def pred_graph(df, y_hat, s=0, title='col'):

    # Adding the end point of the original data to the y_hat DataFrame
    # so the plot looks continuous.
    y_hat.loc[df.index[-1], y_hat.columns] = df.iloc[-1,0]
    y_hat.sort_index(inplace=True)

    fig = plt.figure(figsize = (10,6))

    # Plotting the historic and projected growth of the energy production.

    plt.plot(df.index, df.iloc[:,0], label=str(df.columns[0])[s:]+' -Historic'

    # In case the beginning of the column is "Back Transformation of " but the
    # graph doesn't need that in the leged label.
    if s > 0:
        if str(y_hat.columns[0])[:23] == 'Back Transformation of ':
            ss = s+23
        else:
            ss = s+0
    else:
        ss = s+0

    # Plotting the predicted data
    plt.plot(y_hat.index, y_hat.iloc[:,0], label=str(y_hat.columns[0])[ss:])

    # Formatting the graph
    graph_formatting(df, y_hat, s)

    # Creating the legend.
    plt.legend(loc='best')
    if title == 'col':
        plt.title(y_hat.columns[0], size = 15)
    else:
        plt.title(title, size = 15)
    plt.show()
```

In [35]:
```
pd.set_option('display.max_rows',None)
y_hat_proj = random_walk(model_data_t.iloc[:,1:2])
y_hat_proj.columns = [model_data.columns[0]+' -Random Walk Model']

# Plot the transformed Random Walk
pred_graph(model_data_t.iloc[:,1:2], y_hat_proj.iloc[:,0:1])
```

```
Random Walk with Standard Deviation of Transformed Data set: 113.22
```



In [36]:
```
# Prepare the DataFrame for Back Transformation
ranplot = model_data_t.iloc[:,0:2].copy()
ranplot.rename(columns={str(ranplot.columns[1]): str(y_hat_proj.columns[0])},
               inplace=True)
ranplot = pd.concat([ranplot,y_hat_proj],axis=0)

# Perform Back Transformation
y_hat_proj = inv_diff_inv_sub(ranplot, n=7)

#Plot the new graph
pred_graph(model_data.iloc[:,0:1], y_hat_proj.iloc[22:,0:1], 14)
```



This model is as unreliable as it's name would imply, random walk. One of the things it relies on is a standard deviation of values. I decided to use the standard deviation of the data itself, but since it oscilates at a pretty clear pattern, up then down, at a pretty wide margin, the data of the random walk follows the same rises or falls, but because it is random, it spends too many cycles going up in a row or down in a row. This leads to most of the iterations of this model to outrageously high or low scores once the transformation is put back into place. Perhaps this is just a bad model for this? Perhaps I need to choose a different standard deviation? I spent a lot of time questioning this without getting any good answers.

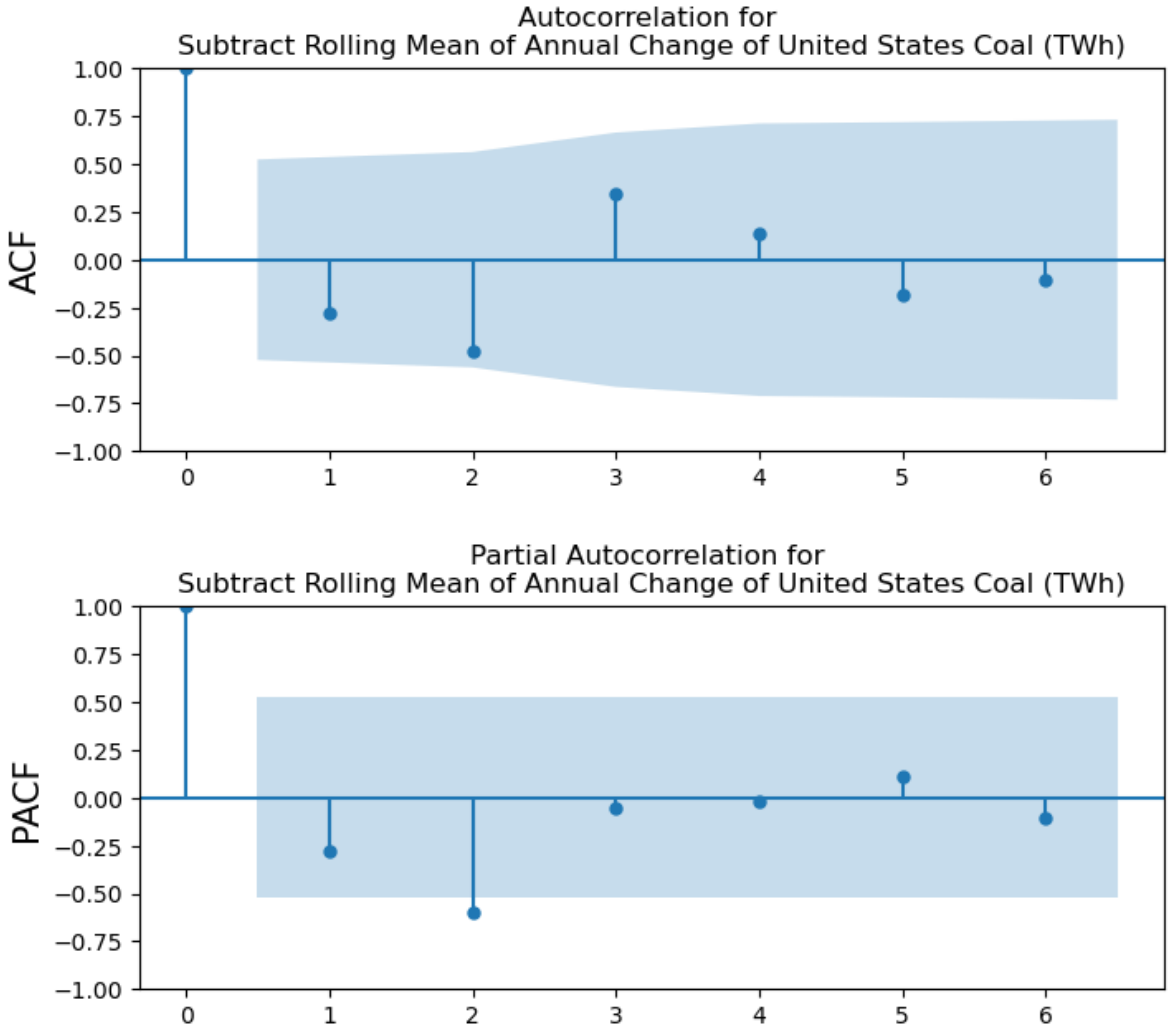
Now I'll move forward with an ARMA model. First, I need to evaluate the ACF and PACF plots to find the best Autoregressive and Moving Average terms.

7.1.5 Evaluating Initial AR and MA Values with ACF and PACF Plots

In [37]:
```
# Plot the ACF
def plotacf(df, lags = 8):
    df_diff = df.diff().dropna()

    # Designed the function to work plot multiple ACFs if enough columns were
    # passed.
    if len(df.columns) > 1:
        for col in df_diff.columns:
            fig, ax = plt.subplots(figsize=(8, 3))
            # The plot ACF function.
            plot_acf(df_diff[col], ax=ax, lags=lags);
            plt.title('Autocorrelation for \n' + col)
            plt.ylabel('ACF', size = 15)
    else:
        fig, ax = plt.subplots(figsize=(8, 3))
        # The plot ACF function.
        plot_acf(df_diff, ax=ax, lags=lags);
        plt.title('Autocorrelation for \n' + df.columns[0])
        plt.ylabel('ACF', size = 15)
```

In [38]:
```
# Plot the PACF
def plotpacf(df, lags = 8):
    df_diff = df.diff().dropna()

    # Designed the function to work plot multiple ACFs if enough columns were
    # passed.
    if len(df.columns) > 1:
        for col in df_diff.columns:
            fig, ax = plt.subplots(figsize=(8, 3))
            # The plot PACF function.
            plot_pacf(df_diff[col], ax=ax, lags=lags, method="ywm");
            plt.title('Partial Autocorrelation for \n' + col)
            plt.ylabel('PACF', size = 15)

    else:
        fig, ax = plt.subplots(figsize=(8, 3))
        # The plot PACF function.
        plot_pacf(df_diff, ax=ax, lags=lags, method="ywm");
        plt.title('Partial Autocorrelation for \n' + df.columns[0])
        plt.ylabel('PACF', size = 15)
```

In [39]:
```
plotacf(model_data_t.iloc[:,1:2], lags = 6)
plotpacf(model_data_t.iloc[:,1:2], lags = 6)
```



Since we have a breach of the confidence interval on the PACF at 2, that means that, for the baseline model, the Autoregressive term, AR, should be 2, while the other terms are 0. Now to create the first ARMA model with (2,0,0).

7.1.6 ARMA Model

In [40]:
```
# This prints the standard statistics of the model.
def summary_print(mod_lst):
    asterisks = ('****************************************' +
                 '****************************************')

    # Works with multiple models if a list is passed.
    if str(type(mod_lst)) != list:
        print(asterisks)
        print(mod_lst.name, 'model results.')
        print()
        print(mod_lst.summary())
        return
    for mod in mod_lst:
        print(asterisks)
        print(mod.name, 'model results.')
        print()
        print(mod.summary())
        print()
        print()
```

In [41]:
```
# Preparing the DataFrame for backwards transformation was the same for all
# backwards transformations, so I put this process into its own function.
# Also passes back the data in the correct format
def back_transformation(df, y_hat, st, en, transformation, n=0):
    # Preparing data for passing to the inverse functions
    df_copy = df.copy()
    df_copy.columns = [df_copy.columns[0], y_hat.columns[0]]
    # Check if this run is modeling future data
    if y_hat.index[-1] > df_copy.index[-1]:
        pred_p = df_copy.iloc[:,1:2].append(y_hat.iloc[1:,:])
        pred_p = pd.concat([df_copy.iloc[:,0:1],pred_p], axis=1)
    # Modeling the given data to check RMSE
    else:
        pred_p = pd.concat([df_copy.iloc[:,0:1],y_hat.iloc[:,0:1]],axis=1)

    # Performing the transformation
    if transformation == 'inv_diff_inv_sub':
        pred = inv_diff_inv_sub(pred_p, n)
    if transformation == 'inv_log_inv_sub':
        pred = inv_log_inv_sub(pred_p, n)
    if transformation == 'inv_subtract_roll_mean':
        pred = inv_subtract_roll_mean(pred_p, n)
    if transformation == 'inv_difference':
        pred = inv_difference(pred_p)
    if transformation == 'inv_log_transform':
        pred = inv_difference(pred_p)

    # To check if the given data was back transformed correctly.
    df_bt = pred.iloc[:len(df_copy),0:1]
    df_bt.rename(columns={df_bt.iloc[:,0:1].columns[0]:
        'Back Transformation of ' + df.iloc[:,0:1].columns[0]},inplace=1)

    # Creating the back transformed prediction
    y_hat_bt = pred.iloc[st:,0:1]

    return df_bt, y_hat_bt
```

In [42]:
```
# Creates the model, calcuates y_hat, makes backwards transformation and
# calculates the model statistics.
def mod_pred(df, o, st, en, message='n', tran=None, n=0):

    # For dataFrames with fewer layers.
    l = len(df.columns)

    # Fitting the Model
    mod_arima = ARIMA(df.iloc[:,l-1:l], order = o)
    fit_arima = mod_arima.fit()
    fit_arima.name = df.columns[0]

    # Predicting the specified timeline, st and en.
    y_hat_t = pd.DataFrame(fit_arima.predict(start = st-1, end = en))
    y_hat_t.columns = [df.columns[0] + ' -ARIMA Prediction ' + str(o) +
                       ' On Transformation']

    # Predicting the original DataFrame for RMSE check
    y_hat_org = pd.DataFrame(fit_arima.predict(start = 0, end = len(df)-1))

    df_bt = None
    y_hat_bt = None
    y_hat_org_bt = None

    # Performing back transformations.
    if tran != None:
        df_bt, y_hat_bt = back_transformation(df, y_hat_t, st, en, tran, n)

        df_bt_jnk, y_hat_org_bt = back_transformation(df, y_hat_org, 0,
                                                      len(df)-1, tran, n)

        # Calculating the RMSE and AIC for the model.
        rmse, aic = mod_stats(fit_arima, df.iloc[:,0:1], y_hat_org_bt,
                              message = message, o = o, n=n)

        # Renaming the column for predicted values.
        y_hat_org_bt.columns = [str(df.columns[0])+
                            ' -ARIMA Prediction '+str(o)+' on Original']
        y_hat_org.columns =    [str(df.columns[0])+
                            ' -ARIMA Prediction '+str(o)+' on Transformation']

    # In case transformations are not needed,
    else:
        # Calculate RMSE and AIC
        rmse, aic = mod_stats(fit_arima, df.iloc[:,0:1], y_hat_org,
                              message = message, o = o)
        # Rename the prediction column
        y_hat_org.columns =    [str(df.columns[0])+
                            ' -ARIMA Prediction '+str(o)+' on Original']
        y_hat_org_bt = y_hat_org.copy()

    return fit_arima, y_hat_t, rmse, aic, df_bt, y_hat_bt, y_hat_org_bt
```

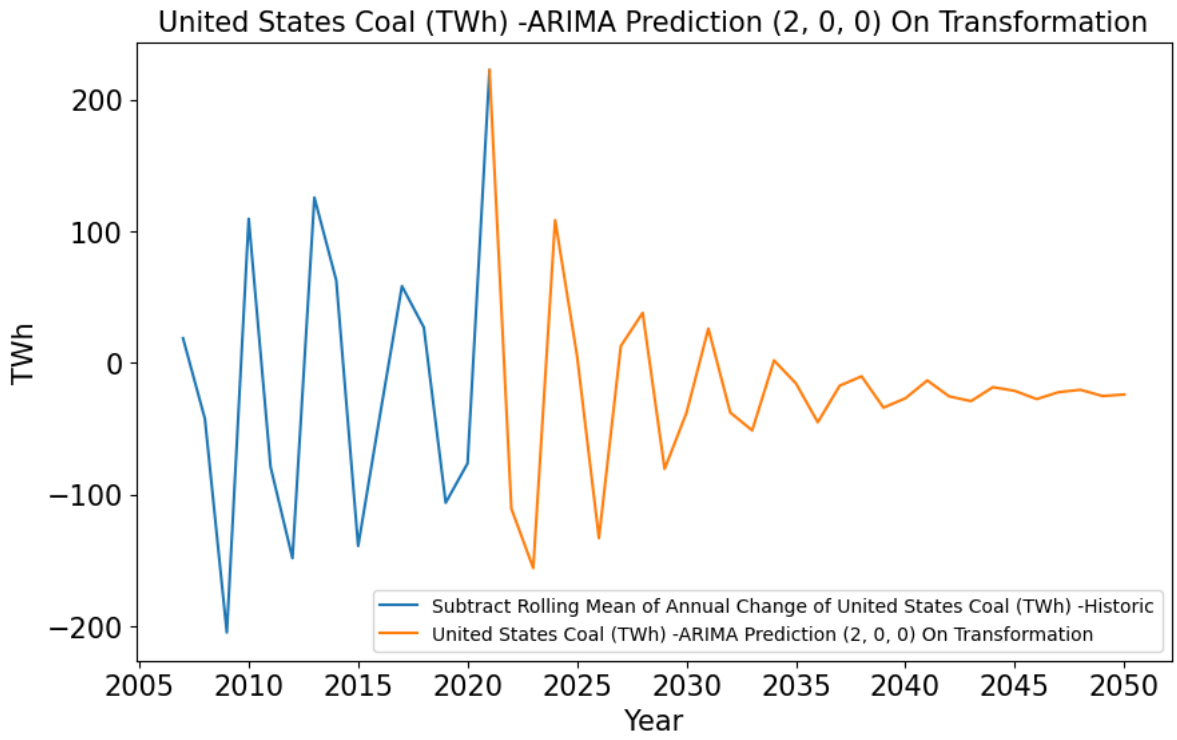
In [43]:
```
# The function mod_pred has a lot of returned parameters. I don't always need
# all of those, so I made this function that runs the mod_pred function, but
# passes back fewer parameters.
def mod_pred_s(df, o, st, en, message='n', tran=None, n=0):
    fit_arima, y_hat_t, rmse, aic, df_bt, y_hat_bt, y_hat_org_bt = mod_pred(df
                                  o, st, en, message=message, tran=tran, n=n)

    return y_hat_t, y_hat_org_bt, y_hat_bt
```

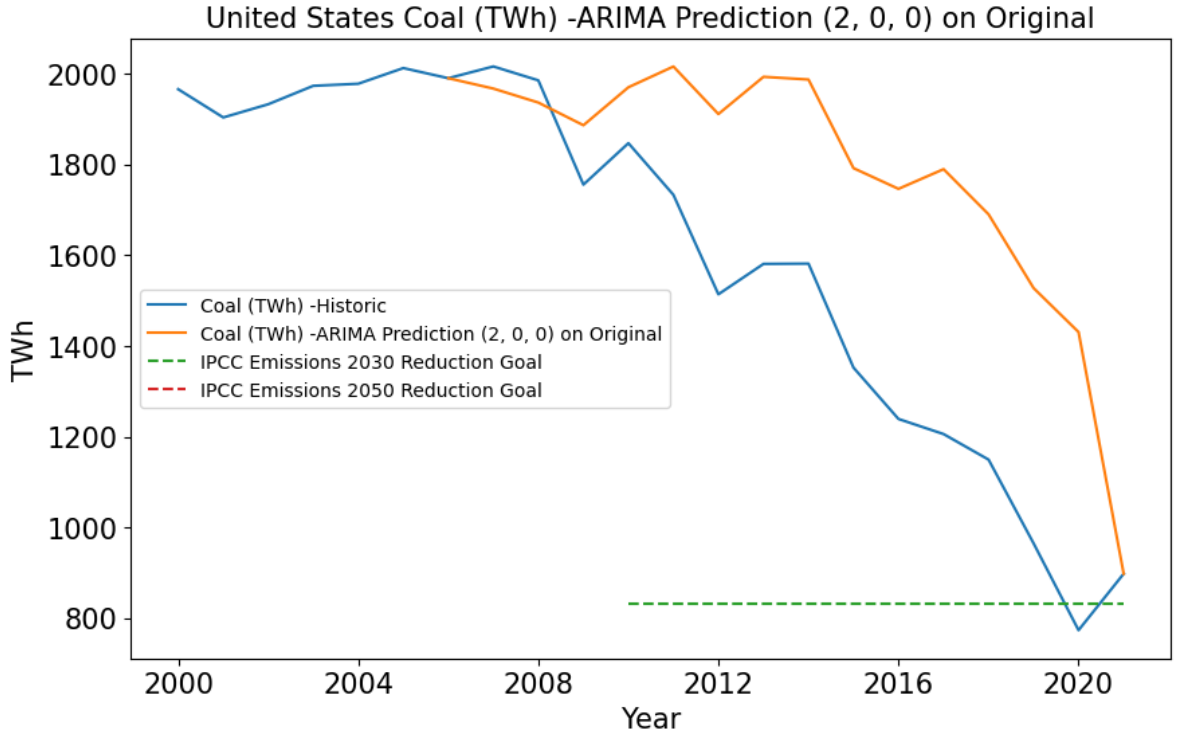
In [44]:
```
# This calculates RMSE and AIC.
def mod_stats(mod, df, y_hat, message='n', o=None, n=0):
    rmse = sk.metrics.mean_squared_error(df.iloc[n:], y_hat.iloc[n:],
                                         squared = False)
    mod_aic = mod.aic
    if message == 'y':
        print()
        print('ARIMA: {}'.format(o) + ', RMSE={:.2f}, AIC={:.2f}'.format(rmse,
                                                                         mod_aic))
    return rmse, mod_aic
```

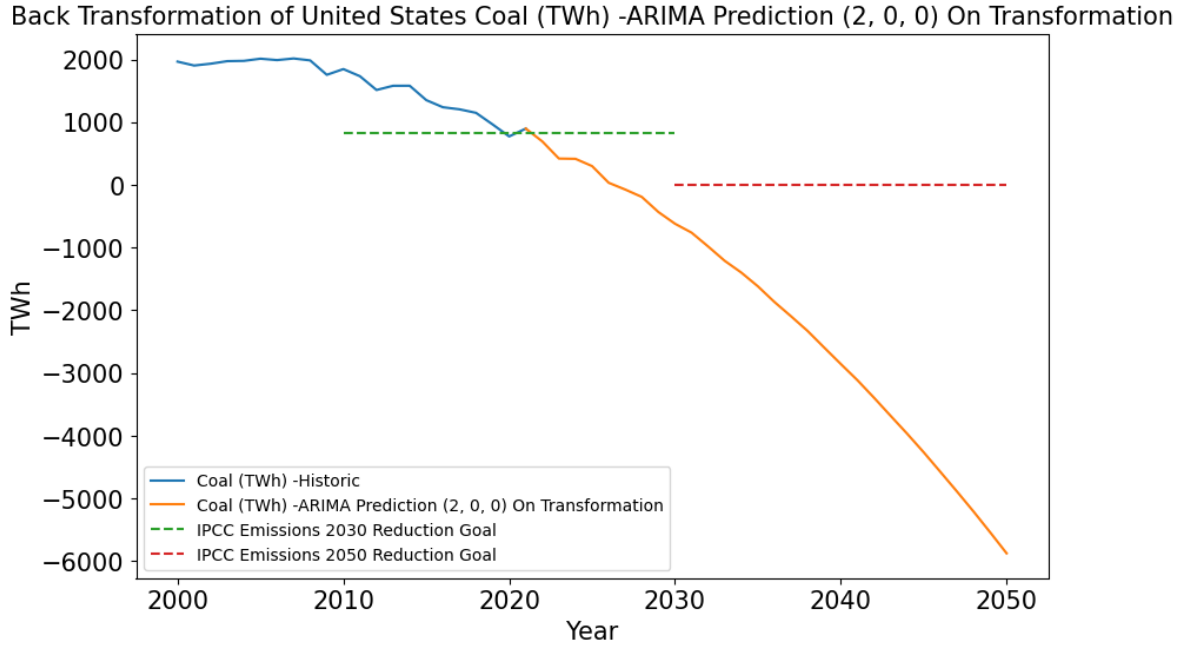
In [45]:
```
# This will perform all of the above functionsin one line of code.
def model_summary(df, o, st, en, s=0, tran=None, n=0):
    l = len(df.columns)

    # Making predictions and calculating RMSE / AIC
    model, y_hat_t, rmse, mod_aic, df_bt, y_hat_bt, y_hat_org_bt = mod_pred(df
                                  o, st, en, message = 'y', tran=tran, n=n)

    # Printing results
    summary_print(model)

    #Plotting result.
    pred_graph(df.iloc[:,l-1:l], y_hat_t, 0)

    # There's no standard starting point for the original data projection sinc
    # some models start their projection after a specific number of copies of
    # the original data. To make this look nicer in the graph, I wrote this
    # for loop. It will determine where to start if the model copies the first
    # few values.
    b_num = n
    for i in range(1,len(y_hat_org_bt)):
        # Rounding the values to add a small degree of uncertainty.
        y_hat_org_bt_point = round(y_hat_org_bt.iloc[i,0:1].values[0],1)
        df_point = round(df.iloc[i,0:1].values[0],1)
        if y_hat_org_bt_point != df_point:
            break
        else:
            b_num -= 1

    # Printing a comparison of the original data and the prediction of the sam
    # time period back transformed. This graph visualizes the data used to
    # calculate the RMSE score.
    pred_graph(df.iloc[:,0:1], y_hat_org_bt.iloc[n-b_num:,:], s)

    # Printing the original data with the projected back transformed data.
    # If there is no transformation, y_hat_bt is still the value used to
    # produce this graph.
    pred_graph(df.iloc[:,0:1], y_hat_bt, s)
```

In [46]:
```
pd.set_option('display.max_rows',60)
order = (2,0,0)
model_summary(model_data_t.iloc[:,0:2], order, 22, 50, 14,
          tran = 'inv_diff_inv_sub', n=7)
```

```
ARIMA: (2, 0, 0), RMSE=424.96, AIC=182.38
**********************************************************************
**
United States Coal (TWh) model results.

                                                           SARIMAX Results
==========================================================================
==============================================
Dep. Variable:     Subtract Rolling Mean of Annual Change of United States Co
al (TWh)   No. Observations:                   22
Model:                                                             ARIMA
(2, 0, 0)   Log Likelihood                -87.189
Date:                                                          Sat, 29
Apr 2023   AIC                            182.378
Time:
06:14:58   BIC                            186.742
Sample:                                                              01
-01-2000   HQIC                           183.406
                                                                   - 01
-01-2021
Covariance Type:
opg
==========================================================================
=
                 coef    std err          z      P>|z|      [0.025      0.97
5]
--------------------------------------------------------------------------
-
const        -23.0493     10.435     -2.209      0.027     -43.501      -2.59
8
ar.L1         -0.5138      0.333     -1.544      0.122      -1.166       0.13
8
ar.L2         -0.7241      0.278     -2.606      0.009      -1.269      -0.18
0
sigma2      5896.0753   2735.819      2.155      0.031     533.969    1.13e+0
4
==========================================================================
======
Ljung-Box (L1) (Q):                   0.15   Jarque-Bera (JB):
2.32
Prob(Q):                              0.70   Prob(JB):
0.31
Heteroskedasticity (H):                inf   Skew:
-0.27
Prob(H) (two-sided):                  0.00   Kurtosis:
4.50
==========================================================================
======

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (comple
x-step).
```


United States Coal (TWh) -ARIMA Prediction (2, 0, 0) On Transformation
TWh
Year
Subtract Rolling Mean of Annual Change of United States Coal (TWh) -Historic
United States Coal (TWh) -ARIMA Prediction (2, 0, 0) On Transformation


United States Coal (TWh) -ARIMA Prediction (2, 0, 0) on Original
TWh
Year
Coal (TWh) -Historic
Coal (TWh) -ARIMA Prediction (2, 0, 0) on Original
IPCC Emissions 2030 Reduction Goal
IPCC Emissions 2050 Reduction Goal



This model looks very promising, but I want to test it through a grid search through the ARIMA models.

7.1.7 ARIMA Model and Grid Search

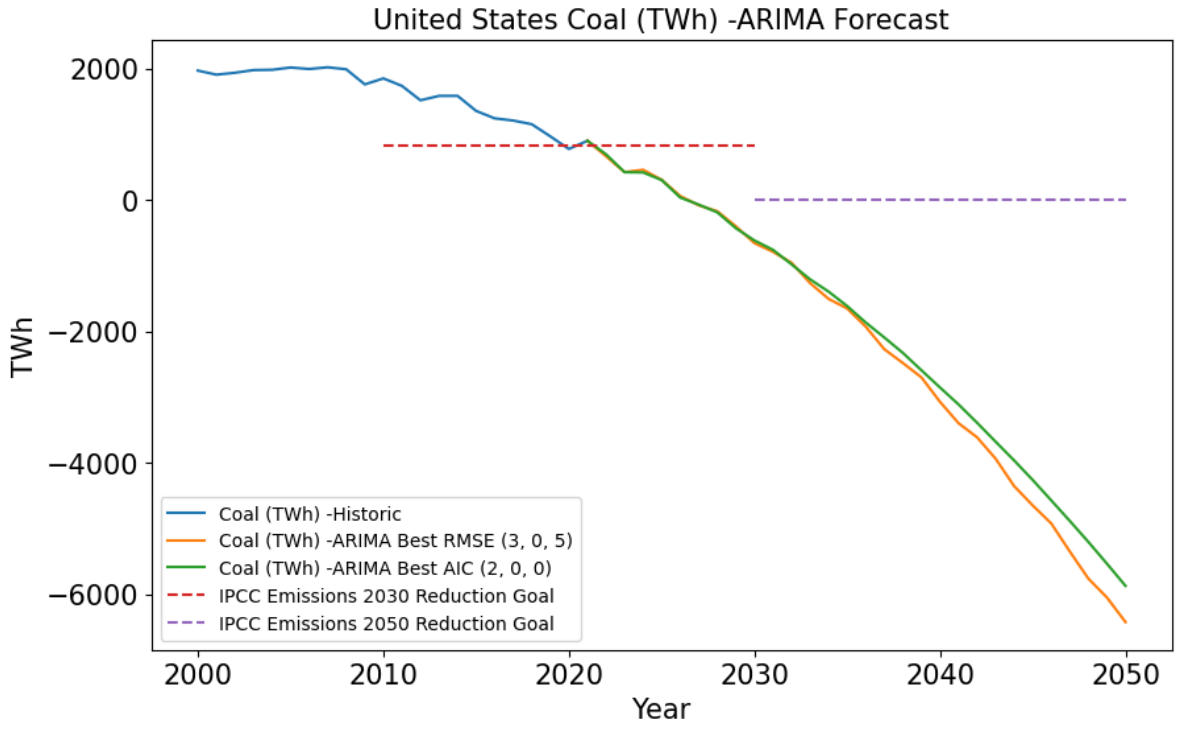
In [47]:
```
# Grid Search CV for transformed data.
def arima_pdq(df, st = 22, en = 50, s=0, tran=None, n=0):
    title = df.columns[0]

    # Initial Parameters
    best_rmse =  500000000000000000000000
    best_aic = 500000000000000000000000

    # I tried several iterations of the CV grid search. These were the maximum
    # parameters that yeilded good results.
    p_val = range(0,9)
    d_val = range(0,2)
    q_val = range(0,8)

    # Iterating through the various pdq values.
    for p in p_val:
        for d in d_val:
            for q in q_val:
                try:
                    order = (p, d, q)

                    # Creatign the model and gleaning stats.
                    model, yht, rmse, aic, df_bt, y_hat_bt, yho = mod_pred(df,
                                        order, 22, 50, tran=tran, n=n)

                    # Removing spurious results
                    if abs(y_hat_bt.iloc[-1].values[0]) > 10000:
                        continue

                    # Storing the pdq order if RMSE is the lowest seen so far.
                    if rmse < best_rmse:
                        best_rmse = rmse
                        best_rmse, rmse_aic, rmse_cfg = rmse, aic, order
                        rmse_2050 = y_hat_bt.iloc[-1].values[0]
                        print('RMSE ARIMA: {}, RMSE= {:.2f},'.format(rmse_cfg,
                                                                     best_rmse),
                              'AIC= {:.2f}, TWh-2050= {:.2f}'.format(rmse_aic,
                                                                    rmse_2050))
                    # Storing the pdq order if AIC is the lowest seen so far.
                    if aic < best_aic:
                        best_aic = aic
                        aic_rmse, best_aic, aic_cfg = rmse, aic, order
                        aic_2050 = y_hat_bt.iloc[-1].values[0]
                        if order == rmse_cfg:
                            continue
                        print('AIC  ARIMA: {}, RMSE= {:.2f},'.format(aic_cfg,
                                                                     aic_rmse),
                              'AIC= {:.2f}, TWh-2050= {:.2f}'.format(best_aic,
                                                                    aic_2050))
                except:
                    continue

    # Printing the results of the CV Search
    print('Best RMSE ARIMA: {} RMSE= {:.2f}'.format(rmse_cfg, best_rmse),
          'AIC= {:.2f}, TWh-2050= {:.2f}'.format(rmse_aic, rmse_2050))
    print('Best AIC  ARIMA: {} RMSE= {:.2f}'.format(aic_cfg, aic_rmse),
          'AIC= {:.2f}, TWh-2050= {:.2f}'.format(best_aic, aic_2050))
```

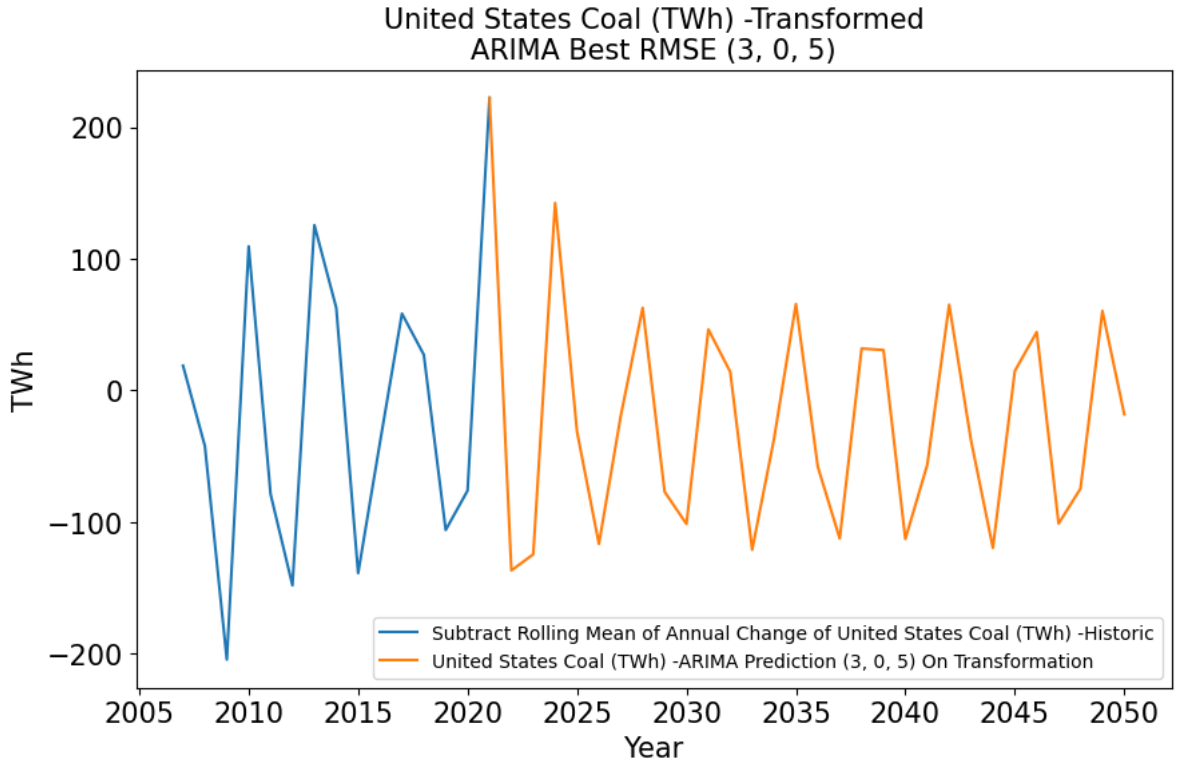
```
# Plotting predictions for both best RMSE and best AIC.
yh_t_rmse, yh_rmse_org_bt, yh_rmse = mod_pred_s(df, rmse_cfg, 22, 50,
                                                tran=tran, n=n)

yh_t_aic,  yh_aic_org_bt,  yh_aic  = mod_pred_s(df, aic_cfg,  22, 50,
                                                tran=tran, n=n)

# Remaming columns
yh_rmse.columns = ['Prediction']
yh_aic.columns = ['Prediction']
yh_rmse_org_bt.columns = [(df.columns[0] + ' -Transformed\nARIMA Best RMSE
                  str(rmse_cfg)  + ' -Test Projection')]
yh_aic_org_bt.columns = [(df.columns[0] + ' -Transformed\nARIMA Best AIC '
                  str(aic_cfg)  + ' -Test Projection')]

# Setting the first point in the prediction df to the same value as
# the last point in the historic df.
yh_rmse.loc[df.index[-1], yh_rmse.columns] = df.iloc[-1,0].copy()
yh_aic.loc[df.index[-1], yh_aic.columns] = df.iloc[-1,0].copy()

# Sorting index
yh_rmse.sort_index(inplace=True)
yh_aic.sort_index(inplace=True)
df.sort_index(inplace=True)

# Plotting figure
fig = plt.figure(figsize = (10,6))

# Original Data
plt.plot(df.index, df.iloc[:,0], label=str(df.columns[0])[s:]+' -Historic'

# Projection from start to end date
plt.plot(yh_rmse.index, yh_rmse.iloc[:,0],
         label=str(df.columns[0])[s:]+' -ARIMA Best RMSE ' + str(rmse_cfg)
plt.plot(yh_aic.index, yh_aic.iloc[:,0],
         label=str(df.columns[0])[s:]+' -ARIMA Best AIC ' + str(aic_cfg))

# Formatting the graph and printing the goal lines
if True:
    graph_formatting(df.iloc[:,0:1], yh_rmse.iloc[:,0:1], s)
#except:
    print('Graph_formatting produced error at',
          str(rmse_cfg) , 'or', str(aic_cfg))

plt.legend(loc='best')
plt.title(title + " -ARIMA Forecast", size=15)
plt.show()

# Plotting Transformed Figures for both estimates and test plots.
title_rmse =      (df.columns[0] + ' -Transformed\nARIMA Best RMSE '+
                  str(rmse_cfg))
title_rmse_test = (df.columns[0] + ' -Transformed\nARIMA Best RMSE '+
                  str(rmse_cfg)  + ' -Test Projection')
title_aic =       (df.columns[0] + ' -Transformed\nARIMA Best AIC '+
                  str(aic_cfg))
title_aic_test =  (df.columns[0] + ' -Transformed\nARIMA Best AIC '+
```

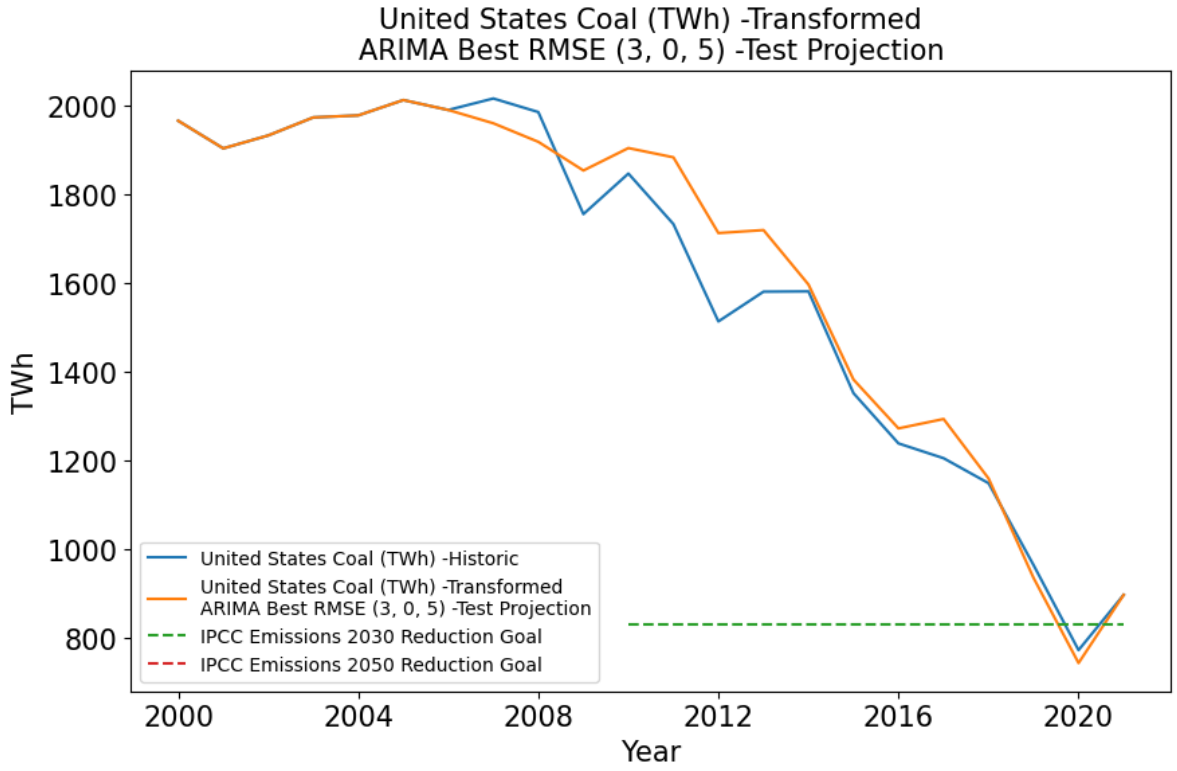
```
                    str(aic_cfg)  + ' -Test Projection')
    pred_graph(df.iloc[:,1:2], yh_t_rmse,  title=title_rmse)
    pred_graph(df.iloc[:,0:1], yh_rmse_org_bt, title=title_rmse_test)
    pred_graph(df.iloc[:,1:2], yh_t_aic,   title=title_aic)
    pred_graph(df.iloc[:,0:1], yh_aic_org_bt,  title=title_aic_test)


    return rmse_cfg, aic_cfg
```

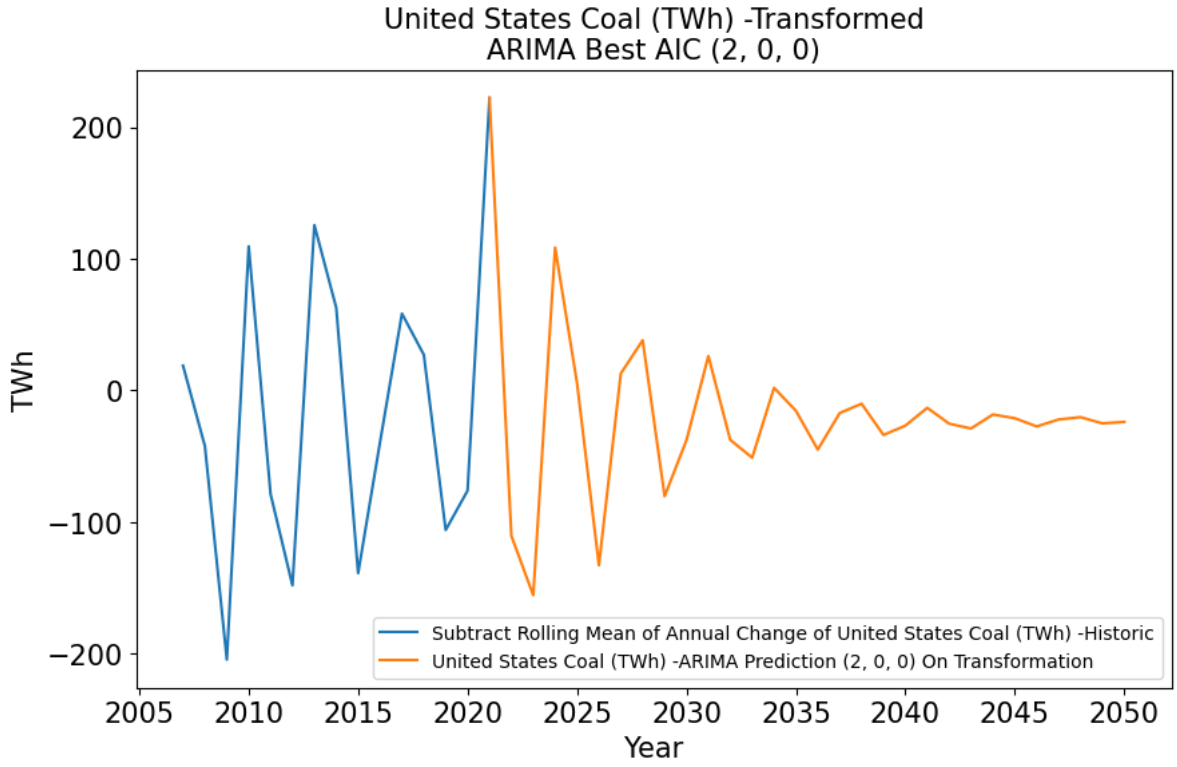
In [48]:
```
coal_rmse_cfg, coal_aic_cfg = arima_pdq(
                model_data_t.iloc[:,0:2], s=14, tran = 'inv_diff_inv_sub', n=7
```

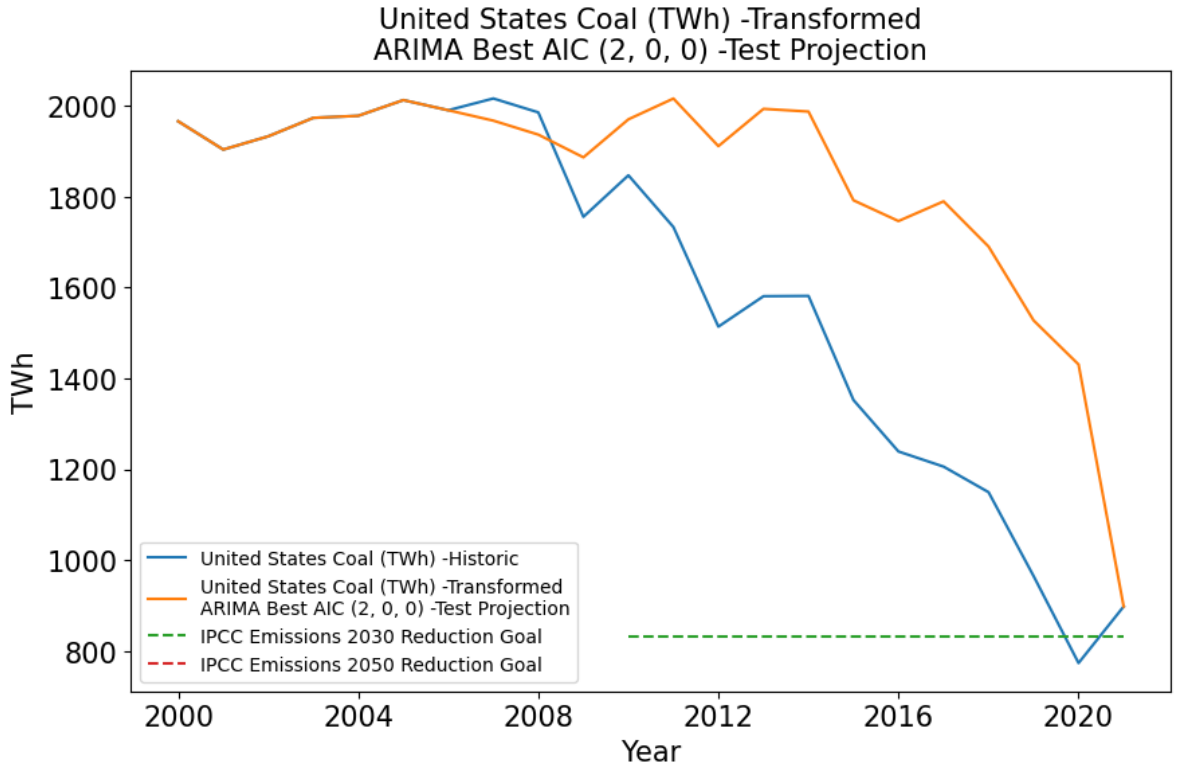
```
RMSE ARIMA: (0, 0, 0), RMSE= 406.03, AIC= 188.45, TWh-2050= -3197.65
AIC  ARIMA: (0, 0, 1), RMSE= 911.88, AIC= 184.47, TWh-2050= -6043.49
RMSE ARIMA: (0, 0, 3), RMSE= 270.90, AIC= 185.56, TWh-2050= -5841.74
RMSE ARIMA: (0, 0, 6), RMSE= 173.83, AIC= 188.29, TWh-2050= -6046.48
RMSE ARIMA: (0, 1, 3), RMSE= 138.26, AIC= 193.42, TWh-2050= 2028.69
RMSE ARIMA: (1, 0, 7), RMSE= 110.60, AIC= 190.91, TWh-2050= -5708.26
AIC  ARIMA: (2, 0, 0), RMSE= 424.96, AIC= 182.38, TWh-2050= -5874.97
RMSE ARIMA: (3, 0, 5), RMSE= 99.73, AIC= 192.37, TWh-2050= -6425.61
Best RMSE ARIMA: (3, 0, 5) RMSE= 99.73 AIC= 192.37, TWh-2050= -6425.61
Best AIC  ARIMA: (2, 0, 0) RMSE= 424.96 AIC= 182.38, TWh-2050= -5874.97
Graph_formatting produced error at (3, 0, 5) or (2, 0, 0)
```




United States Coal (TWh) -Transformed
ARIMA Best RMSE (3, 0, 5)
TWh
Year
Subtract Rolling Mean of Annual Change of United States Coal (TWh) -Historic
United States Coal (TWh) -ARIMA Prediction (3, 0, 5) On Transformation


United States Coal (TWh) -Transformed
ARIMA Best RMSE (3, 0, 5) -Test Projection
TWh
Year
United States Coal (TWh) -Historic
United States Coal (TWh) -Transformed
ARIMA Best RMSE (3, 0, 5) -Test Projection
IPCC Emissions 2030 Reduction Goal
IPCC Emissions 2050 Reduction Goal





I'm not sure I like either of these graphs as much as I the original, order (2,0,0). It just looks like a much better progression and estimate for how things will go. Regardless, one thing I like is that coal is going down and will likely not be used in a little more than a decade. Here's to hoping that trend will continue.

Also, all the models show that the United States will meet its reduction in coal use by 2030. That's also great news.

For Coal, the best pdq is (5, 5, 4).

7.1.8 ARIMA Without Transformation

In [49]:
```
# CV search when no backwards transformation is required.
def arima_pdq_no_tran(df, st, en, s=0):
    title = df.columns[0]
    print(title, 'Grid Search:')

    # Initial Parameters
    best_rmse =  500000000000000000000000
    best_aic = 500000000000000000000000

    # I tried several iterations of the CV grid search. These were the maximum
    # parameters that yeilded good results.
    p_val = range(0,9)
    d_val = range(0,4)
    q_val = range(0,9)

    # Going through each iteration of pdq
    for p in p_val:
        for d in d_val:
            for q in q_val:
                try:
                    order = (p, d, q)
                    # Build the model
                    model = ARIMA(df, order=order).fit()

                    # Build prediction of given data
                    y_hat_org = model.predict(start=0, end=(len(df)-1))

                    # Build projection
                    y_hat = model.predict(start=st, end=en)

                    # Removing spurious results
                    if abs(y_hat.iloc[-1]) > 10000:
                        continue

                    # Make the first and last point of original prediction the
                    # same as the first point in the dataset to not throw off
                    # the rmse score
                    y_hat_org[0] =  df.iloc[0,0:1].values[0]
                    y_hat_org[-1] = df.iloc[-1,0:1].values[0]

                    # Take measurements
                    rmse = sk.metrics.mean_squared_error(
                        df, y_hat_org, squared = False)
                    aic  = model.aic

                    # Store the order if RMSE is the lowest seen so far.
                    if rmse < best_rmse:
                        best_rmse = rmse
                        rmse_aic, rmse_cfg = aic, order
                        rmse_2050 = y_hat.iloc[-1]
                        print('RMSE ARIMA: {}, RMSE= {:.2f},'.format(rmse_cfg,
                                                                     best_rmse),
                              'AIC= {:.2f}, TWh-2050= {:.2f}'.format(rmse_aic,
                                                                    rmse_2050))

                    # Storing the order if AIC is the lowest seen so far.
                    if aic < best_aic:
```

```
                        best_aic = aic
                        aic_rmse, aic_cfg = rmse, order
                        aic_2050 = y_hat.iloc[-1]
                        if order == rmse_cfg:
                            continue
                        print('AIC  ARIMA: {}, RMSE= {:.2f},'.format(aic_cfg,
                                                                     aic_rmse),
                              'AIC= {:.2f}, TWh-2050= {:.2f}'.format(best_aic,
                                                                    aic_2050))
                except:
                    continue

    # Printing the orders and stats of the best RMSE and AIC results.
    print('Best RMSE ARIMA: {} RMSE= {:.2f}'.format(rmse_cfg, best_rmse),
         'AIC= {:.2f}, TWh-2050= {:.2f}'.format(rmse_aic, rmse_2050))
    print('Best AIC  ARIMA: {} RMSE= {:.2f}'.format(aic_cfg, aic_rmse),
         'AIC= {:.2f}, TWh-2050= {:.2f}'.format(best_aic, aic_2050))

    # Creating final model, predictions and projection over given data for bot
    mod_rmse = ARIMA(df, order = rmse_cfg).fit()
    yh_rmse = pd.DataFrame(mod_rmse.predict(st, en))
    yh_rmse_org = pd.DataFrame(mod_rmse.predict(0, len(df)-1))
    yh_rmse.columns = ['Prediction']
    yh_rmse_org.columns = [(df.columns[0] + ' -Transformed\nARIMA Best RMSE '+
                        str(rmse_cfg)  + ' -Test Projection')]

    mod_aic = ARIMA(df, order = aic_cfg).fit()
    yh_aic = pd.DataFrame(mod_aic.predict(st, en))
    yh_aic_org = pd.DataFrame(mod_aic.predict(0, len(df)-1))
    yh_aic.columns = ['Prediction']
    yh_aic_org.columns = [(df.columns[0] + ' -Transformed\nARIMA Best AIC '+
                        str(aic_cfg)  + ' -Test Projection')]

    # Setting the first point in the prediction df to the same value as
    # the last point in the historic df.
    yh_rmse.loc[df.index[-1], yh_rmse.columns] = df.iloc[-1,0].copy()
    yh_aic.loc[df.index[-1], yh_aic.columns] = df.iloc[-1,0].copy()

    # Setting first point of the projection over given data to visually check
    # what was calculated for the RMSE score.
    yh_rmse_org.iloc[0,0:1]  = df.iloc[0,0:1].values[0]
#    yh_rmse_org.iloc[-1,0:1] = df.iloc[-1,0:1].values[0]
    yh_aic_org.iloc[0,0:1]   = df.iloc[0,0:1].values[0]
#    yh_aic_org.iloc[-1,0:1]  = df.iloc[-1,0:1].values[0]

    # Sorting index
    yh_rmse.sort_index(inplace=True)
    yh_aic.sort_index(inplace=True)
    df.sort_index(inplace=True)

    # Plotting figure
    fig = plt.figure(figsize = (10,6))
    plt.plot(df.index, df.iloc[:,0], label=str(df.columns[0])[s:]+' -Historic'
    plt.plot(yh_rmse.index, yh_rmse.iloc[:,0],
             label=str(df.columns[0])[s:]+' -ARIMA Best RMSE ' + str(rmse_cfg)
    plt.plot(yh_aic.index, yh_aic.iloc[:,0],
             label=str(df.columns[0])[s:]+' -ARIMA Best AIC ' + str(aic_cfg))
```

```
    plt.legend(loc='best')
    plt.title(title + " ARIMA Forecast")

    graph_formatting(df.iloc[:,0:1], yh_rmse.iloc[:,0:1], s)

    plt.legend(loc='best')
    plt.title(title + " ARIMA Forecast", size = 15)

    plt.show()

    # Plotting projection over given data, the test plot...
    title_rmse_test = (df.columns[0] + ' -Transformed\nARIMA Best RMSE '+
                       str(rmse_cfg)  + ' -Test Projection')
    title_aic_test =  (df.columns[0] + ' -Transformed\nARIMA Best AIC '+
                       str(aic_cfg)  + ' -Test Projection')
    pred_graph(df.iloc[:,0:1], yh_rmse_org, title=title_rmse_test)
    pred_graph(df.iloc[:,0:1], yh_aic_org,  title=title_aic_test)

    return rmse_cfg, aic_cfg
```

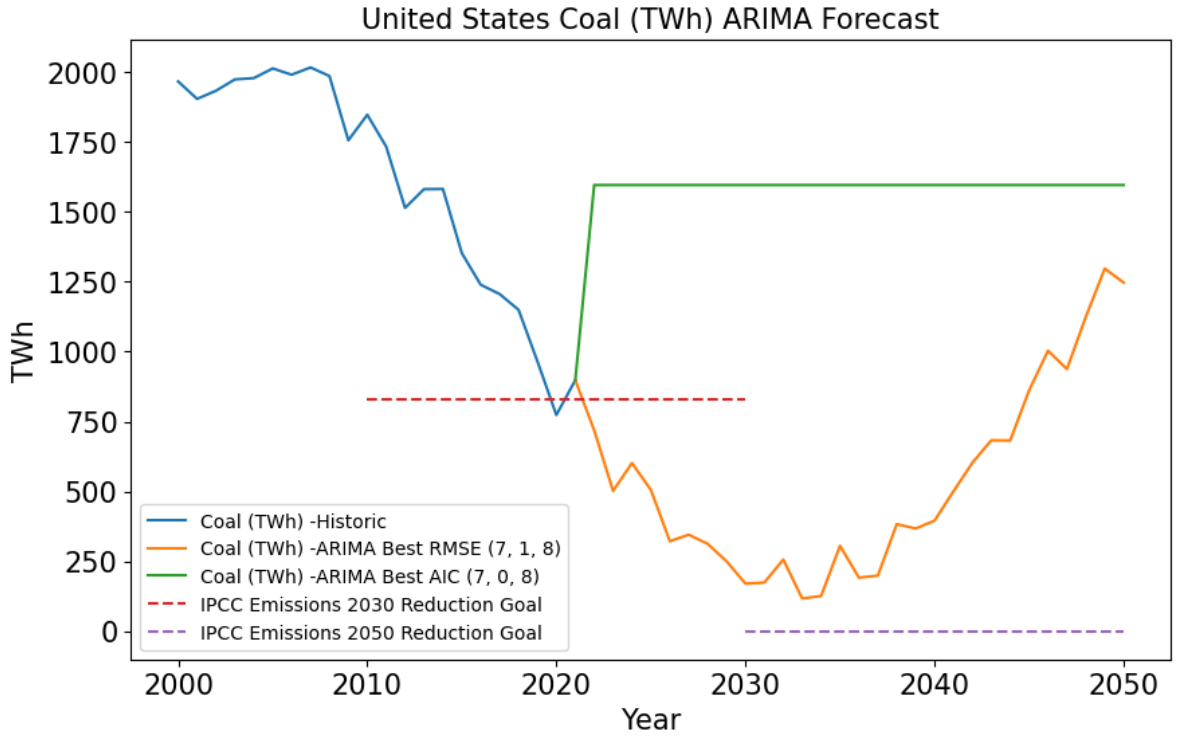
In [50]:

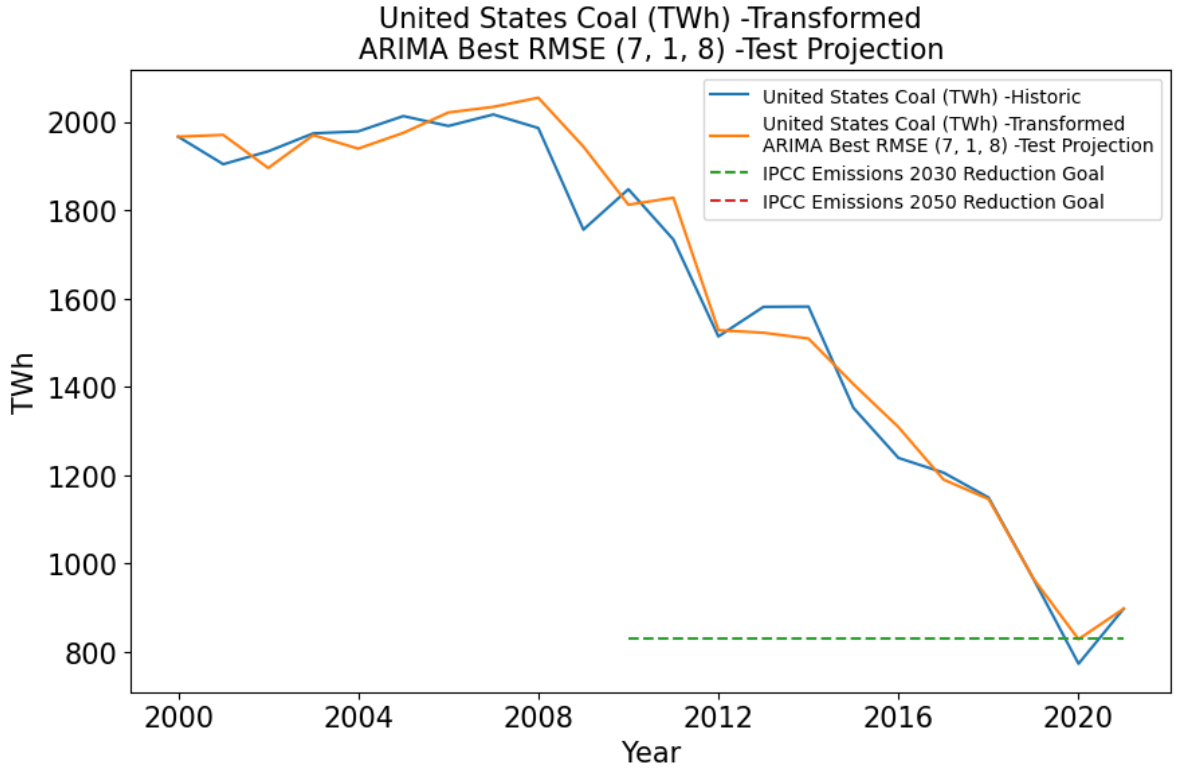
```
# This is the cousin of mod_pred. It creates the model and projects the model.
def arima_no_tran(df, o, st, en, s=0):
    model = ARIMA(df, order=o).fit()
    y_hat_org = model.predict(start=0, end=(len(df)-1))
    y_hat = model.predict(start=st, end=en)

    # Adding the end point of the original data to the y_hat DataFrame
    y_hat = pd.DataFrame(y_hat)
    y_hat.loc[df.index[-1], y_hat.columns] = df.iloc[-1,0]
    y_hat.sort_index(inplace=True)

    # Setting the first point of the original prediction to the same as the
    # last point of data to not throw off the RMSE
    y_hat_org[0] = df.iloc[0,0:1].values[0]

    # Creating DataFrames for the y-hats
    y_hat_org = pd.DataFrame(y_hat_org)

    model.name = df.columns[0] + ' -ARIMA Prediction '+str(o)

    # Setting Column names for y-hat
    y_hat_org.columns = [(df.columns[0] + ' ARIMA Prediction '+str(o) +
                          '\nPrediction on Original Data')]

    y_hat.columns =  [df.columns[0] + ' ARIMA Prediction '+str(o)]

    return model, y_hat, y_hat_org
```

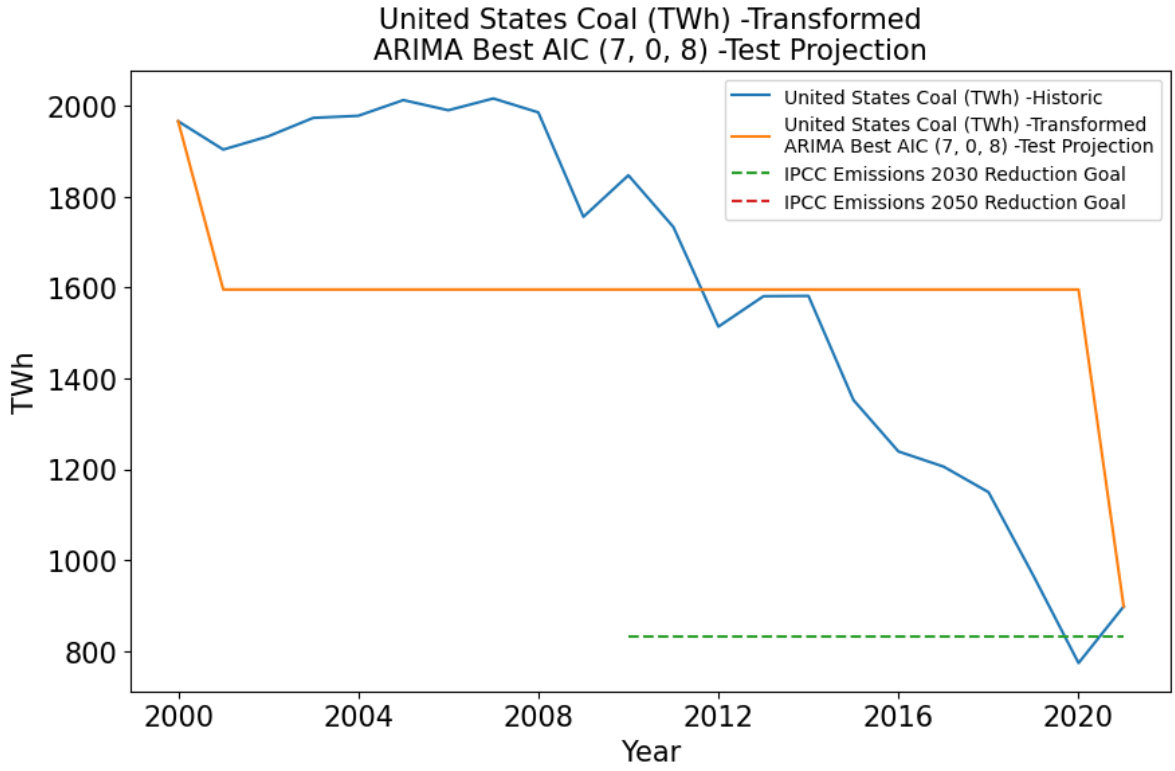
In [51]:
```
# The cousin function to model_summary. This creates the model, projects,
# calculates RMSE and AIC, and plots the projection and test plot.
def model_summary_no_tran(df, o, st, en, s=0):
    model, y_hat, y_hat_org = arima_no_tran(df, o, st, en, s=s)

    # Calculate the RMSE & AIC
    rmse = sk.metrics.mean_squared_error(
                        df, y_hat_org, squared = False)
    aic  = model.aic

    print()
    print('ARIMA: {}'.format(o) + ', RMSE={:.2f}, AIC={:.2f}'.format(rmse,
                                                                     aic))
    summary_print(model)

    pred_graph(df.iloc[:,0:1], y_hat.iloc[:,0:1], s)
    pred_graph(df, y_hat_org, s)


    return y_hat, y_hat_org
```

In [52]:
```
coal_rmse_cfg, coal_aic_cfg = arima_pdq_no_tran(model_data.iloc[:,0:1], 22, 50
```

```
United States Coal (TWh) Grid Search:
RMSE ARIMA: (0, 0, 0), RMSE= 361.38, AIC= 329.96, TWh-2050= 1607.17
RMSE ARIMA: (0, 0, 1), RMSE= 208.99, AIC= 308.34, TWh-2050= 1604.78
RMSE ARIMA: (0, 0, 2), RMSE= 155.05, AIC= 296.05, TWh-2050= 1606.95
RMSE ARIMA: (0, 0, 3), RMSE= 143.30, AIC= 294.81, TWh-2050= 1606.74
RMSE ARIMA: (0, 0, 4), RMSE= 131.30, AIC= 293.67, TWh-2050= 1602.29
RMSE ARIMA: (0, 0, 5), RMSE= 112.24, AIC= 289.08, TWh-2050= 1605.38
RMSE ARIMA: (0, 0, 7), RMSE= 92.09, AIC= 286.09, TWh-2050= 1605.92
RMSE ARIMA: (0, 0, 8), RMSE= 91.25, AIC= 287.03, TWh-2050= 1605.93
AIC  ARIMA: (0, 1, 0), RMSE= 112.20, AIC= 261.97, TWh-2050= 897.89
RMSE ARIMA: (0, 1, 3), RMSE= 83.86, AIC= 256.50, TWh-2050= 775.59
RMSE ARIMA: (0, 1, 4), RMSE= 82.42, AIC= 258.23, TWh-2050= 817.77
RMSE ARIMA: (0, 1, 7), RMSE= 76.35, AIC= 261.77, TWh-2050= 687.31
RMSE ARIMA: (0, 1, 8), RMSE= 75.83, AIC= 263.36, TWh-2050= 746.47
AIC  ARIMA: (0, 2, 1), RMSE= 244.00, AIC= 251.28, TWh-2050= -903.00
AIC  ARIMA: (0, 2, 2), RMSE= 238.99, AIC= 249.26, TWh-2050= -1910.20
AIC  ARIMA: (0, 2, 3), RMSE= 238.23, AIC= 247.91, TWh-2050= -985.13
AIC  ARIMA: (0, 2, 4), RMSE= 236.50, AIC= 246.39, TWh-2050= -644.09
AIC  ARIMA: (0, 3, 2), RMSE= 455.81, AIC= 245.56, TWh-2050= 4559.59
AIC  ARIMA: (0, 3, 3), RMSE= 454.36, AIC= 243.64, TWh-2050= -1039.46
RMSE ARIMA: (1, 1, 7), RMSE= 74.52, AIC= 262.24, TWh-2050= 652.61
RMSE ARIMA: (2, 1, 7), RMSE= 72.68, AIC= 263.08, TWh-2050= 409.75
RMSE ARIMA: (2, 1, 8), RMSE= 71.44, AIC= 266.56, TWh-2050= 1.90
AIC  ARIMA: (2, 3, 1), RMSE= 455.42, AIC= 242.92, TWh-2050= -2527.44
AIC  ARIMA: (2, 3, 2), RMSE= 454.77, AIC= 240.50, TWh-2050= 137.39
RMSE ARIMA: (3, 1, 7), RMSE= 69.61, AIC= 263.09, TWh-2050= 74.04
RMSE ARIMA: (4, 1, 8), RMSE= 68.42, AIC= 269.06, TWh-2050= 256.42
AIC  ARIMA: (4, 3, 0), RMSE= 454.34, AIC= 238.80, TWh-2050= 1603.58
RMSE ARIMA: (5, 1, 2), RMSE= 65.08, AIC= 255.76, TWh-2050= -341.57
RMSE ARIMA: (5, 1, 3), RMSE= 64.62, AIC= 258.54, TWh-2050= -534.77
RMSE ARIMA: (6, 0, 2), RMSE= 64.54, AIC= 274.17, TWh-2050= 2111.11
RMSE ARIMA: (6, 1, 2), RMSE= 63.64, AIC= 259.30, TWh-2050= -1055.23
RMSE ARIMA: (6, 1, 3), RMSE= 63.12, AIC= 259.54, TWh-2050= 889.98
RMSE ARIMA: (6, 1, 8), RMSE= 63.02, AIC= 270.87, TWh-2050= 1673.05
RMSE ARIMA: (7, 0, 3), RMSE= 61.77, AIC= 277.05, TWh-2050= 2271.53
AIC  ARIMA: (7, 0, 8), RMSE= 362.04, AIC= 34.00, TWh-2050= 1595.81
RMSE ARIMA: (7, 1, 8), RMSE= 60.38, AIC= 269.23, TWh-2050= 1246.97
Best RMSE ARIMA: (7, 1, 8) RMSE= 60.38 AIC= 269.23, TWh-2050= 1246.97
Best AIC  ARIMA: (7, 0, 8) RMSE= 362.04 AIC= 34.00, TWh-2050= 1595.81
```


United States Coal (TWh) ARIMA Forecast
TWh
Year
Coal (TWh) -Historic
Coal (TWh) -ARIMA Best RMSE (7, 1, 8)
Coal (TWh) -ARIMA Best AIC (7, 0, 8)
IPCC Emissions 2030 Reduction Goal
IPCC Emissions 2050 Reduction Goal


United States Coal (TWh) -Transformed
ARIMA Best RMSE (7, 1, 8) -Test Projection
TWh
Year
United States Coal (TWh) -Historic
United States Coal (TWh) -Transformed
ARIMA Best RMSE (7, 1, 8) -Test Projection
IPCC Emissions 2030 Reduction Goal
IPCC Emissions 2050 Reduction Goal



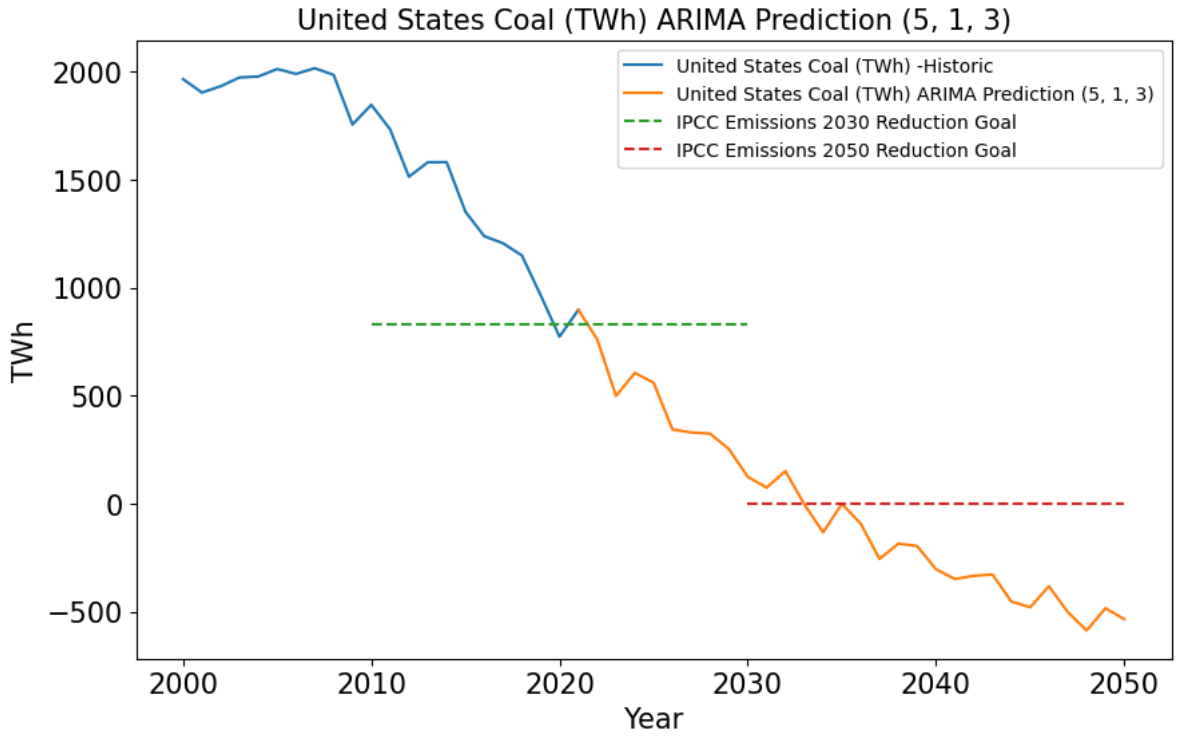
7.1.9 Model Selection

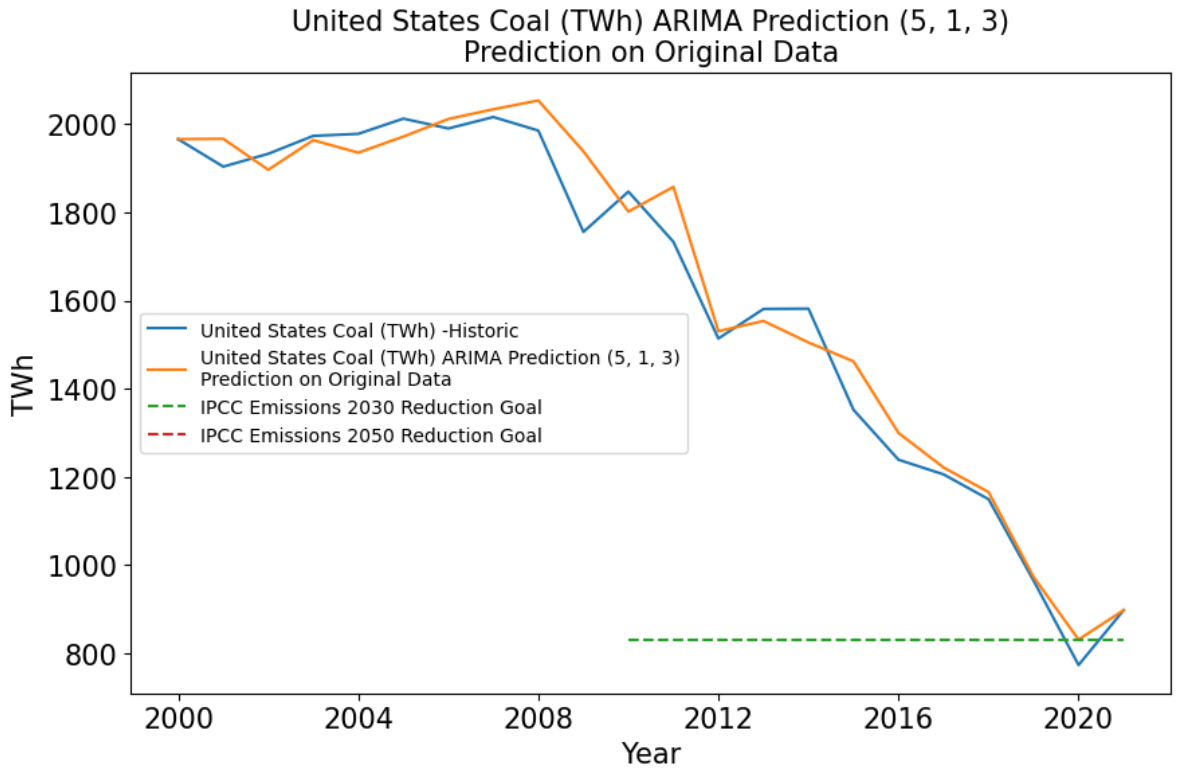
The cell below creates the first list in a list that will be updated throughout the out the notebook. The list includes information that will be read by a custom function to create DataFrame of predicted values. The DataFrame created can be passed to another function to plot the stacked energy graph for the future.

In [53]:
```
df_coal = model_data_t.iloc[:,0:1]

#                   Model,   df_o,     pdq,    tran, n, PDQs
selected_models = [['ARIMA', df_coal, (5,1,3), None, 0, None]]

m = 0
model_summary_no_tran(selected_models[m][1], selected_models[m][2], 22, 50);
```

```
ARIMA: (5, 1, 3), RMSE=65.33, AIC=258.54
**********************************************************************
**
United States Coal (TWh) -ARIMA Prediction (5, 1, 3) model results.

                                SARIMAX Results
==========================================================================
=======
Dep. Variable:     United States Coal (TWh)   No. Observations:
22
Model:                        ARIMA(5, 1, 3)   Log Likelihood               -
120.272
Date:                       Sat, 29 Apr 2023   AIC
258.544
Time:                             06:16:18   BIC
267.945
Sample:                         01-01-2000   HQIC
260.584
                              - 01-01-2021
Covariance Type:                       opg
==========================================================================
=
                 coef    std err          z      P>|z|      [0.025      0.97
5]
--------------------------------------------------------------------------
-
ar.L1         -0.7230      0.323     -2.239      0.025      -1.356      -0.09
0
ar.L2         -0.8289      0.256     -3.238      0.001      -1.331      -0.32
7
ar.L3          0.7030      0.266      2.642      0.008       0.181       1.22
5
ar.L4          0.6240      0.291      2.146      0.032       0.054       1.19
4
ar.L5          0.8749      0.173      5.048      0.000       0.535       1.21
5
ma.L1          0.8378      2.243      0.374      0.709      -3.558       5.23
3
ma.L2          0.8994      4.240      0.212      0.832      -7.412       9.21
1
ma.L3         -0.0598      0.576     -0.104      0.917      -1.188       1.06
8
sigma2      2705.2156   1.14e+04      0.237      0.813   -1.97e+04    2.51e+0
4
==========================================================================
======
Ljung-Box (L1) (Q):                   0.39   Jarque-Bera (JB):
2.25
Prob(Q):                              0.53   Prob(JB):
0.33
Heteroskedasticity (H):               6.19   Skew:
-0.79
Prob(H) (two-sided):                  0.03   Kurtosis:
3.25
==========================================================================
======
```

```
Warnings:
[1] Covariance matrix calculated using the outer product of gradients (comple
x-step).
```





This model is a good mix of decent RMSE for this scale and likely behavior of energy production. The AIC is not the best, but, for this energy source, the models with good AIC do not give realistic results.

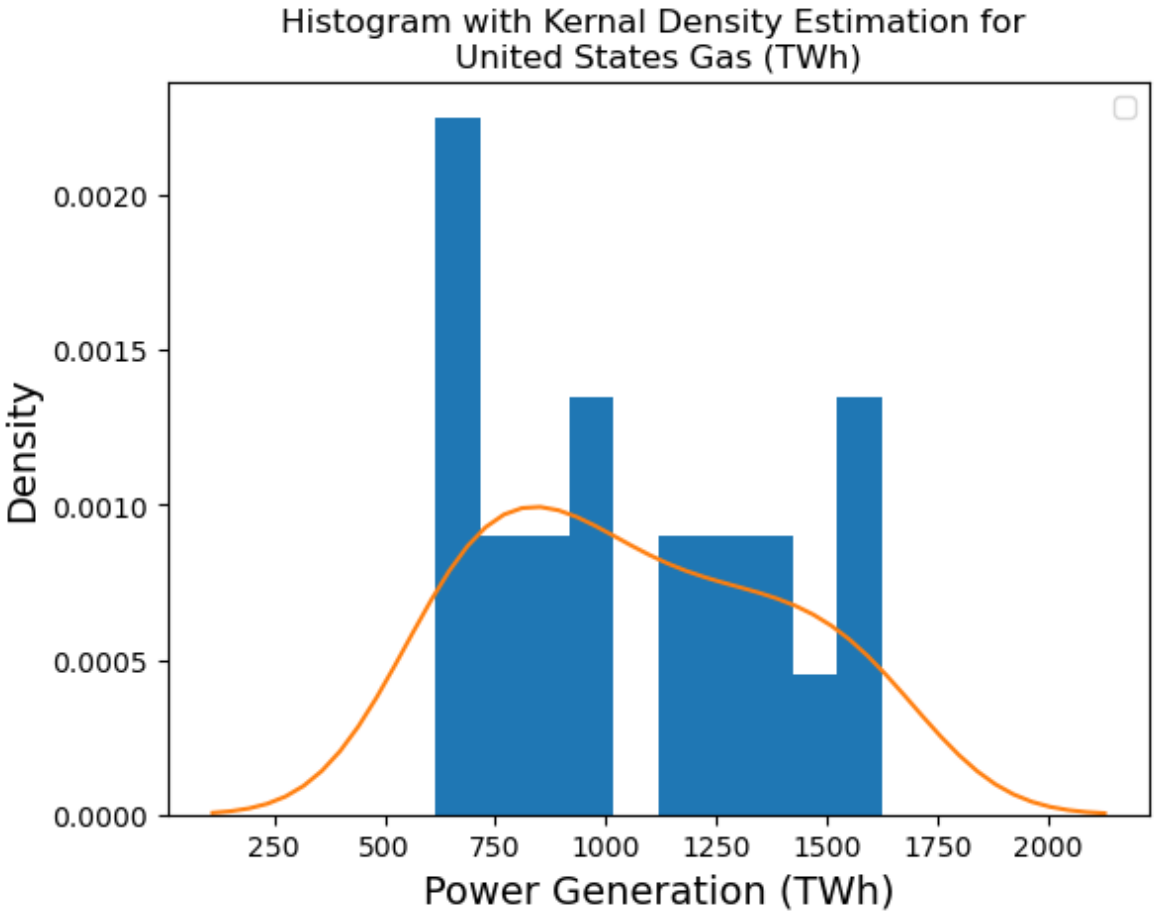
7.2 Gas

7.2.1 Distribution Investigation

In [54]:
```
stored_model_data = model_data.copy()
stored_model_data_t = model_data_t.copy()
```

In [55]:
```
model_data = stored_model_data.copy()
model_data_t = stored_model_data_t.copy()
```

In [56]:
```
hist(model_data.iloc[:,1:2])
```



In [57]:
```
stats_block = s_block(model_data.iloc[:,1:2], stats_block)
stats_block
```

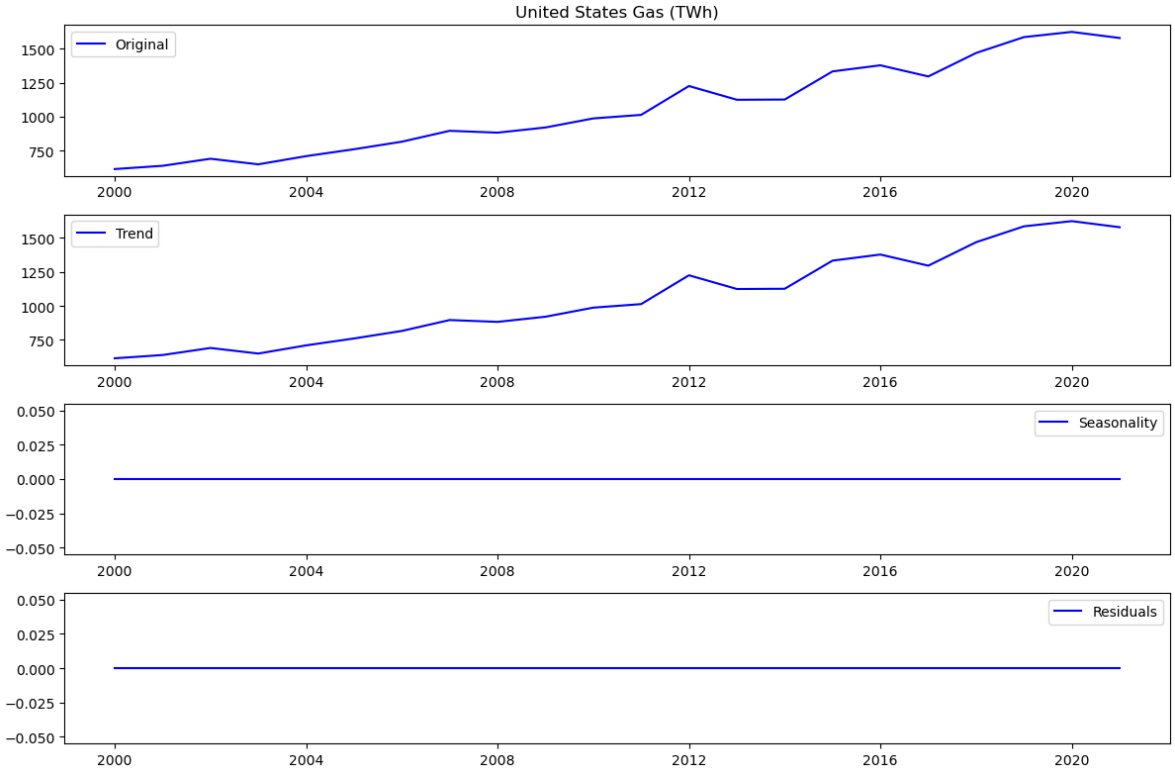
Out[57]:

	Dataset	Mean	Median	Standard Deviation	Skew	Fisher Kurtosis
0	United States Coal (TWh)	1,607.17	1,744.66	399.15	-0.68	-0.90
1	United States Gas (TWh)	1,060.39	1,000.70	325.82	0.29	-1.21

A somewhat of symmetrical distribution with flat tails. It's skewed positively slightly.

7.2.2 Decomposing The Data

In [58]:
```
decomp_graph(model_data.iloc[:,1:2])
```

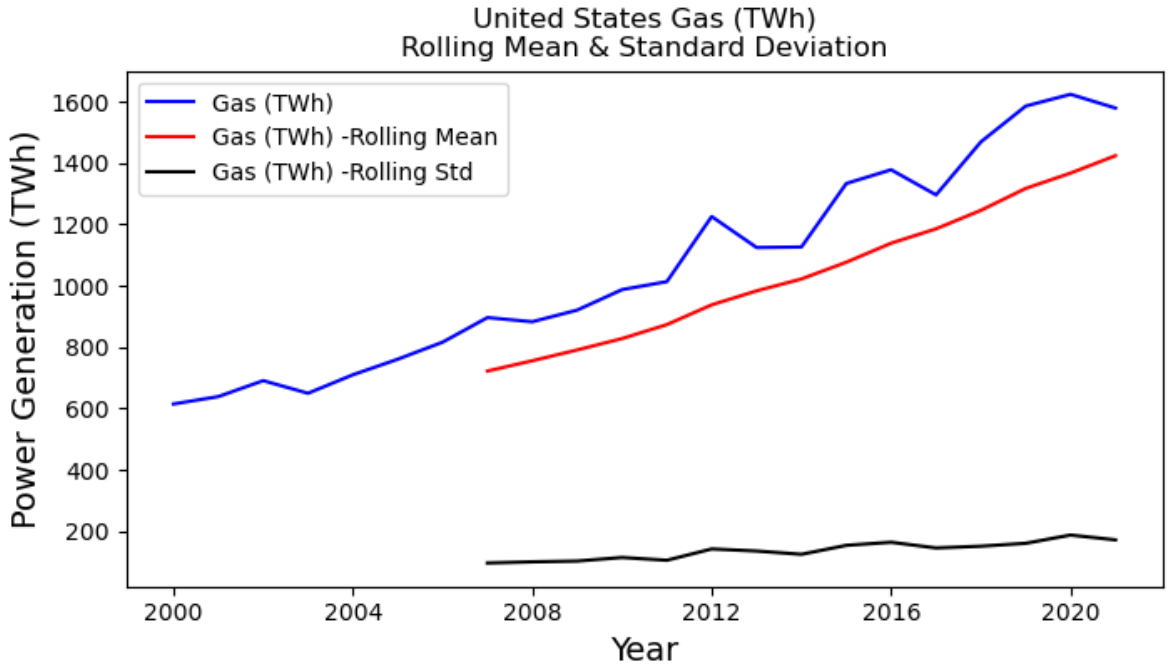


No seasonality or residuals. I'll check for stationarity.

7.2.3 Checking for Stationarity and Flattening

In [59]:
```
pd.set_option('display.max_rows', 30)
stationarity_check(model_data.iloc[:,1:2], s=14)
```

```
United States Gas (TWh)
Results of Dickey-Fuller test:

Test Statistic                  3.30
p-value                         1.00
#Lags Used                      7.00
Number of Observations Used    14.00
Critical Value (1%)            -4.01
Critical Value (5%)            -3.10
Critical Value (10%)           -2.69
dtype: float64
```



This is not close to stationary. After playing around with transformations, I found that if I took the log and then subtracted the rolling average of four values, I can get it stationary.

In [60]:
```
# Takes the log of the data for transformation purposes.
def log_transform(df):
    df_np = df.iloc[:,0:1].values[:,0]
    logged_df = pd.DataFrame(np.log(df_np), index = df.index,
                             columns = 'Natural Logged ' + df.columns)

    return logged_df
```

In [61]:
```
t = subtract_roll_mean(log_transform(model_data.iloc[:,1:2]), n=4)
model_data_t.insert(3,t.columns[0],t)

model_data_t.iloc[:,2:4].head()
```

Out[61]:

Year	United States Gas (TWh)	Subtract Rolling Mean of Natural Logged United States Gas (TWh)
2000-01-01	614.99	NaN
2001-01-01	639.13	NaN
2002-01-01	691.01	NaN
2003-01-01	649.91	0.00
2004-01-01	710.10	0.06

Before I move forward, I want to make sure I can transform this data back after modeling.

In [62]:
```
# Inverses the log of the column, also known as exponent.
def inv_log_transform(df):
    l = len(df.columns)
    df_np = df.iloc[:,l-1:l].values[:,0]

    exp_df = pd.DataFrame(np.exp(df_np), index = df.index,
                          columns = ['Back Transformation of ' + df.columns[0]])

    return exp_df
```

In [63]:
```
# Inverting the data when both log and subtract rolling mean were done to the
# data.
def inv_log_inv_sub(df, n=0):
    test = df.copy()
    # Recreating the original log transformation to pass to the inverse
    # subtract rolling mean function.
    t = log_transform(test.iloc[:,0:1])
    test[str(t.columns[0])] = t
    test = test.iloc[:,[0,2,1]]
    # Reversing the subtract rolling mean function.
    t = inv_subtract_roll_mean(test.iloc[:,1:], n=n, ex_copy=1)
    test[str(t.columns[0])] = t
    # Inversing the log.
    t = inv_log_transform(test.iloc[:,[0,3]])
    test[str(t.columns[0])] = t
    test.rename(columns={test.iloc[:,4:5].columns[0]:
            'Back Transformation of ' + df.iloc[:,1:2].columns[0]},inplace=1)

    return test.iloc[:,4:5]
```

In [64]:
```
test = model_data_t.iloc[:,2:4].copy()

test['Results'] = inv_log_inv_sub(test,n=4)
test
```

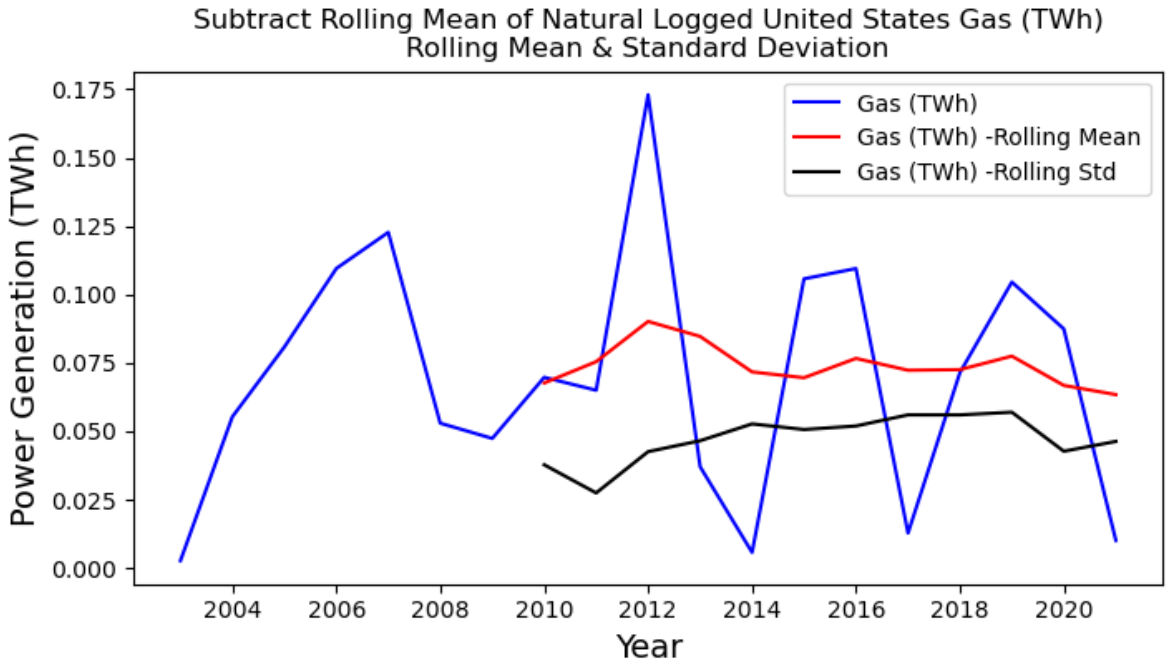
Out[64]:

Year	United States Gas (TWh)	Subtract Rolling Mean of Natural Logged United States Gas (TWh)	Results
2000-01-01	614.99	NaN	614.99
2001-01-01	639.13	NaN	639.13
2002-01-01	691.01	NaN	691.01
2003-01-01	649.91	0.00	649.91
2004-01-01	710.10	0.06	710.10
2005-01-01	760.96	0.08	760.96
2006-01-01	816.44	0.11	816.44
2007-01-01	896.59	0.12	896.59
2008-01-01	882.98	0.05	882.98
2009-01-01	920.98	0.05	920.98
2010-01-01	987.70	0.07	987.70
2011-01-01	1,013.69	0.06	1,013.69
2012-01-01	1,225.89	0.17	1,225.89
2013-01-01	1,124.84	0.04	1,124.84
2014-01-01	1,126.61	0.01	1,126.61
2015-01-01	1,333.48	0.11	1,333.48
2016-01-01	1,378.31	0.11	1,378.31
2017-01-01	1,296.44	0.01	1,296.44
2018-01-01	1,469.13	0.07	1,469.13
2019-01-01	1,585.81	0.10	1,585.81
2020-01-01	1,624.17	0.09	1,624.17
2021-01-01	1,579.36	0.01	1,579.36

Transferring back is a possibility. Therefore, I can move forward. I'll again check for stationarity.

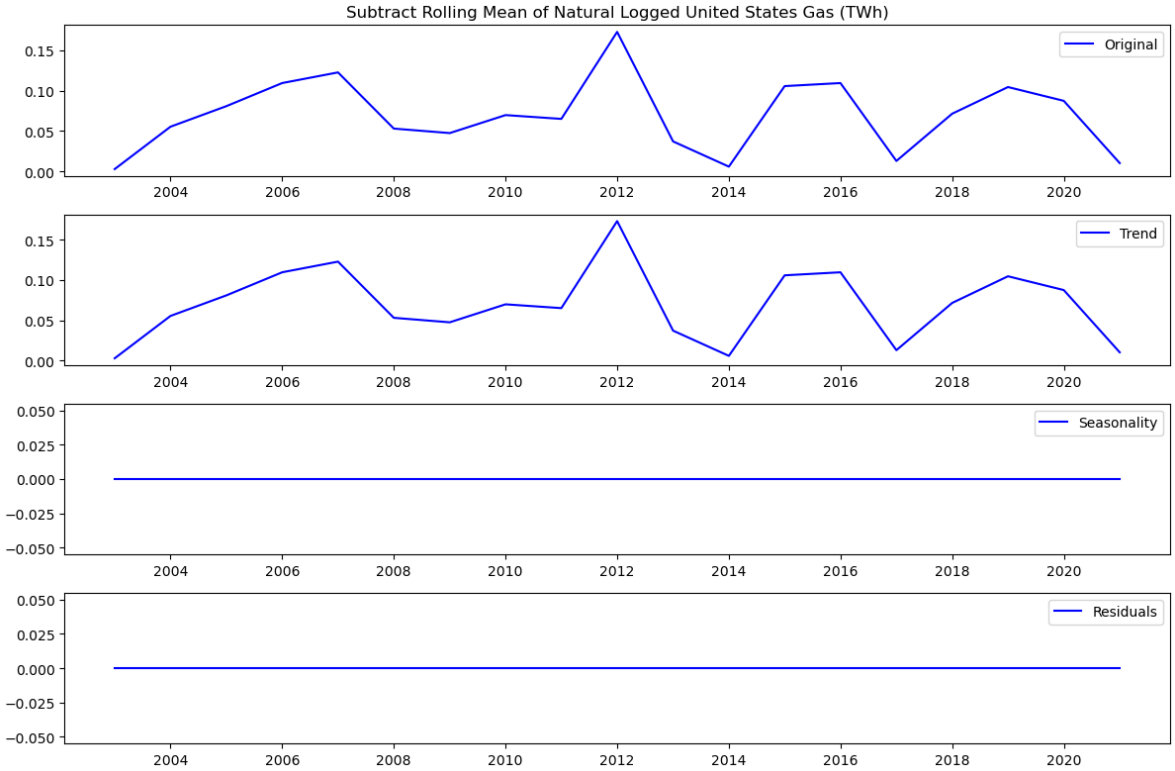
In [65]:
```
stationarity_check(model_data_t.iloc[:,3:4], s=54)
```

```
Subtract Rolling Mean of Natural Logged United States Gas (TWh)
Results of Dickey-Fuller test:

Test Statistic                -5.54
p-value                        0.00
#Lags Used                     1.00
Number of Observations Used   17.00
Critical Value (1%)           -3.89
Critical Value (5%)           -3.05
Critical Value (10%)          -2.67
dtype: float64
```



My p-value is less than .05. Stationarity achieved. I'll look at the decomposition graph again.

In [66]:
```
decomp_graph(model_data_t.iloc[:,3:4].dropna())
```

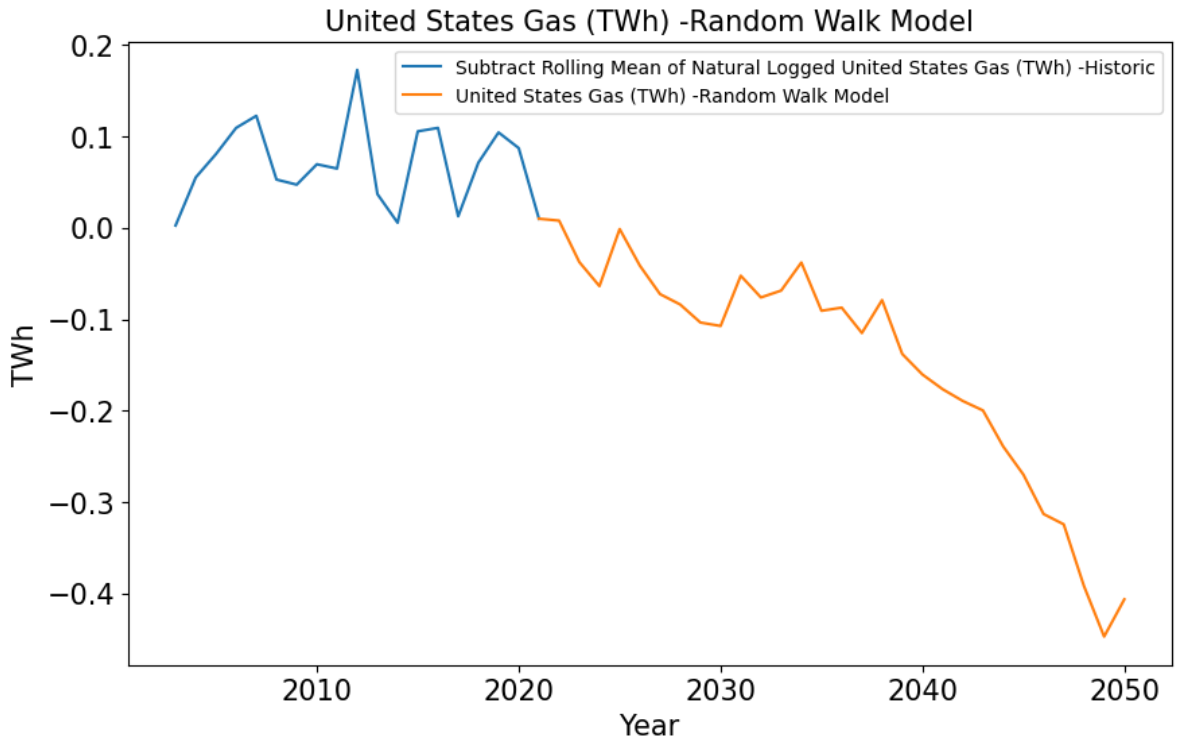


Looks like there's still a trend line, but it does pass stationarity, so I'm moving forward to the baseline random walk model.

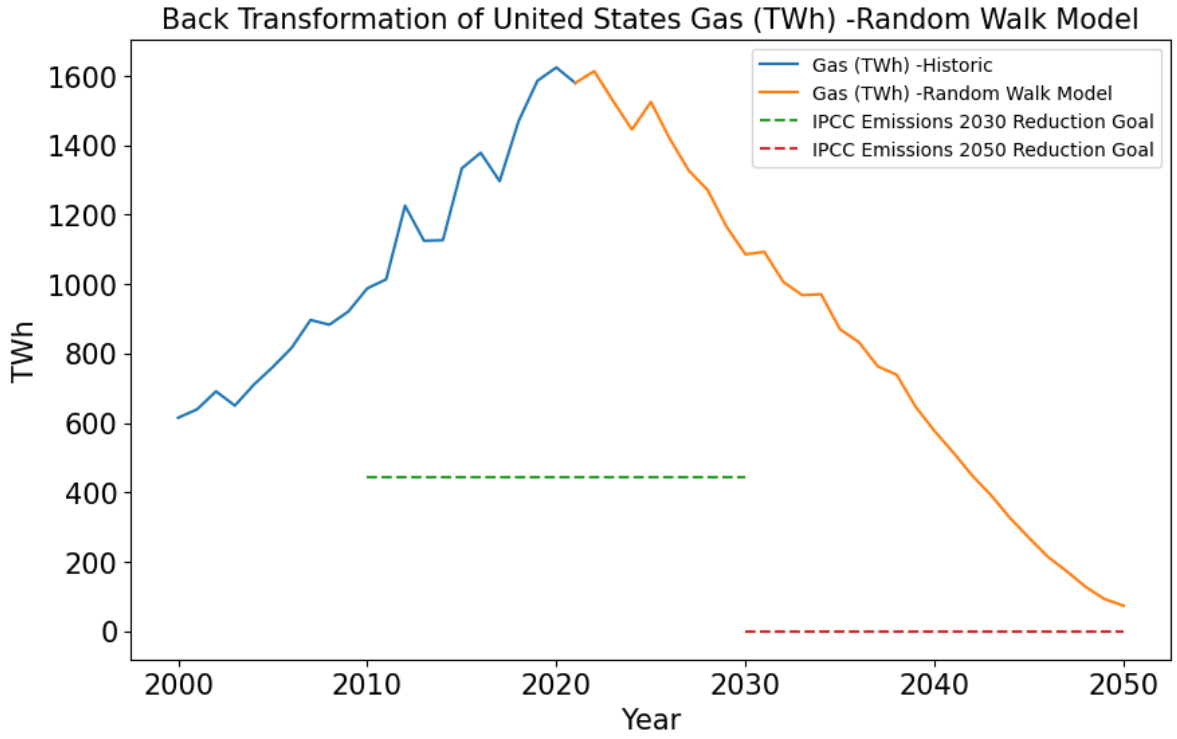
7.2.4 Random Walk Model

In [67]:
```
pd.set_option('display.max_rows',None)
y_hat_proj = random_walk(model_data_t.iloc[:,3:4])
y_hat_proj.columns = [model_data.columns[1]+' -Random Walk Model']

# Plot the transformed Random Walk
pred_graph(model_data_t.iloc[:,3:4], y_hat_proj.iloc[:,0:1])
```

```
Random Walk with Standard Deviation of Transformed Data set: 0.04
```



In [68]:
```
# Prepare the DataFrame for Back Transformation
ranplot = model_data_t.iloc[:,2:4].copy()
ranplot.rename(columns={str(ranplot.columns[1]): str(y_hat_proj.columns[0])},
               inplace=True)

ranplot = pd.concat([ranplot,y_hat_proj],axis=0)

# Perform Back Transformation
y_hat_proj = inv_log_inv_sub(ranplot, n=4)

#Plot the new graph
pred_graph(model_data_t.iloc[:,2:3], y_hat_proj.iloc[22:,0:1], 14)
```

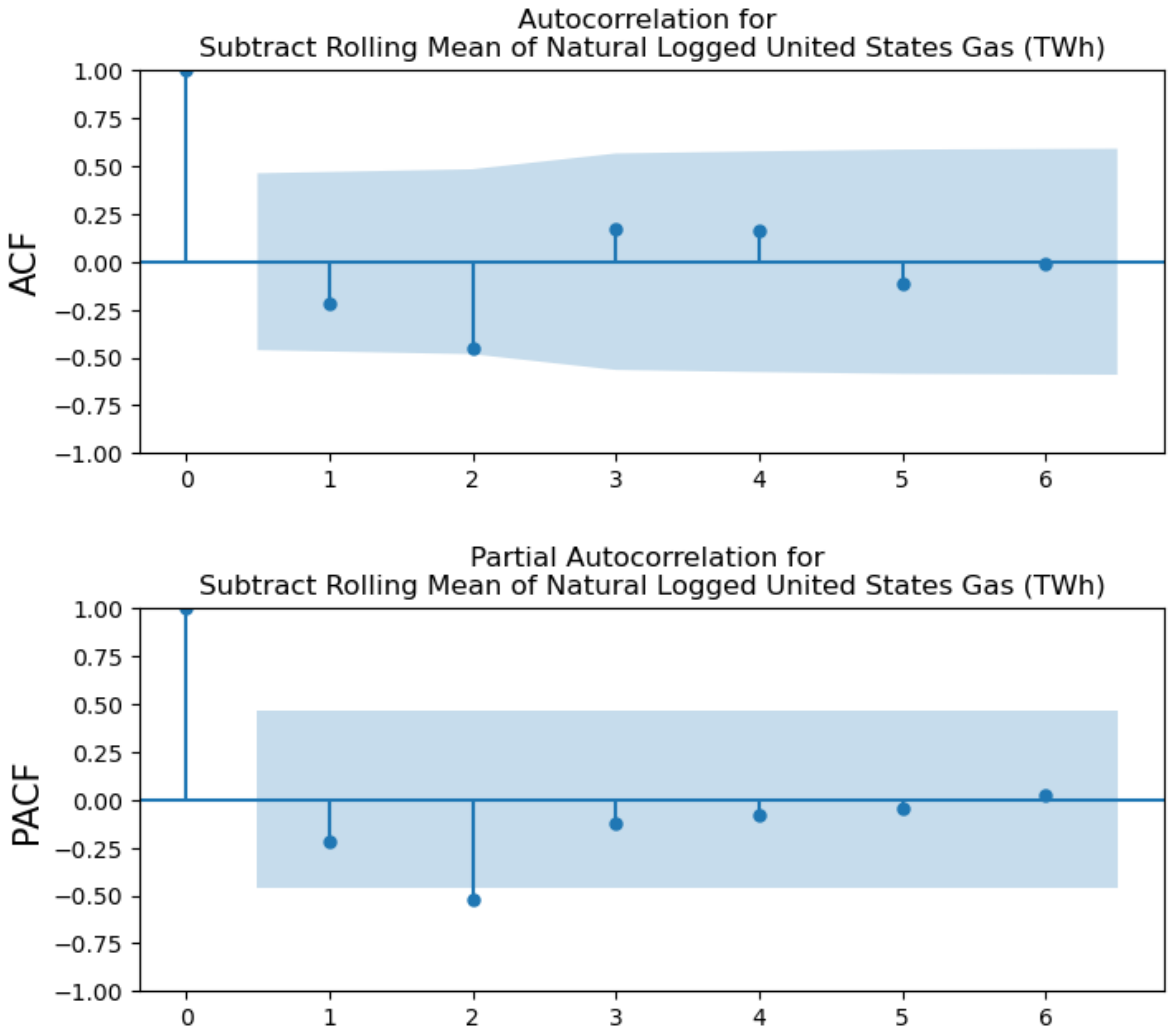


This model is as unreliable as it's name would imply, random walk. This one performs better in part because of a much smaller standard deviation for the dataset, and the transformation doesn't affect it nearly as strong. This one is ok.

Now I'll move forward with an ARMA model. First, I need to evaluate the ACF and PACF plots to find the best Autoregressive and Moving Average terms.

7.2.5 Evaluating Initial AR and MA Values with ACF and PACF Plots

In [69]:
```
plotacf(model_data_t.iloc[:,3:4], lags = 6)
plotpacf(model_data_t.iloc[:,3:4], lags = 6)
```

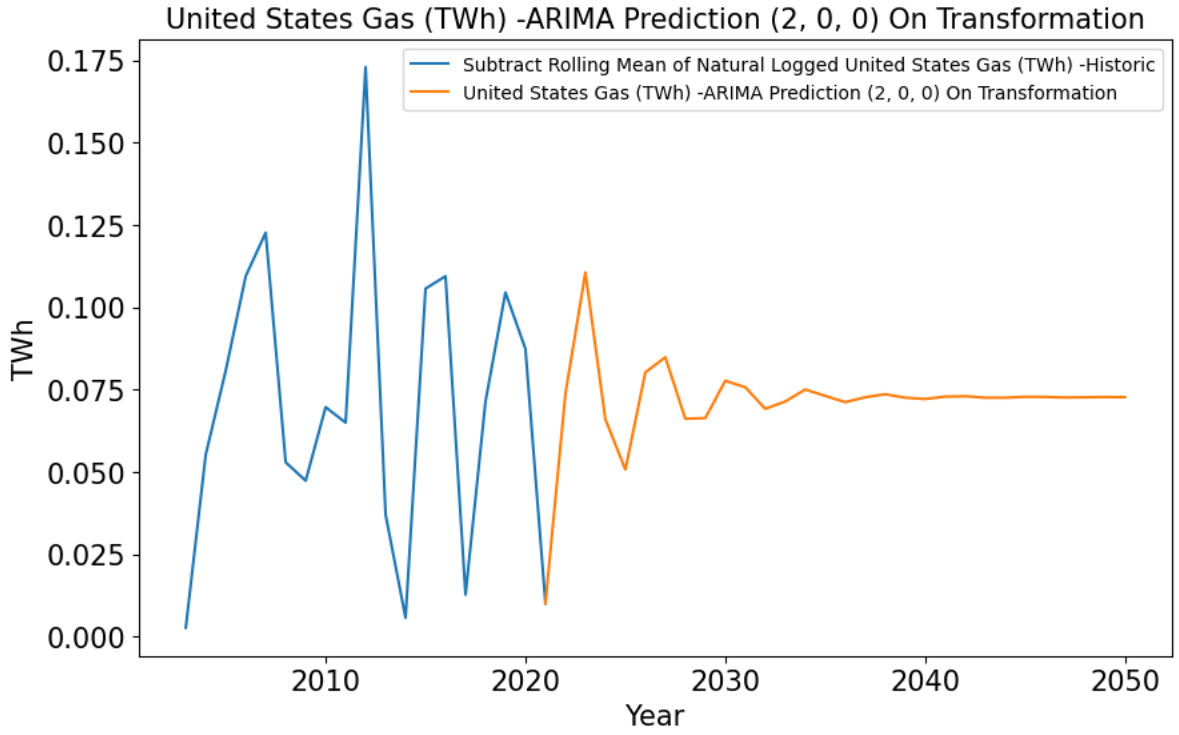


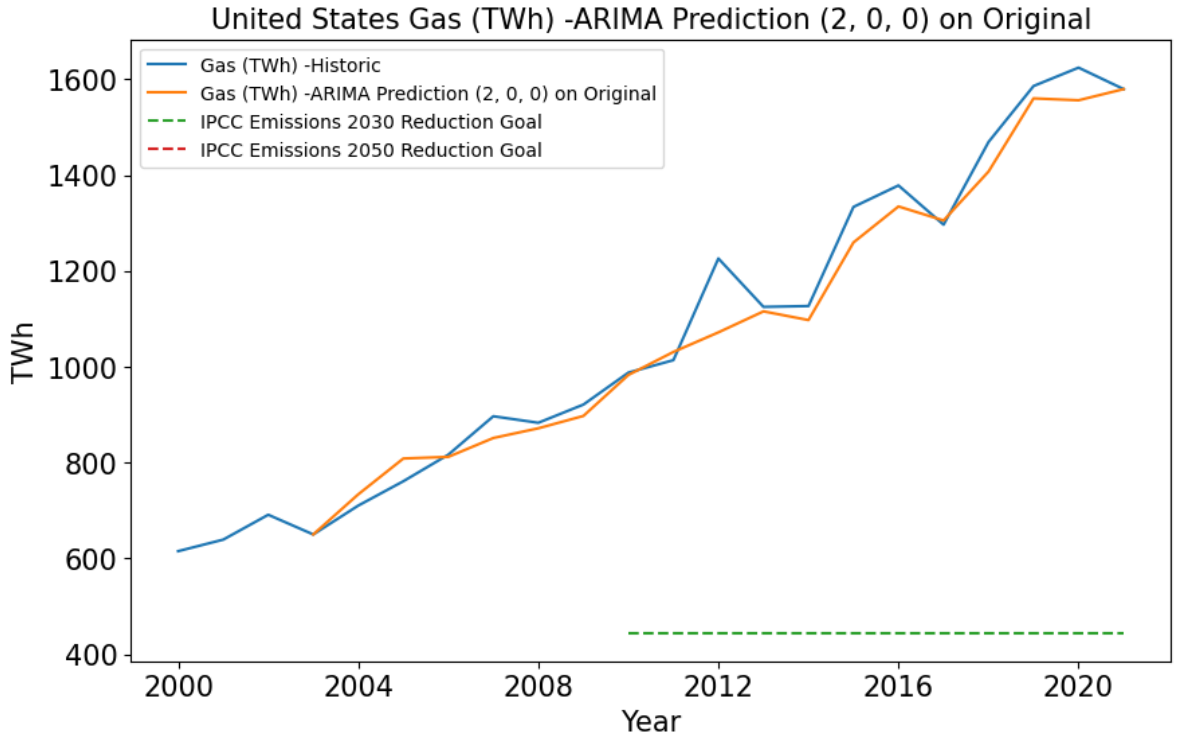
The ACF is awfully close to breaching at 2, and PACF definitely breaches at two. I will try both (2,0,0) and (2,0,2) for my initial ARMA models.

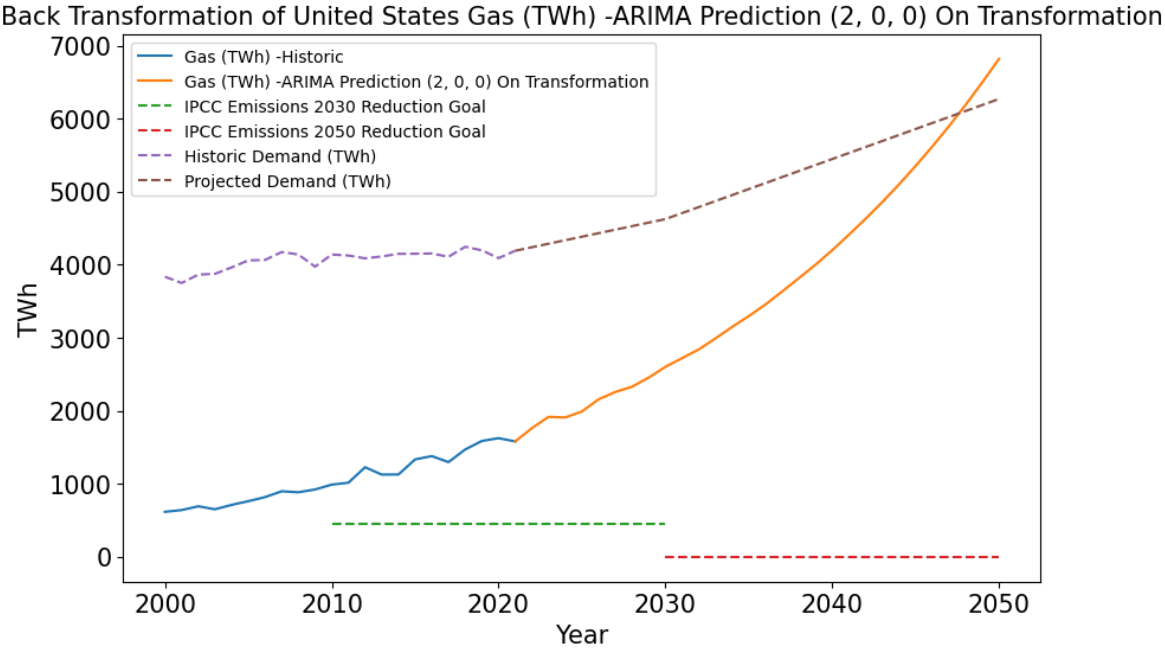
7.2.6 ARMA Model

In [70]:
```
pd.set_option('display.max_rows',60)
order = (2,0,0)
model_summary(model_data_t.iloc[:,2:4], order, 22, 50, 14,
          tran = 'inv_log_inv_sub', n=4)
```

```
ARIMA: (2, 0, 0), RMSE=51.81, AIC=-64.07
**********************************************************************
**
United States Gas (TWh) model results.

                                                          SARIMAX Results
==========================================================================
==============================================
Dep. Variable:     Subtract Rolling Mean of Natural Logged United States Gas
(TWh)   No. Observations:                   22
Model:                                                          ARIMA(2,
0, 0)   Log Likelihood                36.034
Date:                                                       Sat, 29 Apr
2023   AIC                            -64.067
Time:                                                             06:
16:21   BIC                            -59.703
Sample:                                                       01-01
-2000   HQIC                           -63.039
                                                            - 01-01
-2021
Covariance Type:
opg
==========================================================================
=
                 coef    std err          z      P>|z|      [0.025      0.97
5]
--------------------------------------------------------------------------
-
const          0.0727      0.007     10.359      0.000       0.059       0.08
6
ar.L1         -0.1594      0.356     -0.448      0.654      -0.856       0.53
8
ar.L2         -0.6077      0.331     -1.837      0.066      -1.256       0.04
1
sigma2         0.0013      0.000      2.720      0.007       0.000       0.00
2
==========================================================================
======
Ljung-Box (L1) (Q):                   0.22   Jarque-Bera (JB):
8.60
Prob(Q):                              0.64   Prob(JB):
0.01
Heteroskedasticity (H):               0.60   Skew:
1.19
Prob(H) (two-sided):                  0.52   Kurtosis:
4.92
==========================================================================
======

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (comple
x-step).
```


United States Gas (TWh) -ARIMA Prediction (2, 0, 0) On Transformation
Subtract Rolling Mean of Natural Logged United States Gas (TWh) -Historic
United States Gas (TWh) -ARIMA Prediction (2, 0, 0) On Transformation
TWh
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
2010
2020
2030
2040
2050
Year

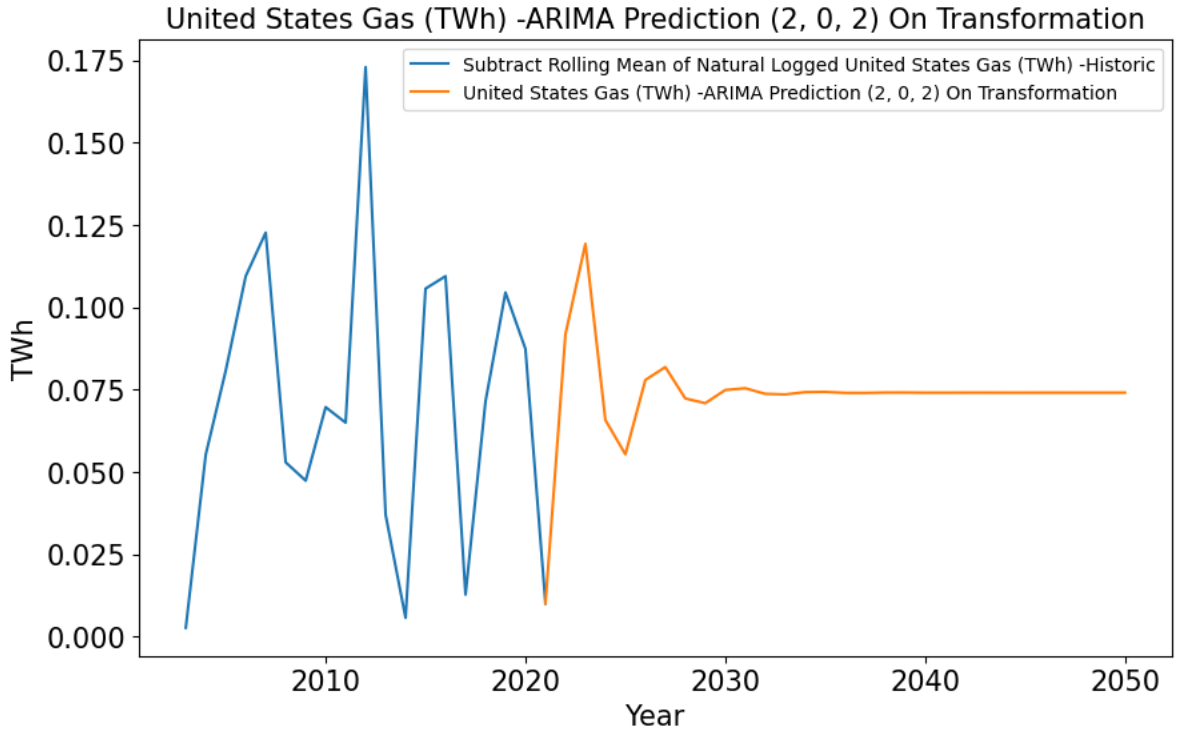

United States Gas (TWh) -ARIMA Prediction (2, 0, 0) on Original
Gas (TWh) -Historic
Gas (TWh) -ARIMA Prediction (2, 0, 0) on Original
IPCC Emissions 2030 Reduction Goal
IPCC Emissions 2050 Reduction Goal
TWh
1600
1400
1200
1000
800
600
400
2000
2004
2008
2012
2016
2020
Year

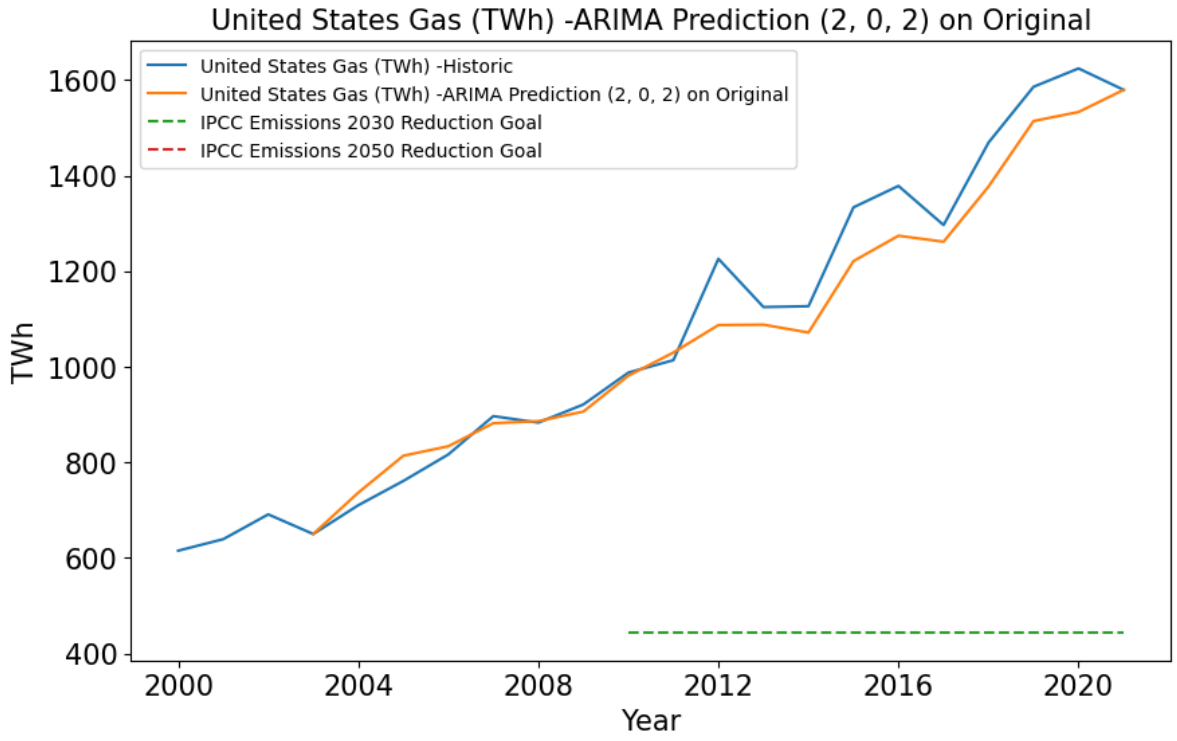

Back Transformation of United States Gas (TWh) -ARIMA Prediction (2, 0, 0) On Transformation
Gas (TWh) -Historic
Gas (TWh) -ARIMA Prediction (2, 0, 0) On Transformation
IPCC Emissions 2030 Reduction Goal
IPCC Emissions 2050 Reduction Goal
Historic Demand (TWh)
Projected Demand (TWh)
TWh
Year
0
1000
2000
3000
4000
5000
6000
7000
2000
2010
2020
2030
2040
2050

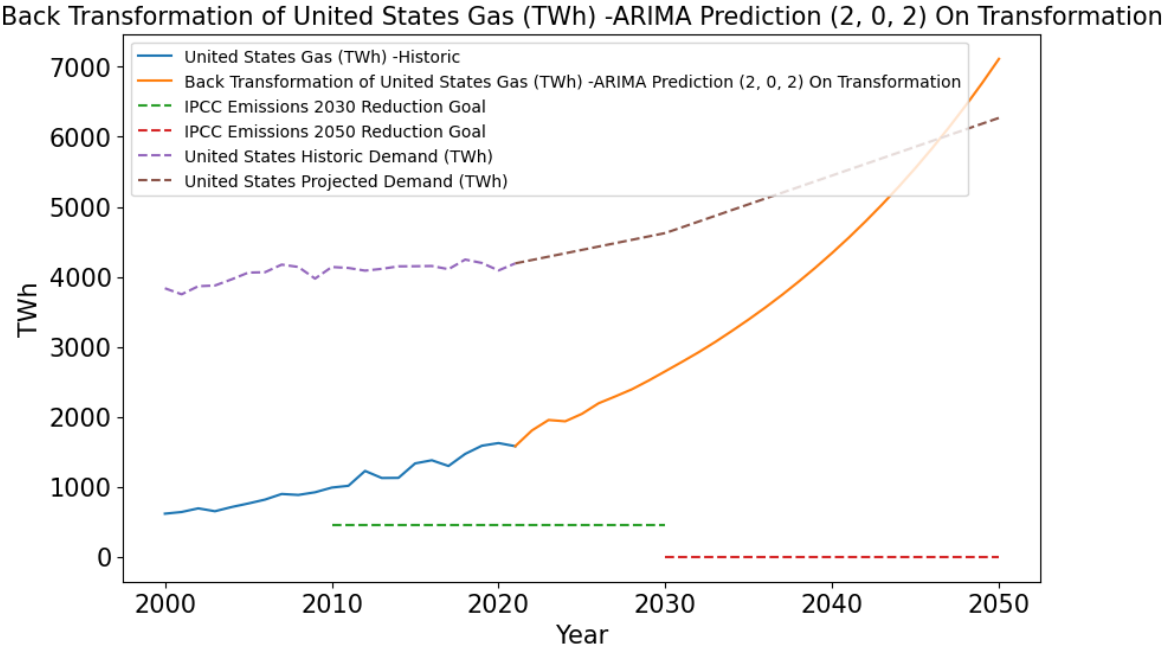
In [71]:
```
pd.set_option('display.max_rows',60)
order = (2,0,2)
model_summary(model_data_t.iloc[:,2:4], order, 22, 50, 0,
          tran = 'inv_log_inv_sub', n=4)
```

```
ARIMA: (2, 0, 2), RMSE=64.56, AIC=-63.61
**********************************************************************
**
United States Gas (TWh) model results.

                                                          SARIMAX Results
==========================================================================
==============================================
Dep. Variable:     Subtract Rolling Mean of Natural Logged United States Gas
(TWh)   No. Observations:                   22
Model:                                                          ARIMA(2,
0, 2)   Log Likelihood                37.804
Date:                                                       Sat, 29 Apr
2023   AIC                            -63.608
Time:                                                             06:
16:22   BIC                            -57.062
Sample:                                                       01-01
-2000   HQIC                           -62.066
                                                            - 01-01
-2021
Covariance Type:
opg
==========================================================================
=
                 coef    std err          z      P>|z|      [0.025      0.97
5]
--------------------------------------------------------------------------
-
const          0.0741      0.002     32.911      0.000       0.070       0.07
9
ar.L1         -0.0200      0.865     -0.023      0.982      -1.715       1.67
5
ar.L2         -0.4183      0.615     -0.680      0.497      -1.625       0.78
8
ma.L1         -0.5186      1.381     -0.375      0.707      -3.226       2.18
8
ma.L2         -0.3823      0.924     -0.414      0.679      -2.194       1.42
9
sigma2         0.0010      0.001      1.433      0.152      -0.000       0.00
2
==========================================================================
======
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):
5.85
Prob(Q):                              0.98   Prob(JB):
0.05
Heteroskedasticity (H):               1.00   Skew:
0.79
Prob(H) (two-sided):                  1.00   Kurtosis:
4.97
==========================================================================
======

Warnings:
```

```
[1] Covariance matrix calculated using the outer product of gradients (comple
x-step).
```





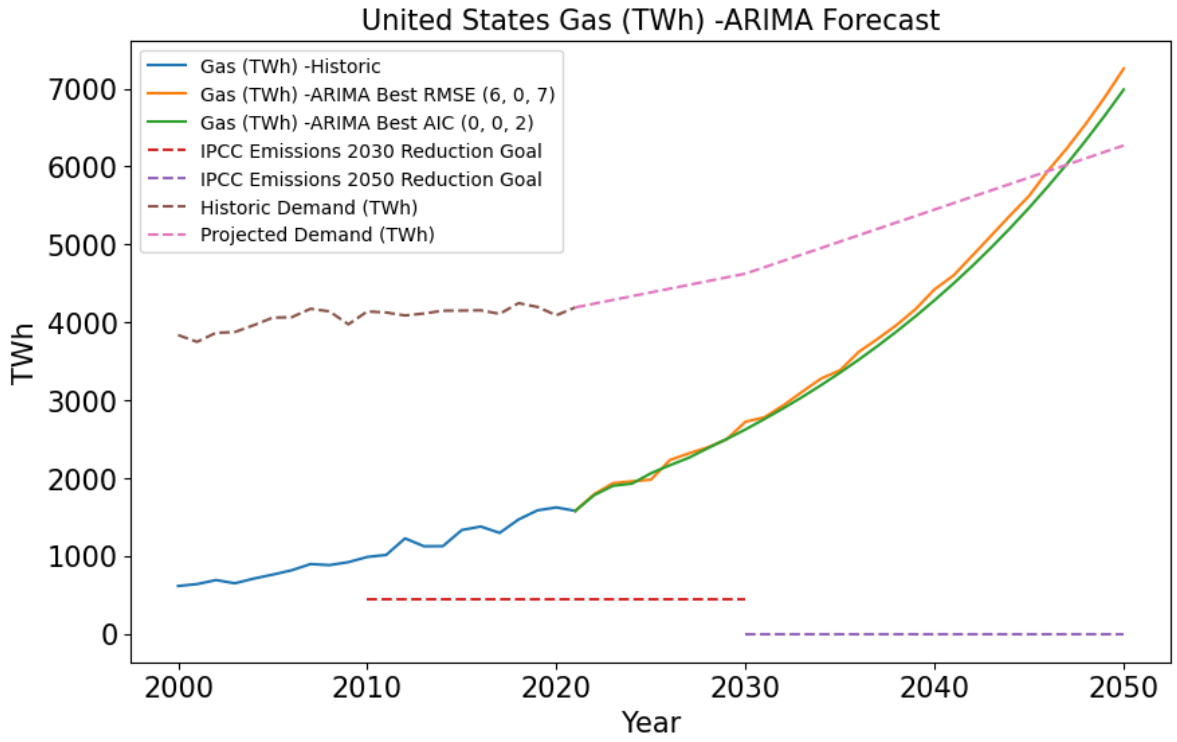


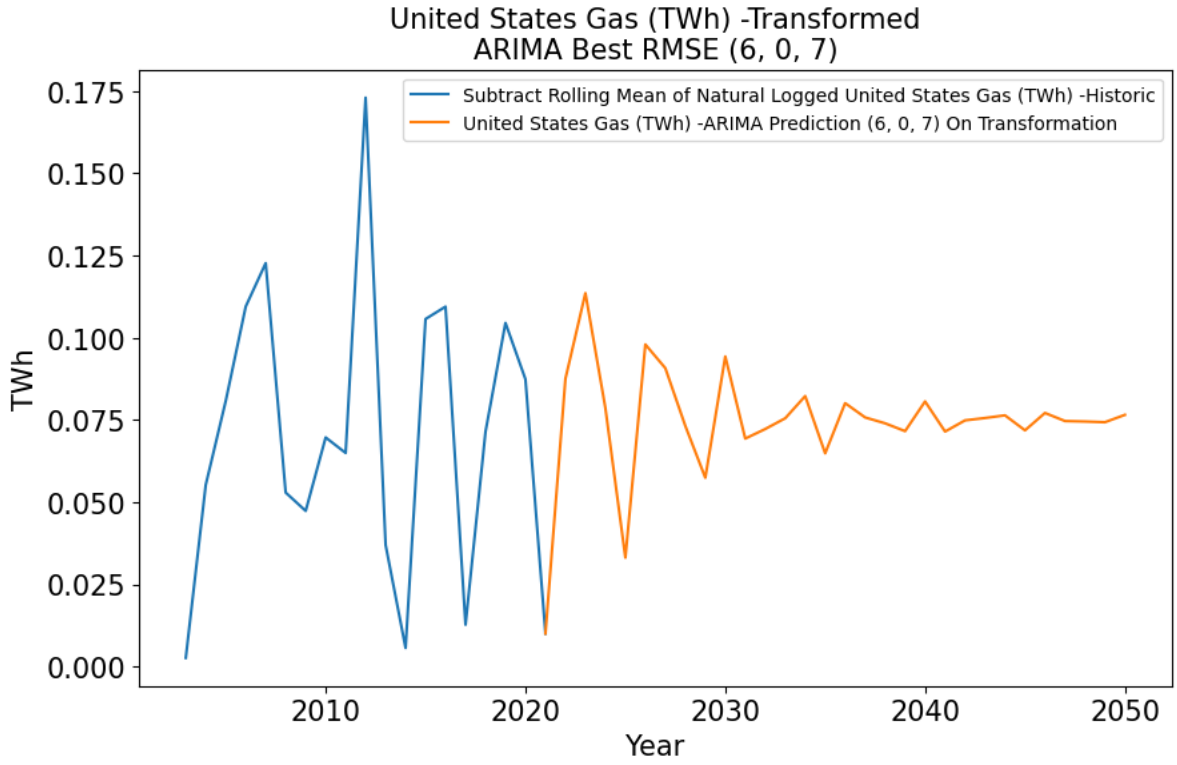
According to both models, Gas will overtake our entire electricity demand before 2050. I find this plausible.

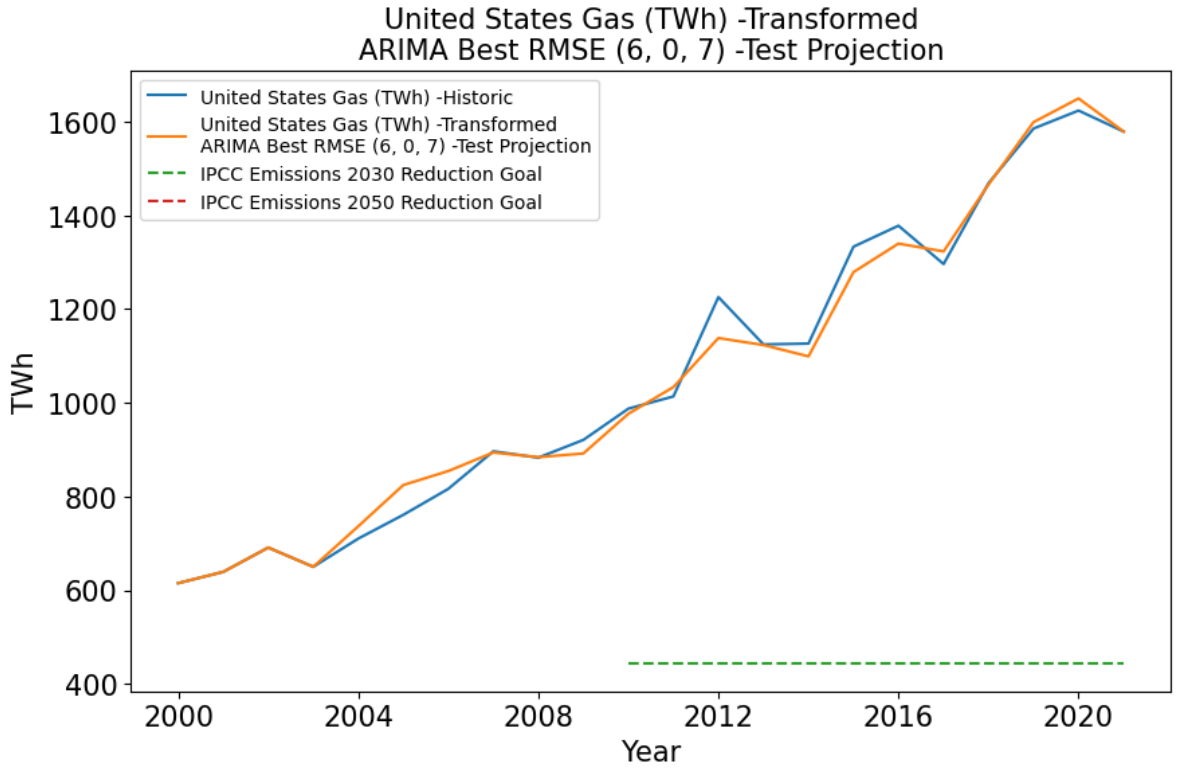
7.2.7 ARIMA Model and Grid Search

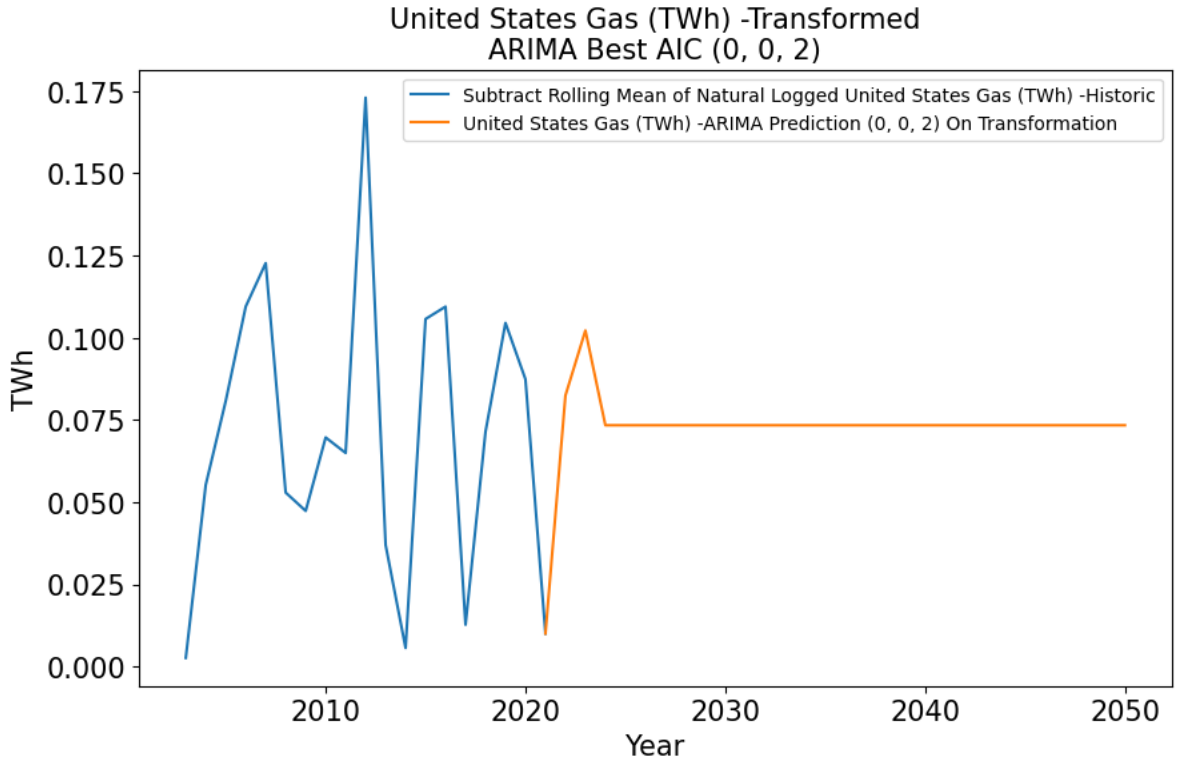
In [72]:
```
gas_rmse_cfg, gas_aic_cfg = arima_pdq(
                model_data_t.iloc[:,2:4], s=14, tran = 'inv_log_inv_sub', n=4)
```

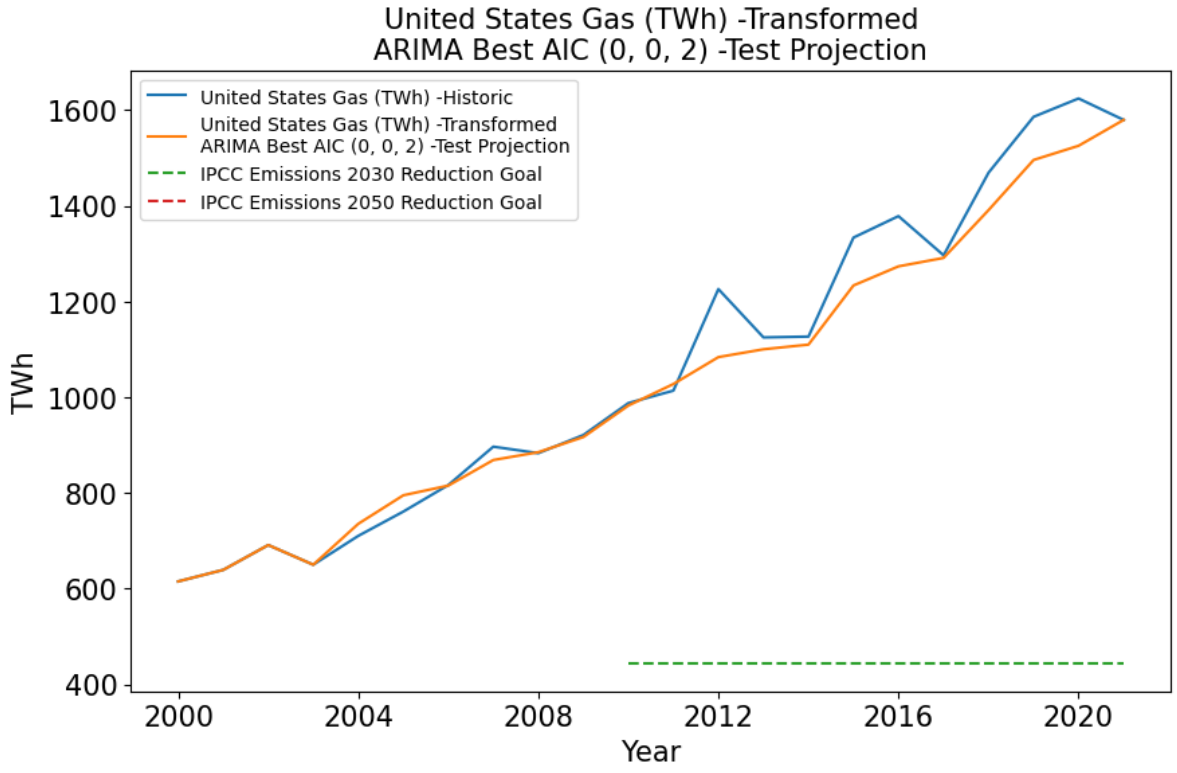
```
RMSE ARIMA: (0, 0, 0), RMSE= 86.19, AIC= -60.63, TWh-2050= 6322.94
RMSE ARIMA: (0, 0, 1), RMSE= 65.99, AIC= -64.47, TWh-2050= 7146.80
RMSE ARIMA: (0, 0, 2), RMSE= 62.03, AIC= -65.37, TWh-2050= 6987.87
RMSE ARIMA: (0, 0, 4), RMSE= 46.19, AIC= -63.56, TWh-2050= 6758.34
RMSE ARIMA: (0, 0, 6), RMSE= 45.82, AIC= -64.17, TWh-2050= 7161.13
RMSE ARIMA: (0, 0, 7), RMSE= 44.99, AIC= -61.79, TWh-2050= 7058.95
RMSE ARIMA: (1, 0, 7), RMSE= 43.46, AIC= -60.56, TWh-2050= 7096.00
RMSE ARIMA: (3, 0, 4), RMSE= 38.24, AIC= -61.32, TWh-2050= 7001.50
RMSE ARIMA: (6, 0, 7), RMSE= 37.63, AIC= -51.93, TWh-2050= 7256.92
Best RMSE ARIMA: (6, 0, 7) RMSE= 37.63 AIC= -51.93, TWh-2050= 7256.92
Best AIC  ARIMA: (0, 0, 2) RMSE= 62.03 AIC= -65.37, TWh-2050= 6987.87
Graph_formatting produced error at (6, 0, 7) or (0, 0, 2)
```




United States Gas (TWh) -Transformed
ARIMA Best RMSE (6, 0, 7)
Subtract Rolling Mean of Natural Logged United States Gas (TWh) -Historic
United States Gas (TWh) -ARIMA Prediction (6, 0, 7) On Transformation
TWh
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
2010
2020
2030
2040
2050
Year


United States Gas (TWh) -Transformed
ARIMA Best RMSE (6, 0, 7) -Test Projection
United States Gas (TWh) -Historic
United States Gas (TWh) -Transformed
ARIMA Best RMSE (6, 0, 7) -Test Projection
IPCC Emissions 2030 Reduction Goal
IPCC Emissions 2050 Reduction Goal
TWh
1600
1400
1200
1000
800
600
400
2000
2004
2008
2012
2016
2020
Year



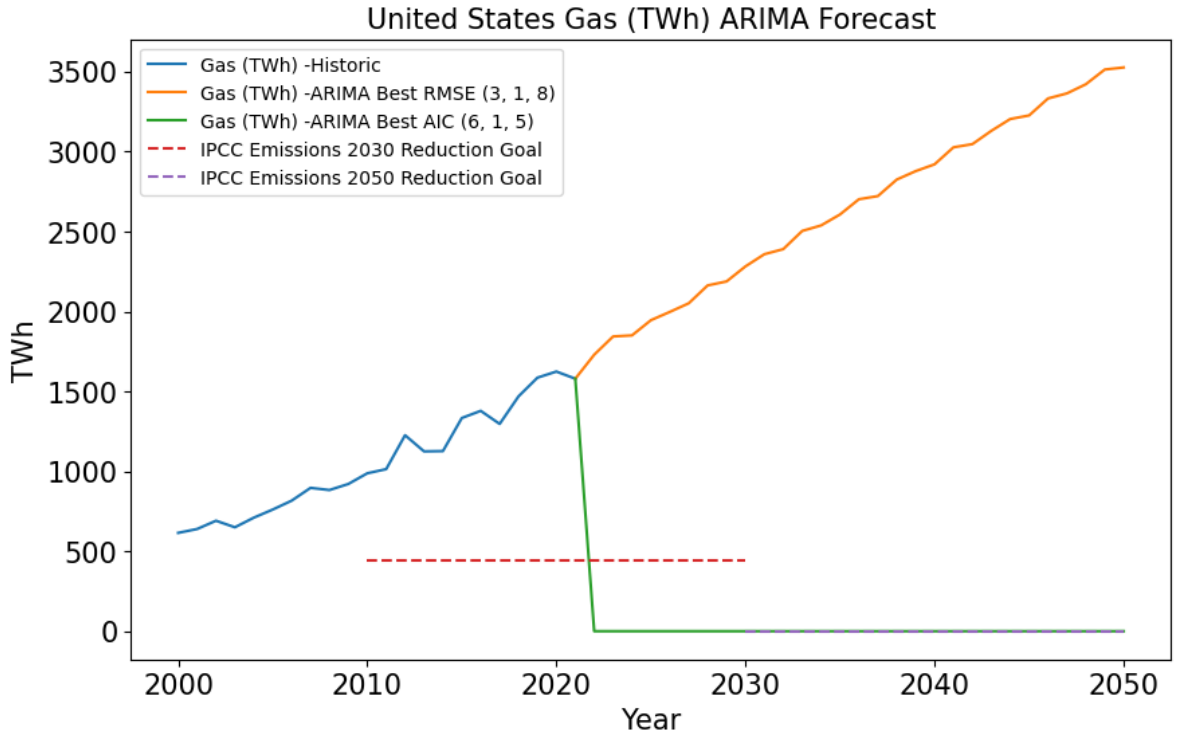


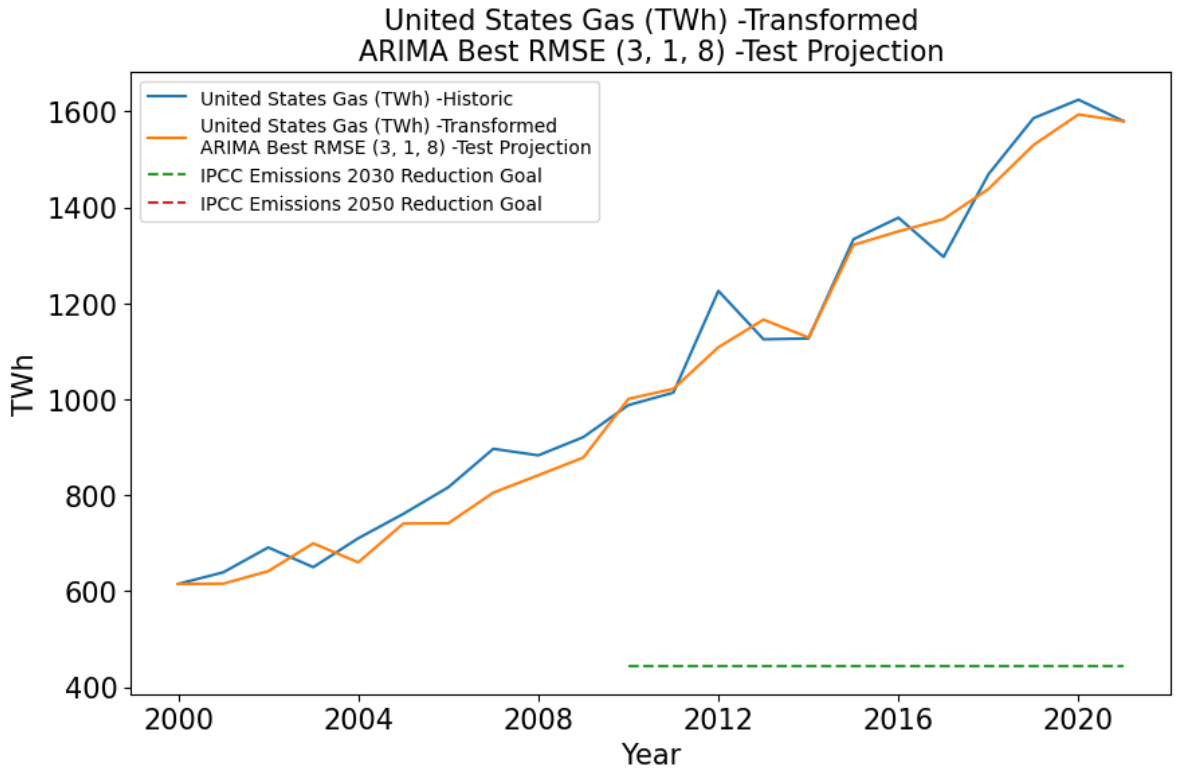
This predicts that gas will be on an exponential curve, which I think is more realistic than I'd like to admit. I think these models are the best I've got. I'll still try it without transformations though.

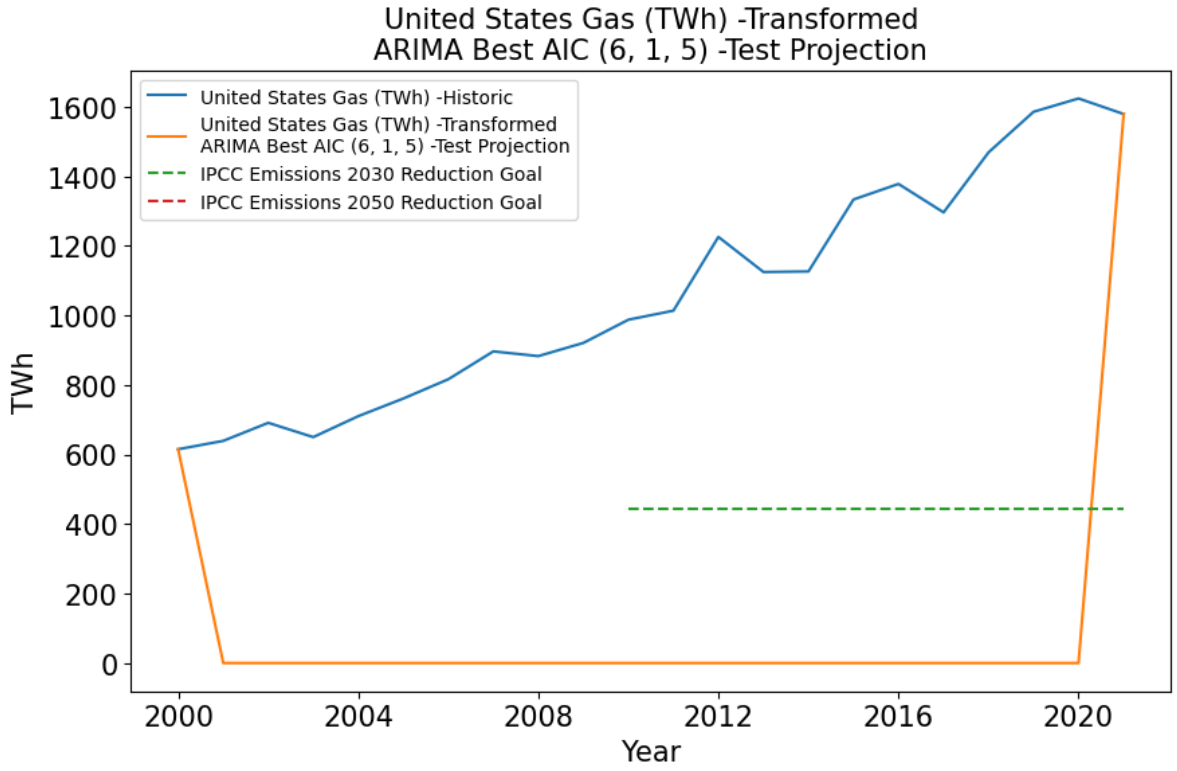
7.2.8 ARIMA Without Transformation

In [73]:
```
gas_rmse_cfg, gas_aic_cfg = arima_pdq_no_tran(
                                        model_data.iloc[:,1:2], 22, 50, s=14)
```

```
United States Gas (TWh) Grid Search:
RMSE ARIMA: (0, 0, 0), RMSE= 291.37, AIC= 321.03, TWh-2050= 1060.39
RMSE ARIMA: (0, 0, 1), RMSE= 175.67, AIC= 300.44, TWh-2050= 1060.43
RMSE ARIMA: (0, 0, 2), RMSE= 130.77, AIC= 290.23, TWh-2050= 1065.74
RMSE ARIMA: (0, 0, 3), RMSE= 122.32, AIC= 289.26, TWh-2050= 1060.20
RMSE ARIMA: (0, 0, 4), RMSE= 117.19, AIC= 289.79, TWh-2050= 1060.51
RMSE ARIMA: (0, 0, 5), RMSE= 89.31, AIC= 281.80, TWh-2050= 1061.51
RMSE ARIMA: (0, 0, 7), RMSE= 80.43, AIC= 280.75, TWh-2050= 1060.84
RMSE ARIMA: (0, 0, 8), RMSE= 75.16, AIC= 279.95, TWh-2050= 1060.72
AIC  ARIMA: (0, 1, 0), RMSE= 90.22, AIC= 251.90, TWh-2050= 1579.36
RMSE ARIMA: (0, 1, 4), RMSE= 71.91, AIC= 251.43, TWh-2050= 1730.23
RMSE ARIMA: (0, 1, 7), RMSE= 63.42, AIC= 252.73, TWh-2050= 1789.11
RMSE ARIMA: (0, 1, 8), RMSE= 62.33, AIC= 254.16, TWh-2050= 1800.18
AIC  ARIMA: (0, 2, 1), RMSE= 100.21, AIC= 240.38, TWh-2050= 2916.05
AIC  ARIMA: (0, 2, 2), RMSE= 85.83, AIC= 233.30, TWh-2050= 3466.02
AIC  ARIMA: (0, 2, 4), RMSE= 82.71, AIC= 233.05, TWh-2050= 3486.44
AIC  ARIMA: (0, 3, 3), RMSE= 146.15, AIC= 230.95, TWh-2050= 4319.87
AIC  ARIMA: (0, 3, 4), RMSE= 144.99, AIC= 230.78, TWh-2050= 4528.69
AIC  ARIMA: (0, 3, 8), RMSE= 144.53, AIC= 230.52, TWh-2050= 4592.29
RMSE ARIMA: (1, 1, 3), RMSE= 62.24, AIC= 247.51, TWh-2050= 3308.01
RMSE ARIMA: (1, 1, 7), RMSE= 52.47, AIC= 248.28, TWh-2050= 3676.23
AIC  ARIMA: (1, 3, 7), RMSE= 142.81, AIC= 229.92, TWh-2050= 4654.40
RMSE ARIMA: (2, 1, 7), RMSE= 50.49, AIC= 247.73, TWh-2050= 3495.19
AIC  ARIMA: (2, 3, 3), RMSE= 144.52, AIC= 227.43, TWh-2050= 4519.99
RMSE ARIMA: (3, 1, 8), RMSE= 49.47, AIC= 250.91, TWh-2050= 3526.45
AIC  ARIMA: (4, 0, 7), RMSE= 291.56, AIC= 26.00, TWh-2050= 1068.30
AIC  ARIMA: (6, 1, 5), RMSE= 1048.81, AIC= 24.00, TWh-2050= 0.00
Best RMSE ARIMA: (3, 1, 8) RMSE= 49.47 AIC= 250.91, TWh-2050= 3526.45
Best AIC  ARIMA: (6, 1, 5) RMSE= 1048.81 AIC= 24.00, TWh-2050= 0.00
```


United States Gas (TWh) ARIMA Forecast
Gas (TWh) -Historic
Gas (TWh) -ARIMA Best RMSE (3, 1, 8)
Gas (TWh) -ARIMA Best AIC (6, 1, 5)
IPCC Emissions 2030 Reduction Goal
IPCC Emissions 2050 Reduction Goal
TWh
Year
0
500
1000
1500
2000
2500
3000
3500
2000
2010
2020
2030
2040
2050


United States Gas (TWh) -Transformed
ARIMA Best RMSE (3, 1, 8) -Test Projection
United States Gas (TWh) -Historic
United States Gas (TWh) -Transformed
ARIMA Best RMSE (3, 1, 8) -Test Projection
IPCC Emissions 2030 Reduction Goal
IPCC Emissions 2050 Reduction Goal
TWh
Year
400
600
800
1000
1200
1400
1600
2000
2004
2008
2012
2016
2020



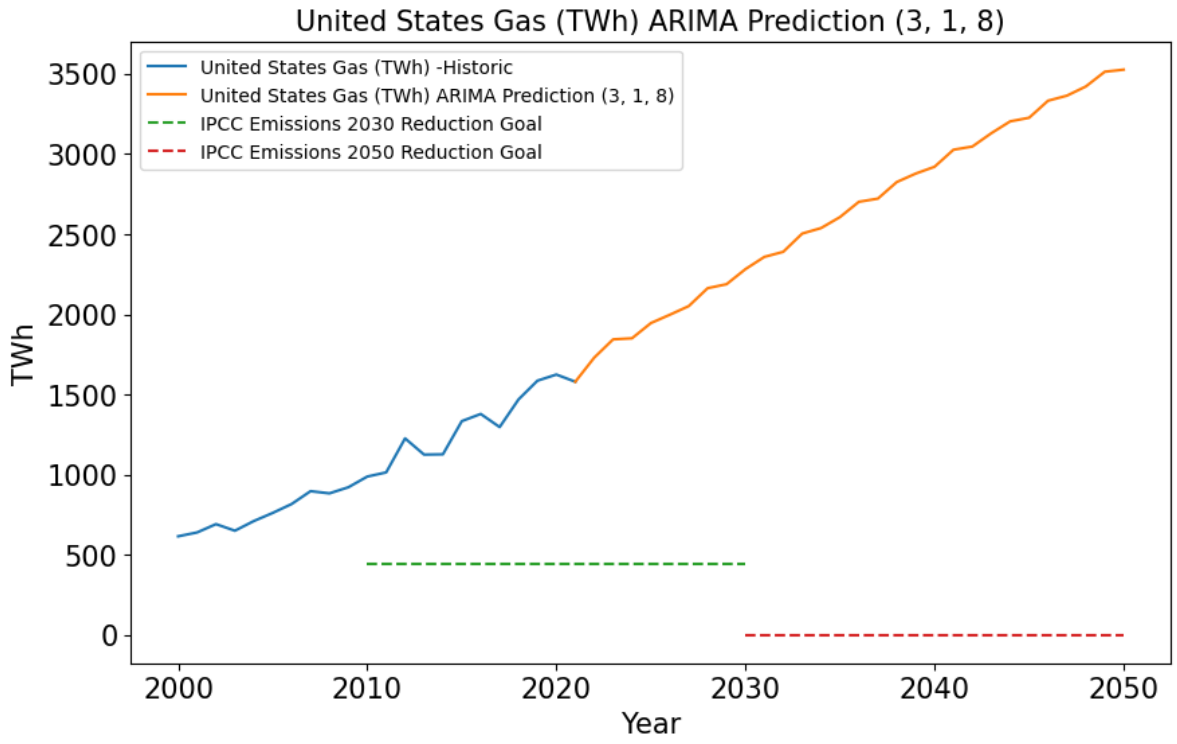
7.2.9 Model Selection

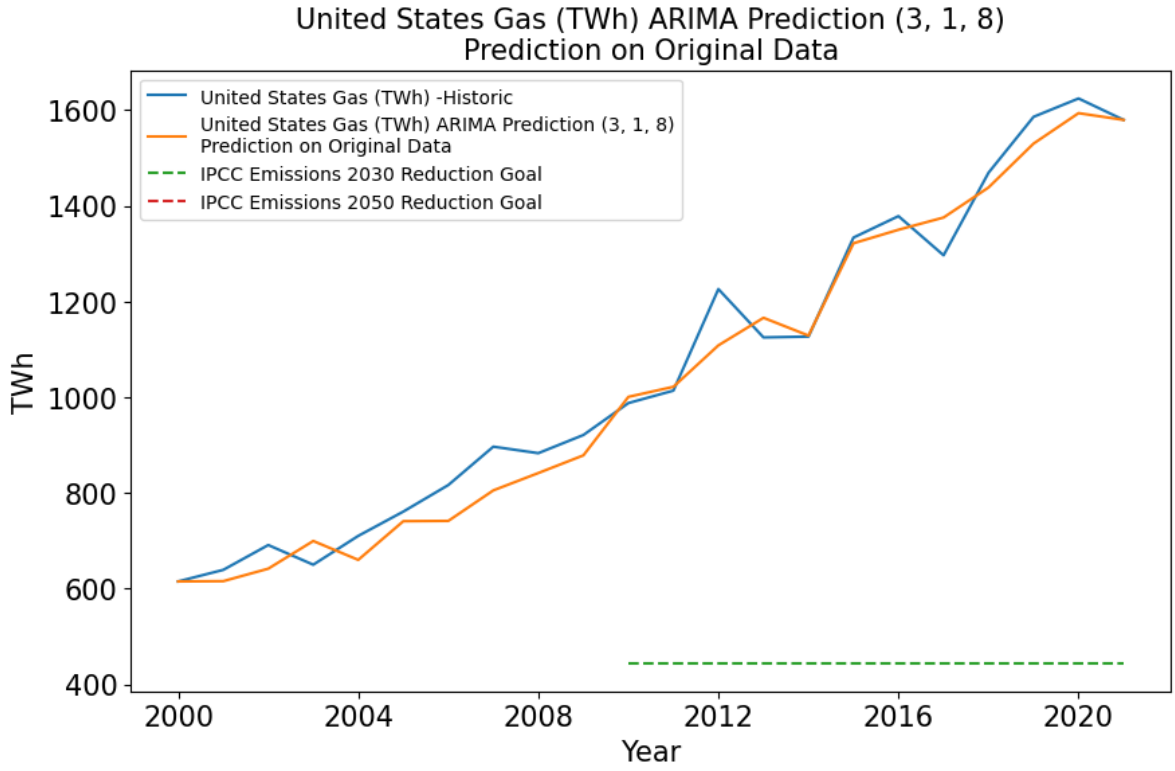
In [74]:
```
#Deleteme
stored_gas = selected_models.copy()
selected_models = stored_gas.copy()
```

In [75]:
```
selected_models = stored_gas.copy()
df_gas = model_data_t.iloc[:,2:3]

#                        Model,  df_o,   pdq,     tran, n, PDQs
selected_models.append(['ARIMA', df_gas, (3,1,8), None, 0, None])

m = 1
model_summary_no_tran(selected_models[m][1], selected_models[m][2], 22, 50);
```

```
ARIMA: (3, 1, 8), RMSE=49.48, AIC=250.91
**********************************************************************
**
United States Gas (TWh) -ARIMA Prediction (3, 1, 8) model results.

                                SARIMAX Results
==========================================================================
======
Dep. Variable:     United States Gas (TWh)   No. Observations:
22
Model:                       ARIMA(3, 1, 8)   Log Likelihood               -1
13.455
Date:                     Sat, 29 Apr 2023   AIC                          2
50.909
Time:                             06:17:40   BIC                          2
63.444
Sample:                         01-01-2000   HQIC                         2
53.630
                              - 01-01-2021
Covariance Type:                       opg
==========================================================================
=
                 coef    std err          z      P>|z|      [0.025      0.97
5]
--------------------------------------------------------------------------
-
ar.L1         -0.4936      1.697     -0.291      0.771      -3.820       2.83
3
ar.L2          0.4748      0.726      0.654      0.513      -0.947       1.89
7
ar.L3          0.9910      1.310      0.757      0.449      -1.576       3.55
8
ma.L1         -0.2718     32.789     -0.008      0.993     -64.536      63.99
3
ma.L2         -1.3443     12.196     -0.110      0.912     -25.247      22.55
9
ma.L3         -0.0217     64.316     -0.000      1.000    -126.078     126.03
5
ma.L4          1.9865     46.812      0.042      0.966     -89.763      93.73
6
ma.L5         -0.2026     49.044     -0.004      0.997     -96.327      95.92
2
ma.L6         -1.2371     42.049     -0.029      0.977     -83.652      81.17
7
ma.L7         -0.0502     24.595     -0.002      0.998     -48.255      48.15
5
ma.L8          0.7899     26.644      0.030      0.976     -51.431      53.01
1
sigma2      1454.2088   5.02e+04      0.029      0.977    -9.7e+04    9.99e+0
4
==========================================================================
======
Ljung-Box (L1) (Q):                   0.74   Jarque-Bera (JB):
0.84
Prob(Q):                              0.39   Prob(JB):
0.66
```

```
Heteroskedasticity (H):                 1.28   Skew:
-0.17
Prob(H) (two-sided):                    0.76   Kurtosis:
3.92
================================================================================
======

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (comple
x-step).
[2] Covariance matrix is singular or near-singular, with condition number 9.9
6e+14. Standard errors may be unstable.
```





The models produced by the transformation model all show a shockingly high rate of natural gas into the future. Nearly all of the predict natural gas could be used as our sole power source by 2050. There are many reasons that is unlikely.

The non-transformation model still shows a steady incline in natural gas, and predicts about half our energy use by 2050 will be natural gas. This is much more reasonable. The RMSE difference between the two (49.56 and 37.63) is less than 15. I'll select the non-transformation model for this source.

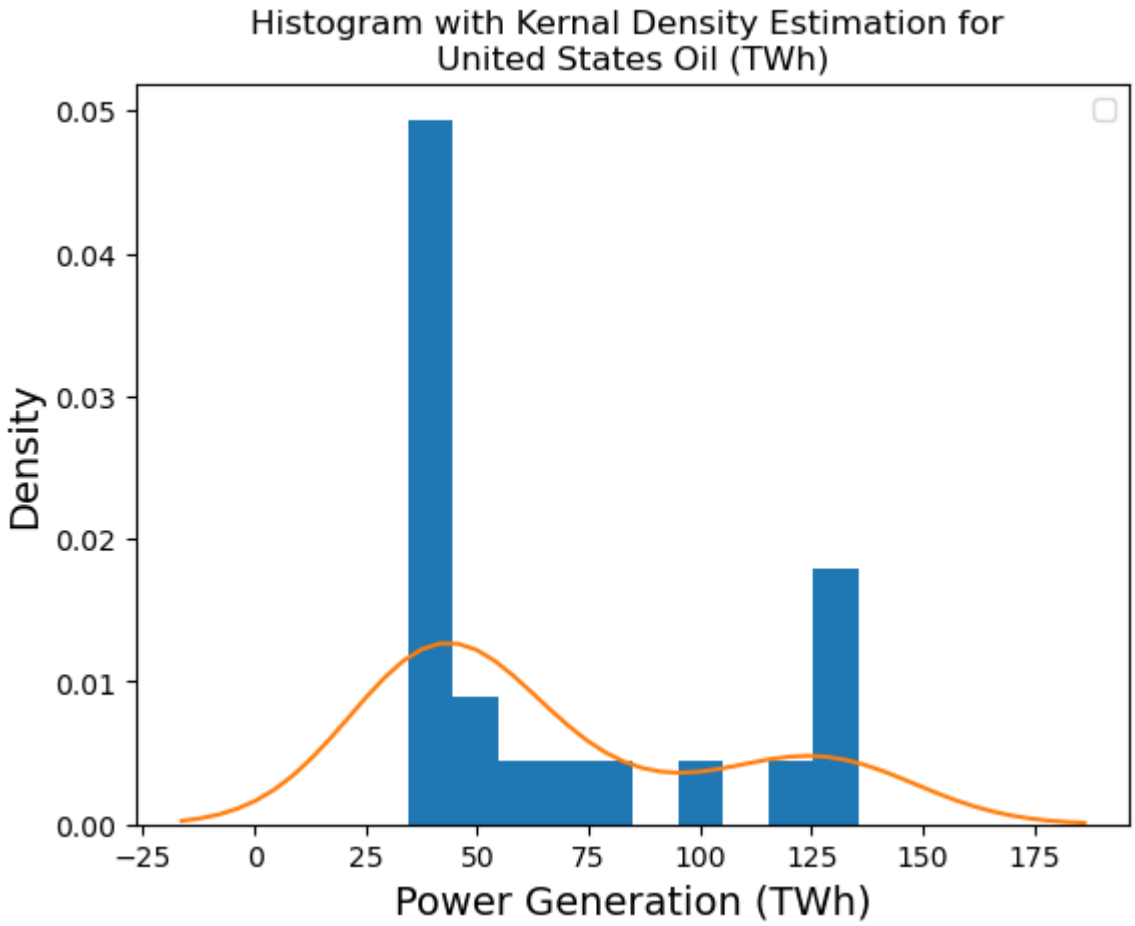
7.3 Oil

7.3.1 Distribution Investigation

In [76]:
```
stored_model_data = model_data.copy()
stored_model_data_t = model_data_t.copy()
```

In [77]:
```
model_data = stored_model_data.copy()
model_data_t = stored_model_data_t.copy()
```

In [78]:
```
hist(model_data.iloc[:,2:3])
```



In [79]:
```
stats_block = s_block(model_data.iloc[:,2:3], stats_block)
stats_block
```

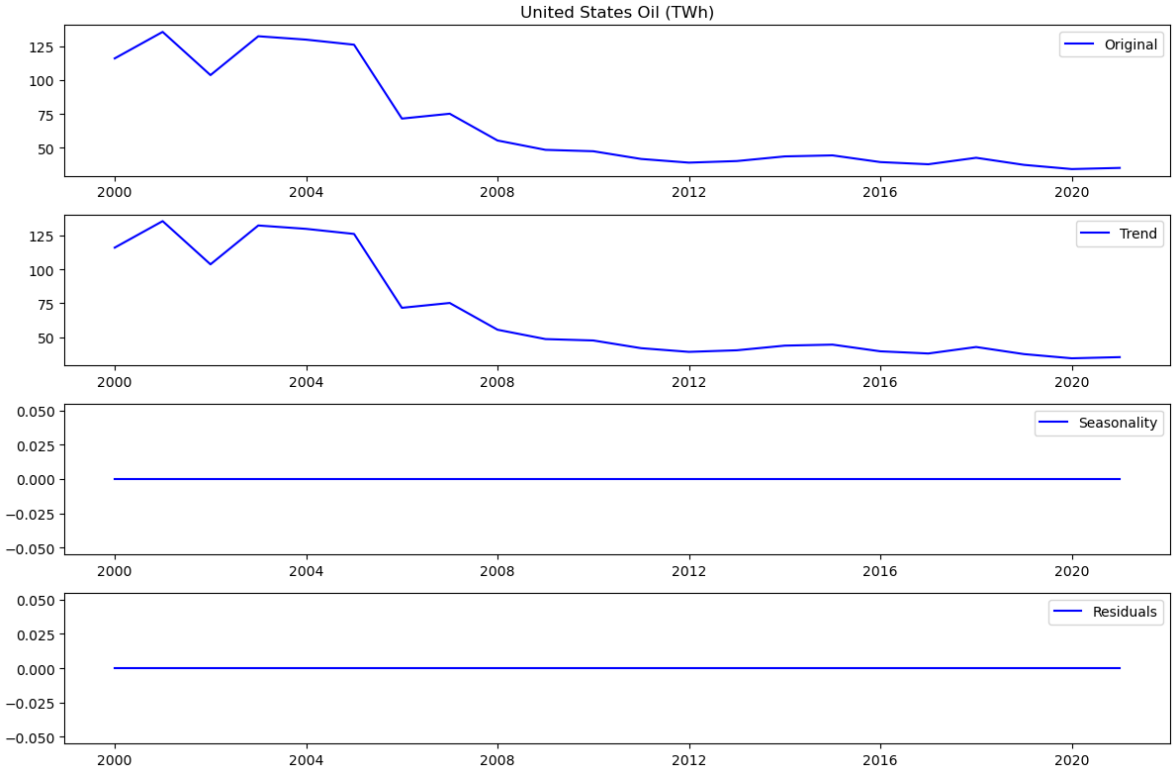
Out[79]:

	Dataset	Mean	Median	Standard Deviation	Skew	Fisher Kurtosis
0	United States Coal (TWh)	1,607.17	1,744.66	399.15	-0.68	-0.90
1	United States Gas (TWh)	1,060.39	1,000.70	325.82	0.29	-1.21
2	United States Oil (TWh)	67.20	45.97	36.59	0.89	-0.92

This has a positive skew distribution with wide and flat tails.

7.3.2 Decomposing The Data

In [80]:
```
decomp_graph(model_data.iloc[:,2:3])
```

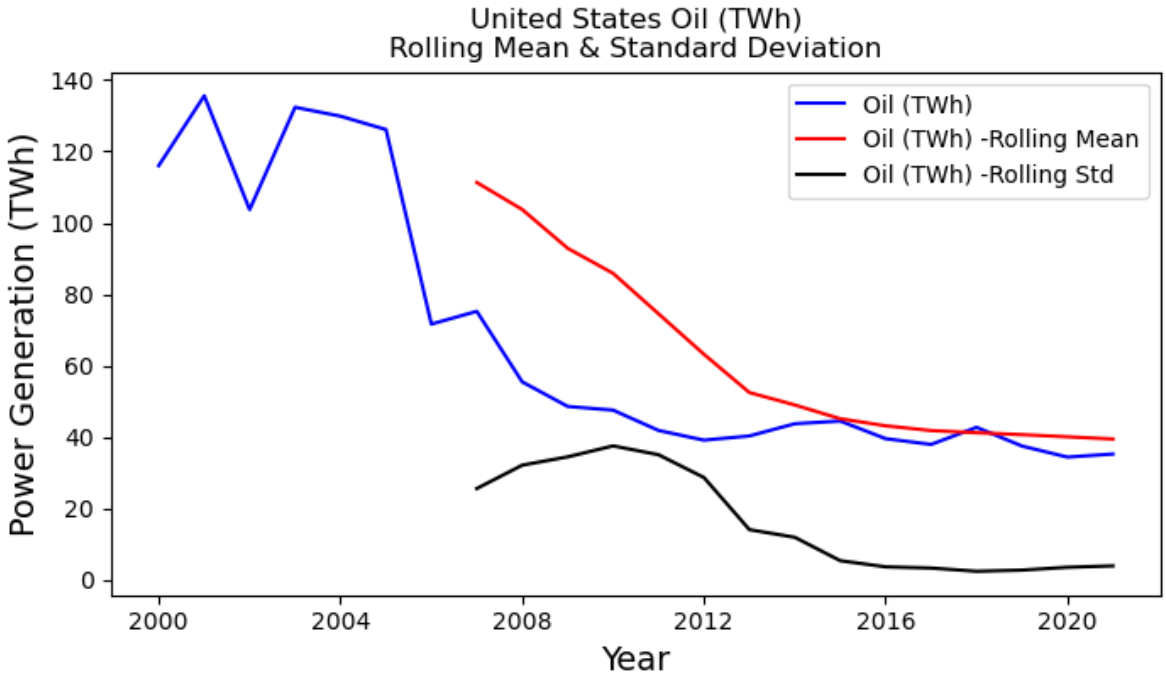


No seasonality or residuals. I'll check for stationarity.

7.3.3 Checking for Stationarity and Flattening

In [81]:
```
pd.set_option('display.max_rows', 30)
stationarity_check(model_data.iloc[:,2:3], s=14)
```

```
United States Oil (TWh)
Results of Dickey-Fuller test:

Test Statistic                -0.47
p-value                        0.90
#Lags Used                     9.00
Number of Observations Used   12.00
Critical Value (1%)           -4.14
Critical Value (5%)           -3.15
Critical Value (10%)          -2.71
dtype: float64
```



This is not close to stationary. After playing around with transformations, I found that if I took the difference of values, I can get it stationary.

In [82]:
```
t = difference(model_data.iloc[:,2:3])
model_data_t.insert(5,t.columns[0],t)

model_data_t.iloc[:,4:6].head()
```

Out[82]:

Year	United States Oil (TWh)	Annual Change of United States Oil (TWh)
2000-01-01	116.02	NaN
2001-01-01	135.59	19.57
2002-01-01	103.69	-31.90
2003-01-01	132.38	28.69
2004-01-01	129.88	-2.50

Before I move forward, I want to make sure I can transform this data back after modeling.

In [83]:
```
test = model_data_t.iloc[:,4:6].copy()

test['Results'] = inv_difference(test)
test
```

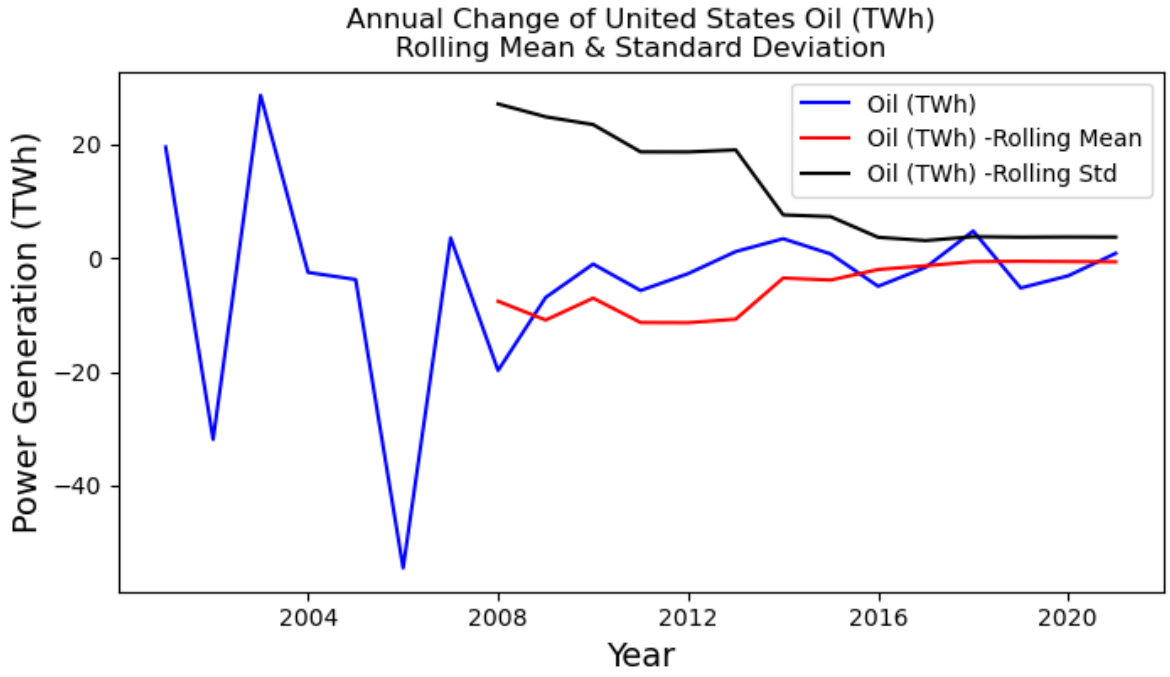
Out[83]:

Year	United States Oil (TWh)	Annual Change of United States Oil (TWh)	Results
2000-01-01	116.02	NaN	116.02
2001-01-01	135.59	19.57	135.59
2002-01-01	103.69	-31.90	103.69
2003-01-01	132.38	28.69	132.38
2004-01-01	129.88	-2.50	129.88
2005-01-01	126.13	-3.75	126.13
2006-01-01	71.61	-54.52	71.61
2007-01-01	75.19	3.58	75.19
2008-01-01	55.43	-19.76	55.43
2009-01-01	48.53	-6.90	48.53
2010-01-01	47.50	-1.03	47.50
2011-01-01	41.81	-5.69	41.81
2012-01-01	39.09	-2.72	39.09
2013-01-01	40.26	1.17	40.26
2014-01-01	43.69	3.43	43.69
2015-01-01	44.44	0.75	44.44
2016-01-01	39.50	-4.94	39.50
2017-01-01	37.89	-1.61	37.89
2018-01-01	42.68	4.79	42.68
2019-01-01	37.44	-5.24	37.44
2020-01-01	34.34	-3.10	34.34
2021-01-01	35.20	0.86	35.20

Transferring back is a possibility. Therefore, I can move forward. I'll again check for stationarity.

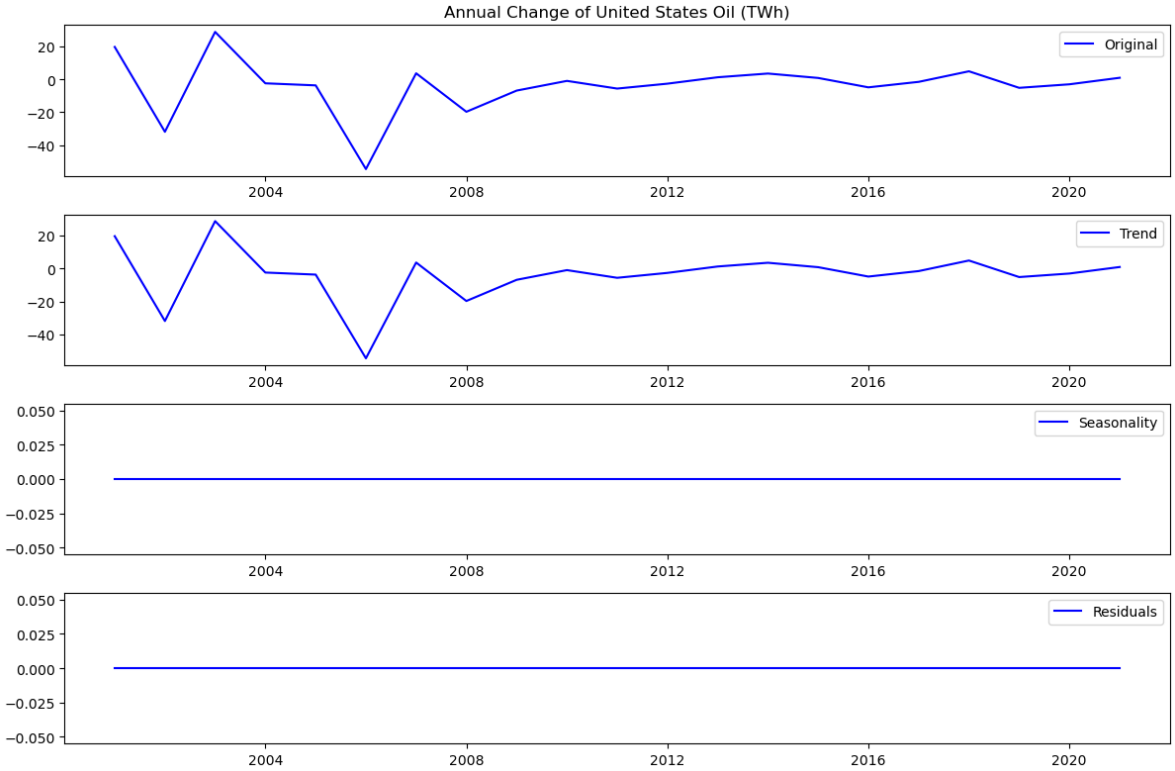
In [84]:
```
stationarity_check(model_data_t.iloc[:,5:6], s=31)
```

```
Annual Change of United States Oil (TWh)
Results of Dickey-Fuller test:

Test Statistic                 -2.84
p-value                         0.05
#Lags Used                      8.00
Number of Observations Used    12.00
Critical Value (1%)            -4.14
Critical Value (5%)            -3.15
Critical Value (10%)           -2.71
dtype: float64
```



My p-value is .05. Stationarity achieved. I'll look at the decomposition graph again.

In [85]:
```
decomp_graph(model_data_t.iloc[:,5:6].dropna())
```

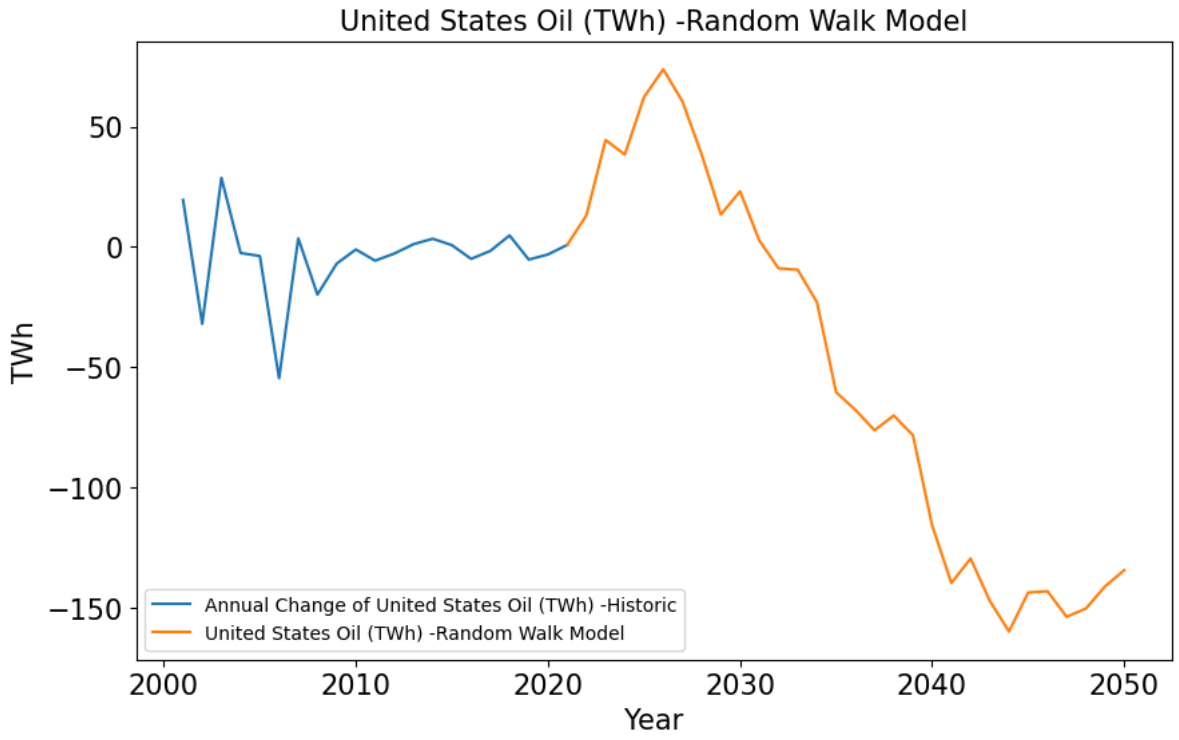


Looks like there's still a trend line, but it does pass stationarity, so I'm moving forward to the baseline random walk model.

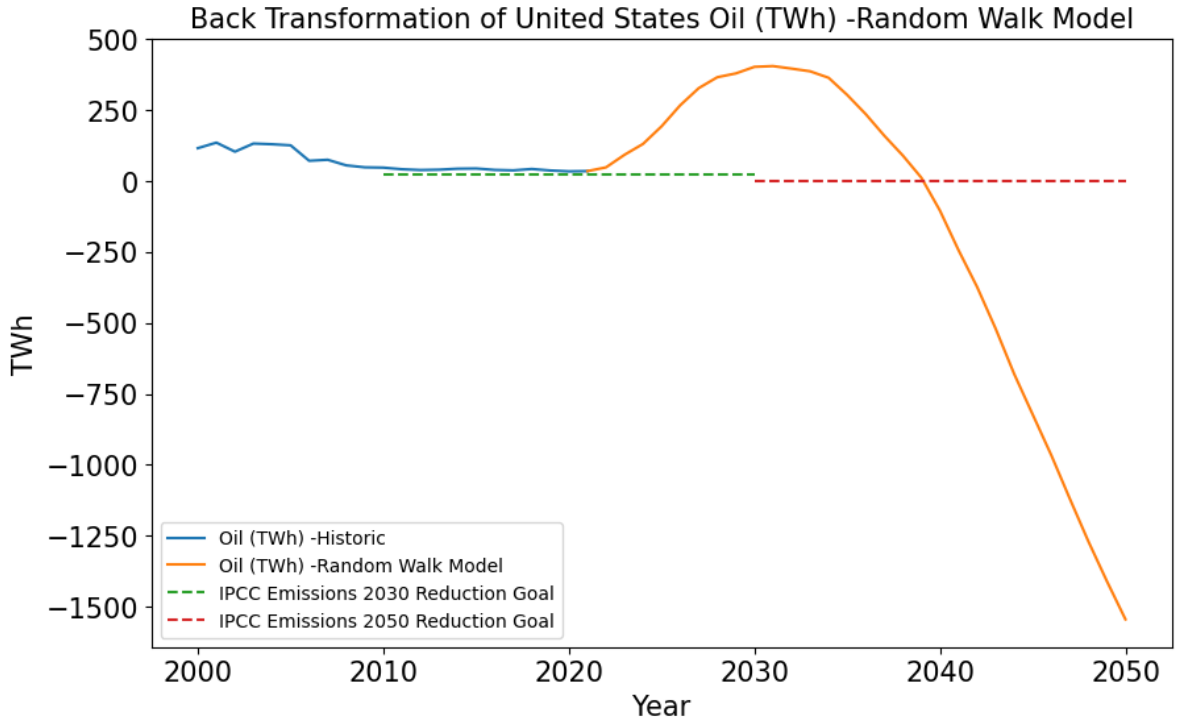
7.3.4 Random Walk Model

In [86]:
```
pd.set_option('display.max_rows',None)
y_hat_proj = random_walk(model_data_t.iloc[:,5:6])
y_hat_proj.columns = [model_data.columns[2]+' -Random Walk Model']

# Plot the transformed Random Walk
pred_graph(model_data_t.iloc[:,5:6], y_hat_proj.iloc[:,0:1])
```

```
Random Walk with Standard Deviation of Transformed Data set: 16.18
```



In [87]:
```
# Prepare the DataFrame for Back Transformation
ranplot = model_data_t.iloc[:,4:6].copy()
ranplot.rename(columns={str(ranplot.columns[1]): str(y_hat_proj.columns[0])},
               inplace=True)

ranplot = pd.concat([ranplot,y_hat_proj],axis=0)

# Perform Back Transformation
y_hat_proj = inv_difference(ranplot)

#Plot the new graph
pred_graph(model_data_t.iloc[:,4:5], y_hat_proj.iloc[22:,0:1], 14)
```

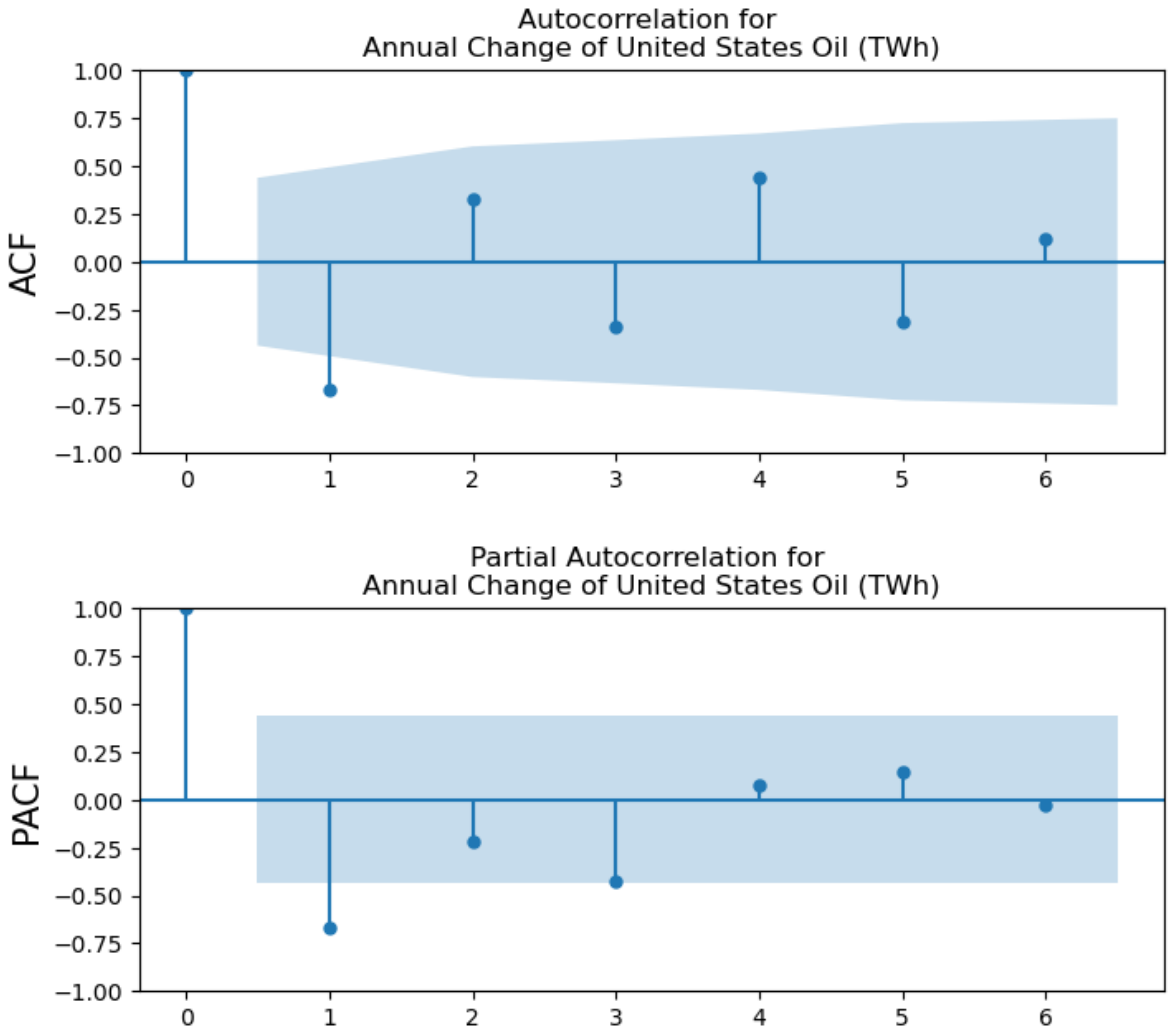


This model is as unreliable as it's name would imply, random walk. I think any of these random walks that oscillate about zero in transformed state are going to appear really strange after transformed back.

Now I'll move forward with an ARMA model. First, I need to evaluate the ACF and PACF plots to find the best Autoregressive and Moving Average terms.

7.3.5 Evaluating Initial AR and MA Values with ACF and PACF Plots

In [88]:
```
plotacf(model_data_t.iloc[:,5:6], lags = 6)
plotpacf(model_data_t.iloc[:,5:6], lags = 6)
```

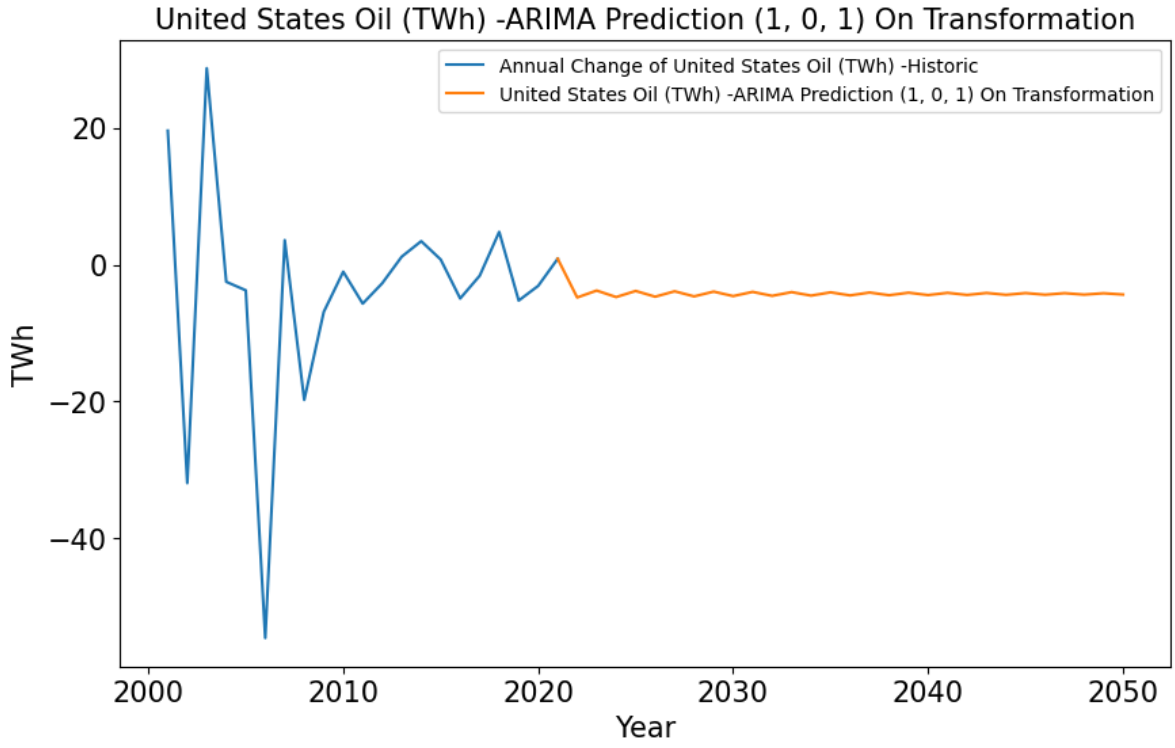


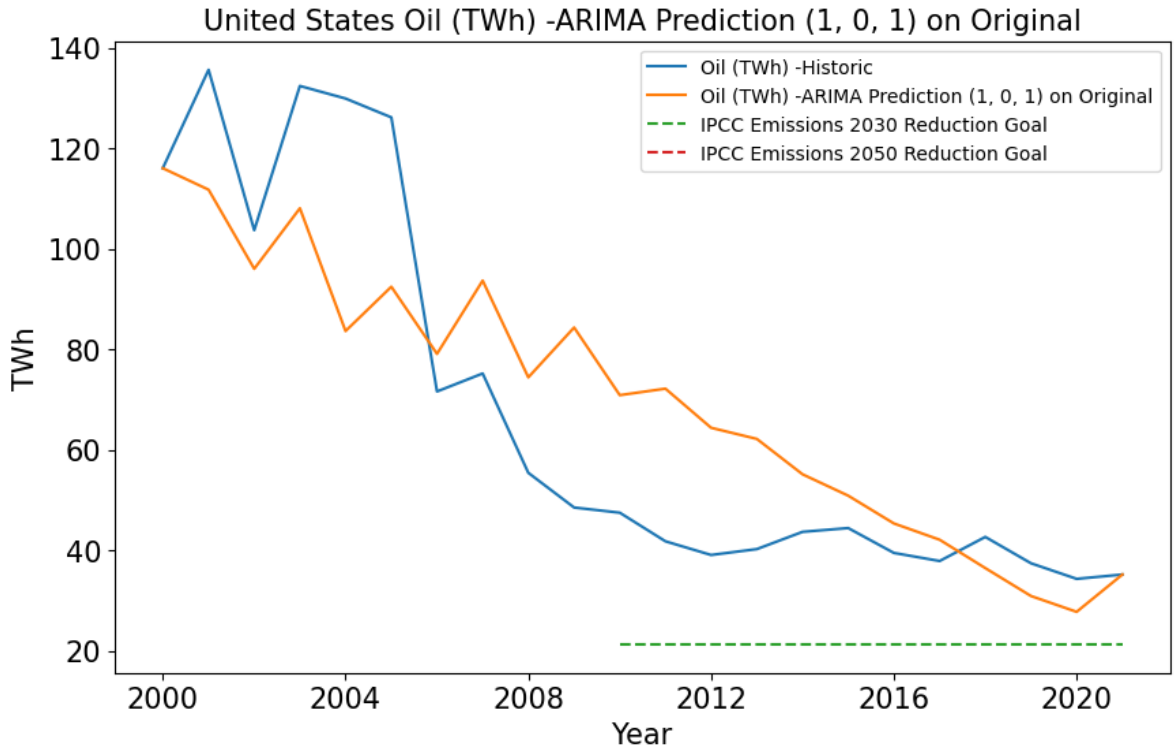
The ACF breaches at 1, and PACF definitely breaches at 1 and 3. I will try both (1,0,1) and (3,0,1) for my initial ARMA models.

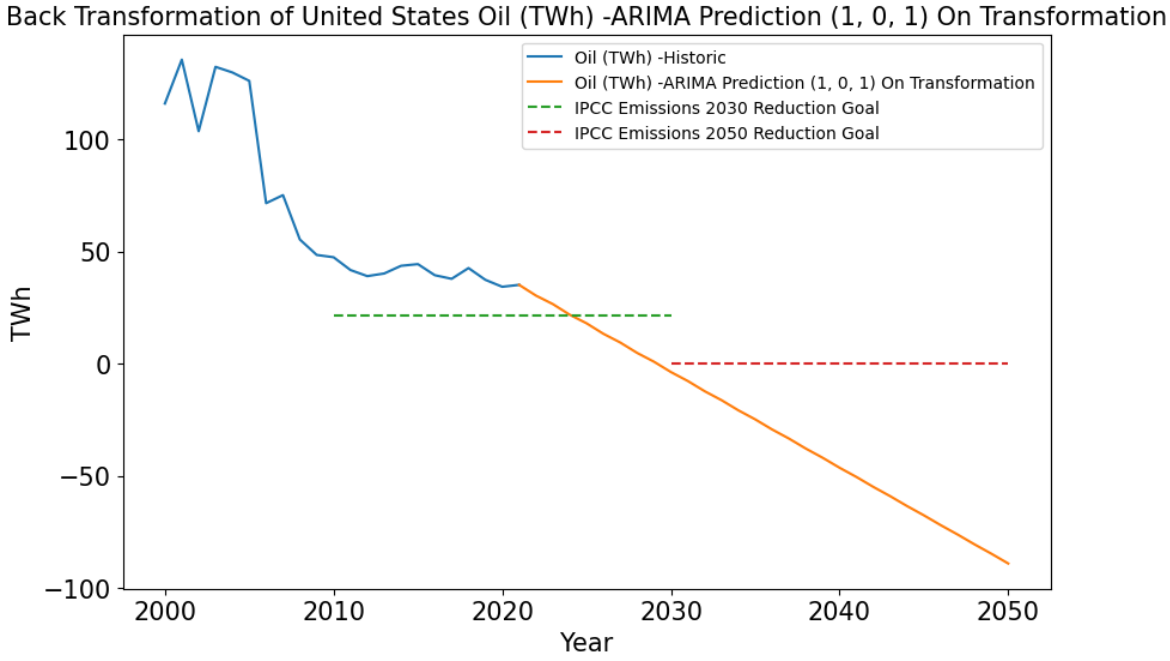
7.3.6 ARMA Model

In [89]:
```
order = (1,0,1)
model_summary(model_data_t.iloc[:,4:6], order, 22, 50, 14,
          tran = 'inv_difference')
```

```
ARIMA: (1, 0, 1), RMSE=20.93, AIC=179.01
********************************************************************************
**
United States Oil (TWh) model results.

                                      SARIMAX Results
==============================================================================
=======================
Dep. Variable:     Annual Change of United States Oil (TWh)   No. Observation
s:                   22
Model:                                   ARIMA(1, 0, 1)   Log Likelihood
-85.504
Date:                                  Sat, 29 Apr 2023   AIC
179.008
Time:                                          06:17:43   BIC
183.372
Sample:                                      01-01-2000   HQIC
180.036
                                           - 01-01-2021
Covariance Type:                                    opg
==============================================================================
=
                 coef    std err          z      P>|z|      [0.025      0.97
5]
------------------------------------------------------------------------------
-
const         -4.2724      3.291     -1.298      0.194     -10.722       2.17
7
ar.L1         -0.9392      0.116     -8.113      0.000      -1.166      -0.71
2
ma.L1          0.6933      0.327      2.122      0.034       0.053       1.33
4
sigma2       194.9214     85.379      2.283      0.022      27.581     362.26
2
==============================================================================
======
Ljung-Box (L1) (Q):                   0.15   Jarque-Bera (JB):
6.82
Prob(Q):                              0.69   Prob(JB):
0.03
Heteroskedasticity (H):               0.06   Skew:
-1.07
Prob(H) (two-sided):                  0.00   Kurtosis:
4.70
==============================================================================
======

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (comple
x-step).
```

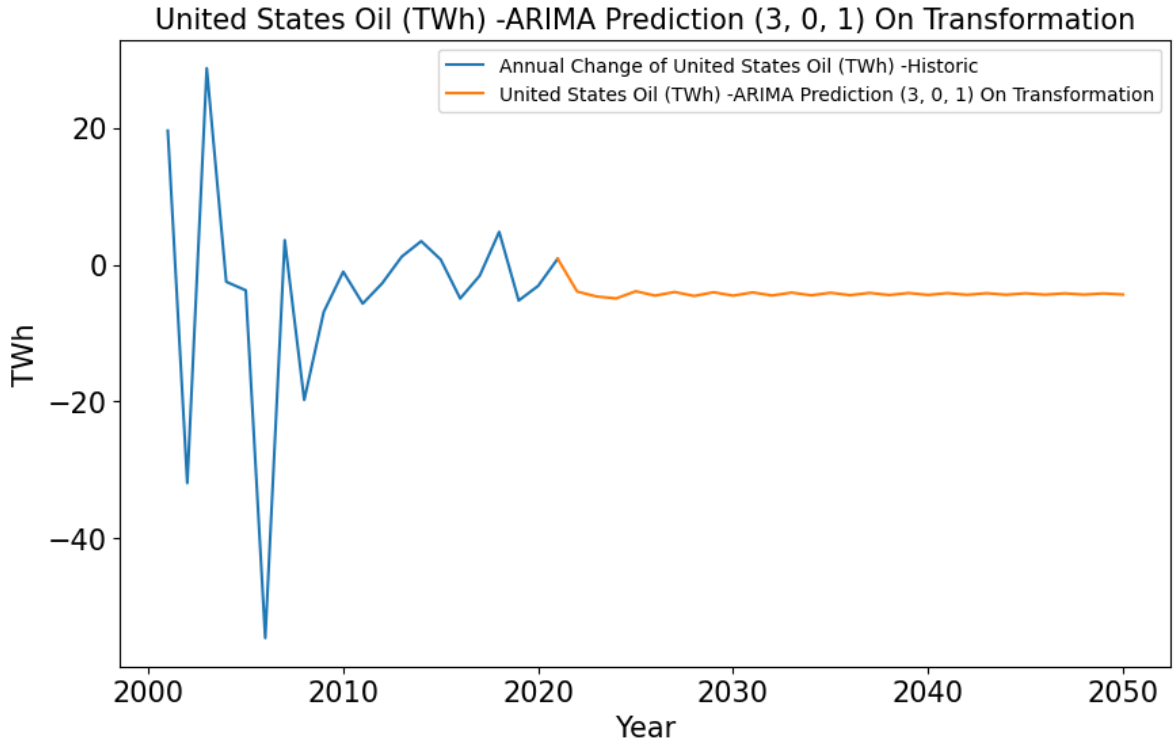

United States Oil (TWh) -ARIMA Prediction (1, 0, 1) On Transformation
Annual Change of United States Oil (TWh) -Historic
United States Oil (TWh) -ARIMA Prediction (1, 0, 1) On Transformation
TWh
Year
20
0
−20
−40
2000
2010
2020
2030
2040
2050

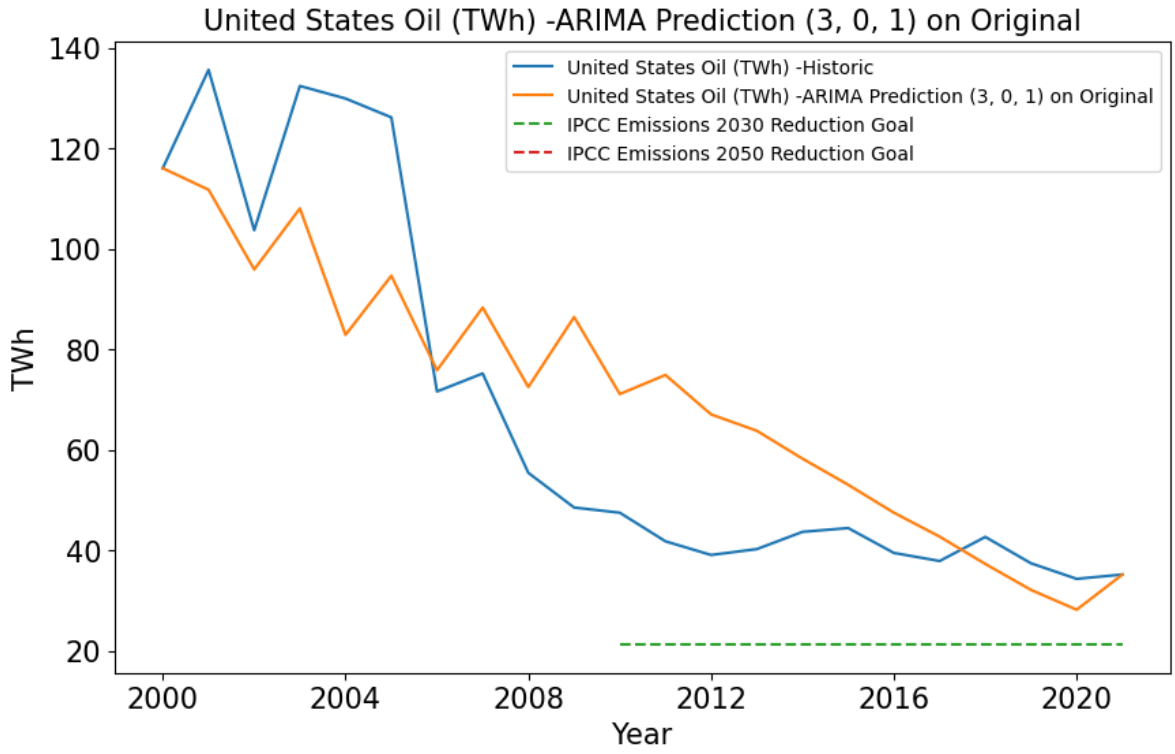

United States Oil (TWh) -ARIMA Prediction (1, 0, 1) on Original
Oil (TWh) -Historic
Oil (TWh) -ARIMA Prediction (1, 0, 1) on Original
IPCC Emissions 2030 Reduction Goal
IPCC Emissions 2050 Reduction Goal
TWh
Year
140
120
100
80
60
40
20
2000
2004
2008
2012
2016
2020

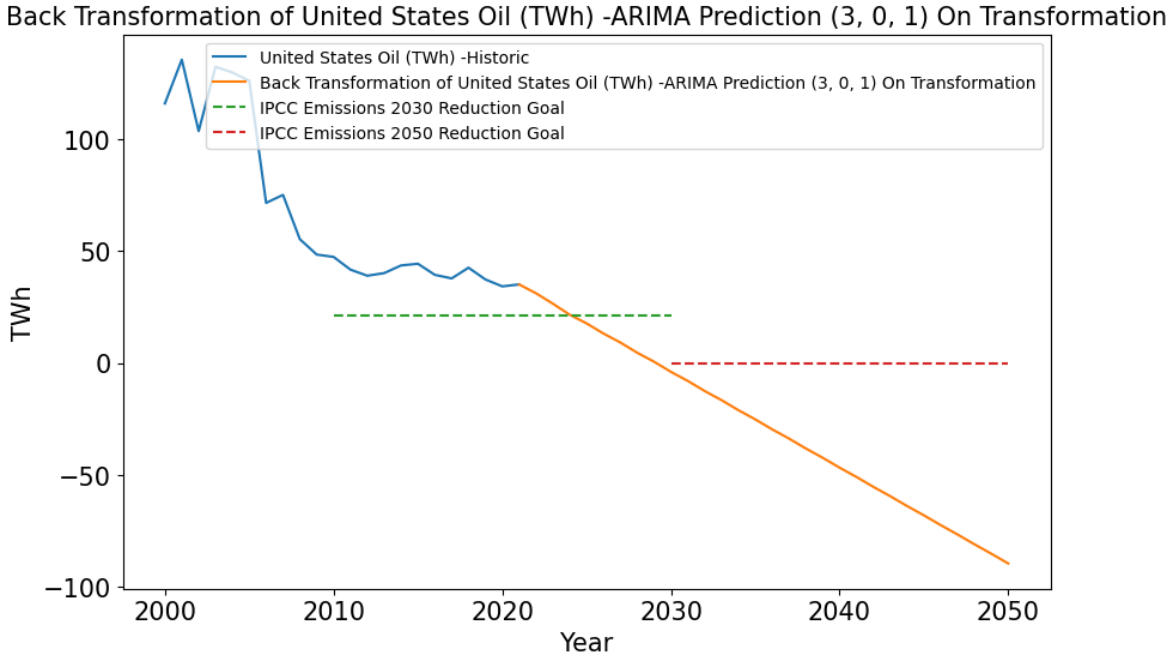

Back Transformation of United States Oil (TWh) -ARIMA Prediction (1, 0, 1) On Transformation
Oil (TWh) -Historic
Oil (TWh) -ARIMA Prediction (1, 0, 1) On Transformation
IPCC Emissions 2030 Reduction Goal
IPCC Emissions 2050 Reduction Goal
TWh
100
50
0
−50
−100
2000
2010
2020
2030
2040
2050
Year

In [90]:
```
order = (3,0,1)
model_summary(model_data_t.iloc[:,4:6], order, 22, 50, 0,
          tran = 'inv_difference')
```

```
ARIMA: (3, 0, 1), RMSE=21.29, AIC=182.45
**********************************************************************
**
United States Oil (TWh) model results.

                                      SARIMAX Results
==========================================================================
=======================
Dep. Variable:     Annual Change of United States Oil (TWh)   No. Observation
s:                   22
Model:                                   ARIMA(3, 0, 1)   Log Likelihood
-85.226
Date:                                  Sat, 29 Apr 2023   AIC
182.453
Time:                                          06:17:44   BIC
188.999
Sample:                                     01-01-2000   HQIC
183.995
                                          - 01-01-2021
Covariance Type:                                    opg
==========================================================================
=
                 coef    std err          z      P>|z|      [0.025      0.97
5]
--------------------------------------------------------------------------
-
const         -4.2835      2.944     -1.455      0.146     -10.053       1.48
6
ar.L1         -0.7214      0.714     -1.011      0.312      -2.120       0.67
8
ar.L2          0.0192      0.452      0.043      0.966      -0.866       0.90
4
ar.L3         -0.1794      0.360     -0.499      0.618      -0.884       0.52
6
ma.L1          0.5384      0.739      0.729      0.466      -0.910       1.98
6
sigma2       187.7522     93.959      1.998      0.046       3.595     371.90
9
==========================================================================
======
Ljung-Box (L1) (Q):                   0.05   Jarque-Bera (JB):
2.38
Prob(Q):                              0.82   Prob(JB):
0.30
Heteroskedasticity (H):               0.07   Skew:
-0.77
Prob(H) (two-sided):                  0.00   Kurtosis:
3.49
==========================================================================
======

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (comple
x-step).
```


United States Oil (TWh) -ARIMA Prediction (3, 0, 1) On Transformation
Annual Change of United States Oil (TWh) -Historic
United States Oil (TWh) -ARIMA Prediction (3, 0, 1) On Transformation
TWh
Year
20
0
−20
−40
2000
2010
2020
2030
2040
2050


United States Oil (TWh) -ARIMA Prediction (3, 0, 1) on Original
United States Oil (TWh) -Historic
United States Oil (TWh) -ARIMA Prediction (3, 0, 1) on Original
IPCC Emissions 2030 Reduction Goal
IPCC Emissions 2050 Reduction Goal
TWh
Year
140
120
100
80
60
40
20
2000
2004
2008
2012
2016
2020

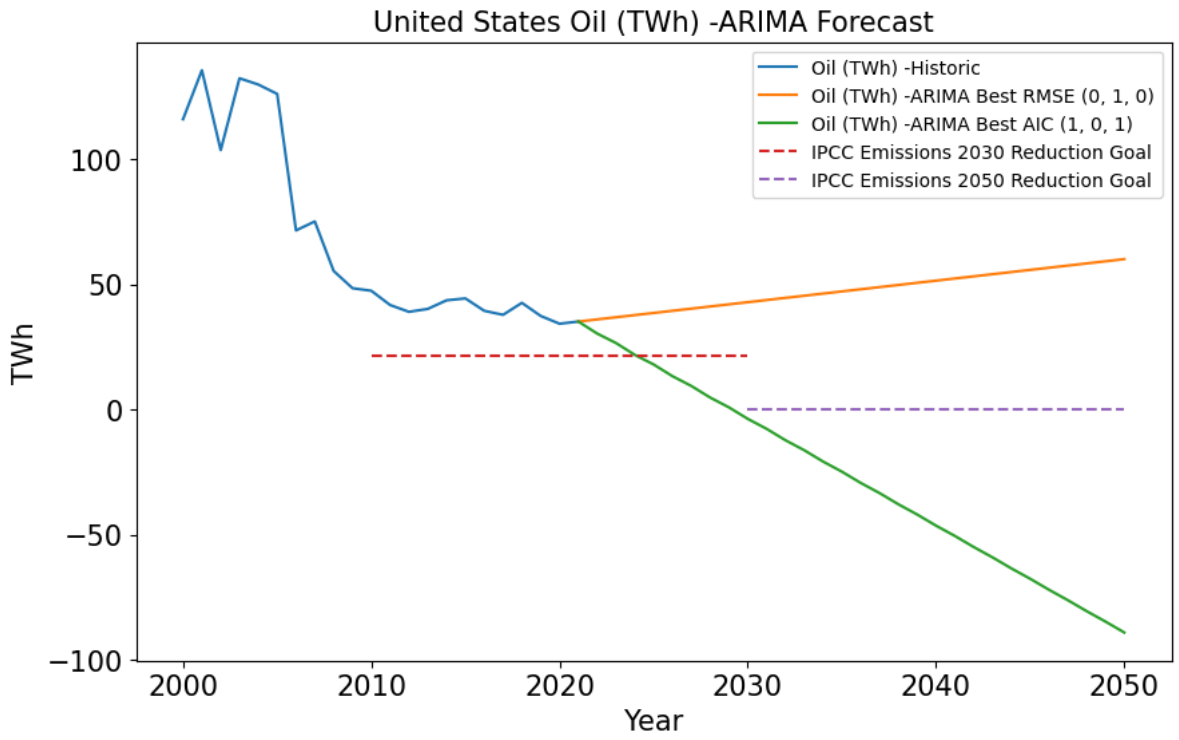


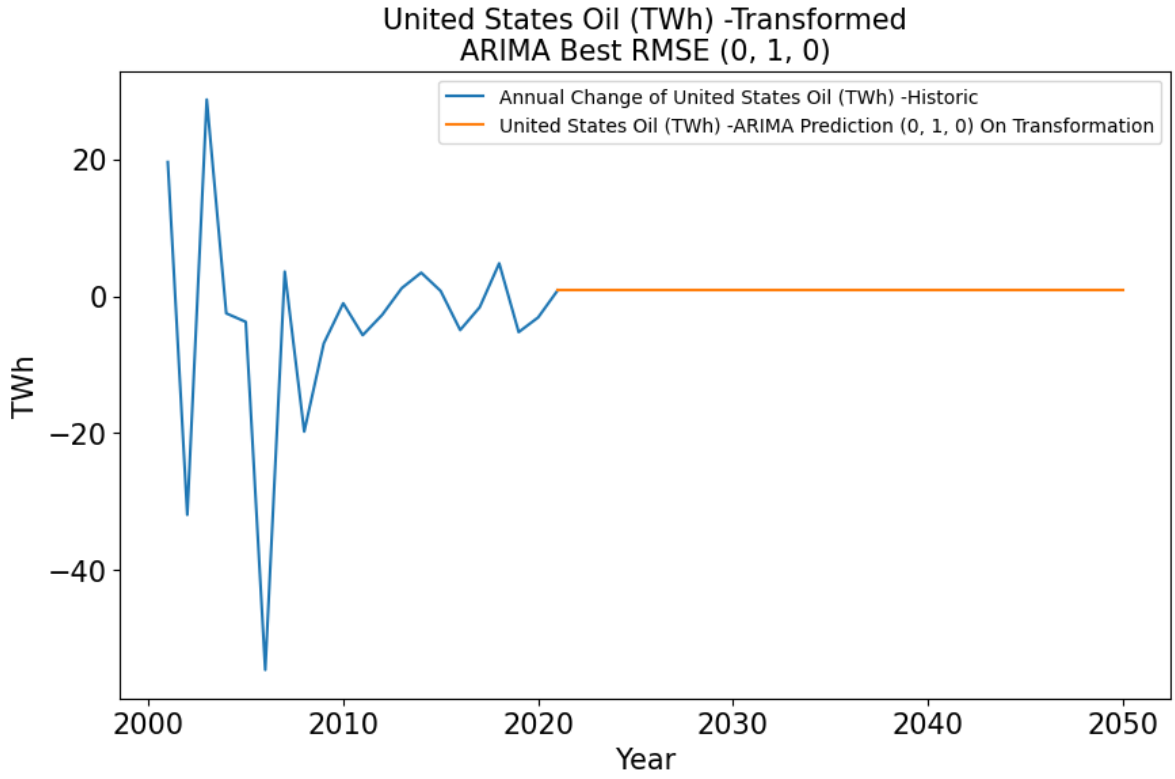
These models both look convincing. Oil as a power source for electricity is on its way out. As always, I'll see what the grid search says.

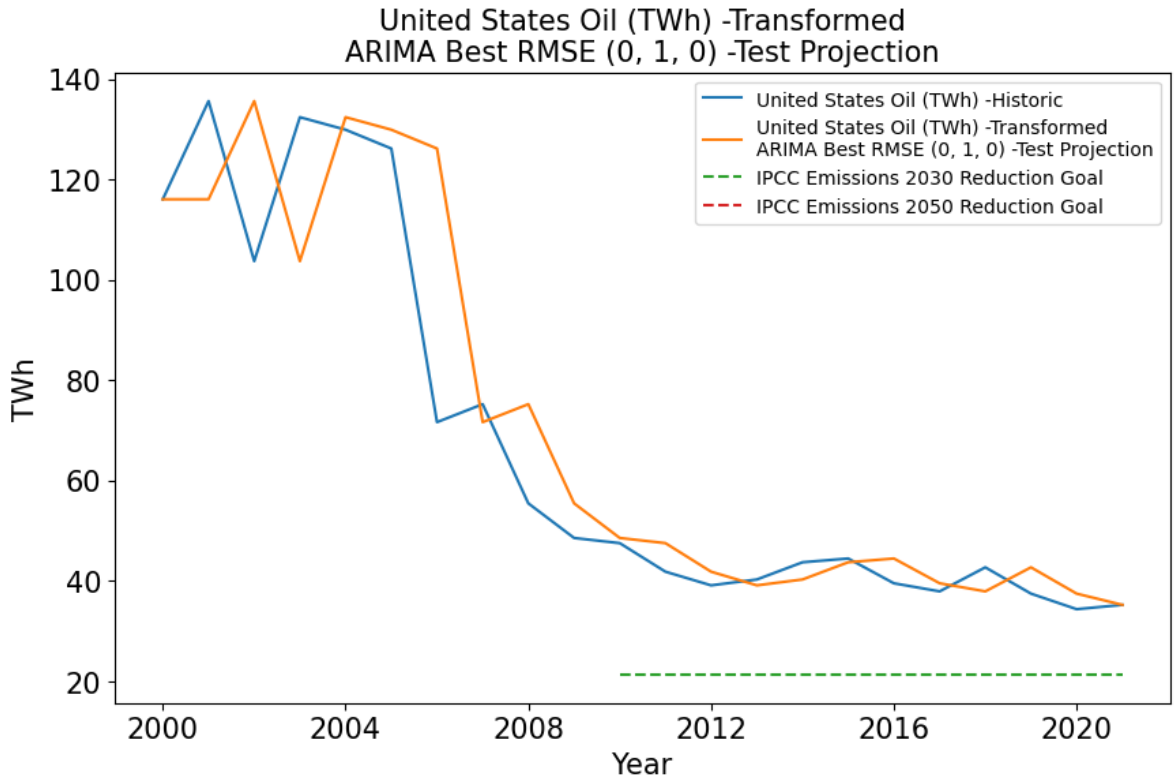
7.3.7 ARIMA Model and Grid Search

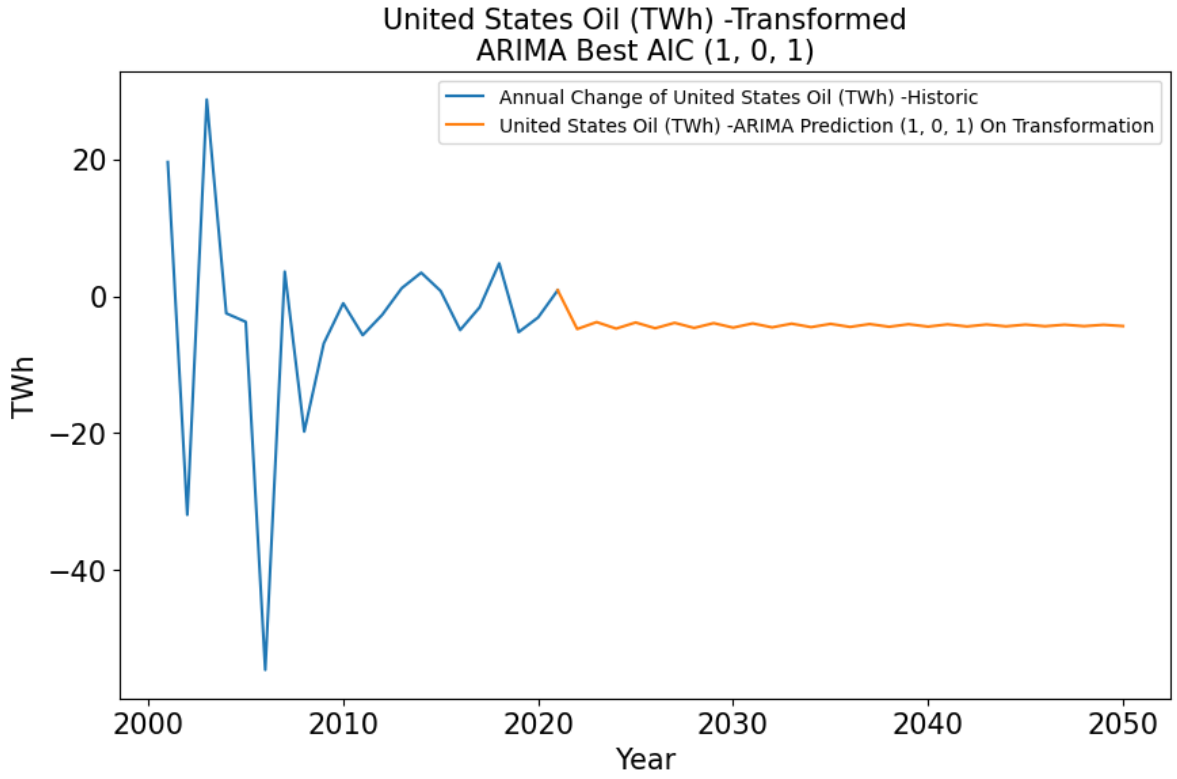
In [91]:
```
oil_rmse_cfg, oil_aic_cfg = arima_pdq(
                model_data_t.iloc[:,4:6], s=14, tran = 'inv_difference')
```

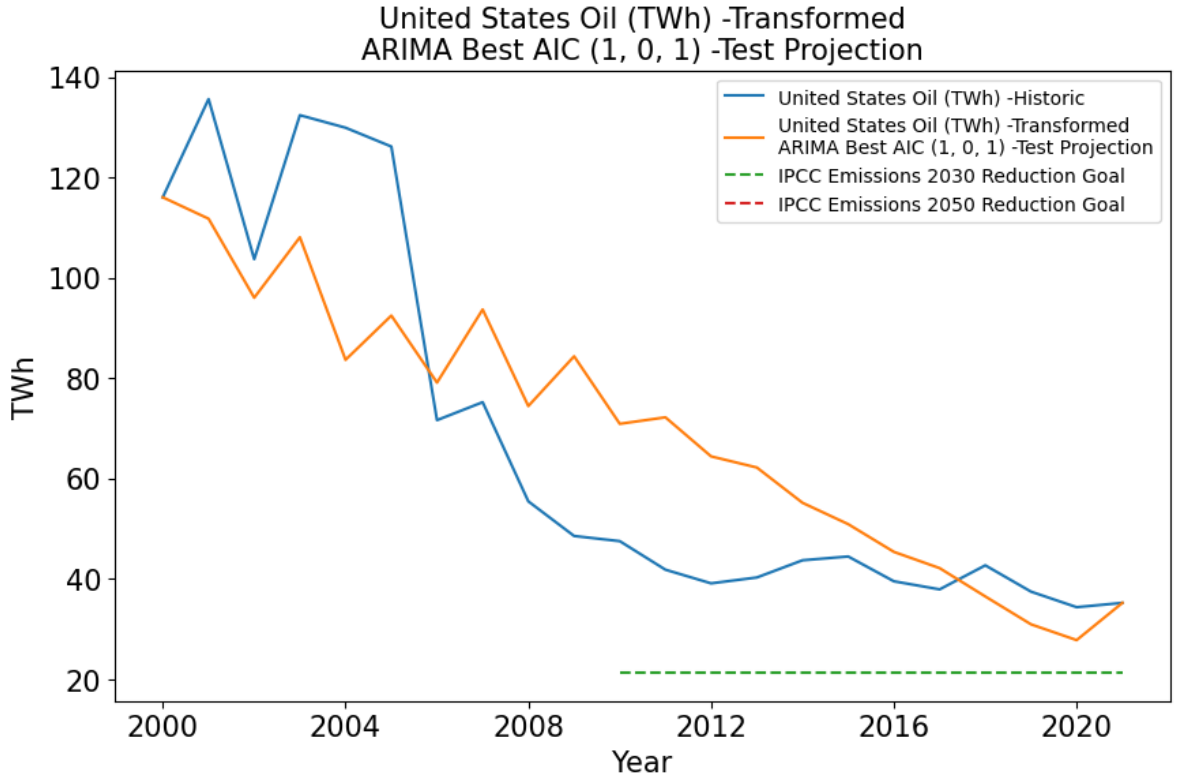
```
RMSE ARIMA: (0, 0, 0), RMSE= 21.52, AIC= 180.52, TWh-2050= -76.41
AIC  ARIMA: (0, 0, 1), RMSE= 25.38, AIC= 180.44, TWh-2050= -87.15
RMSE ARIMA: (0, 1, 0), RMSE= 16.25, AIC= 205.80, TWh-2050= 60.14
AIC  ARIMA: (1, 0, 0), RMSE= 25.08, AIC= 179.60, TWh-2050= -88.59
AIC  ARIMA: (1, 0, 1), RMSE= 20.93, AIC= 179.01, TWh-2050= -89.01
Best RMSE ARIMA: (0, 1, 0) RMSE= 16.25 AIC= 205.80, TWh-2050= 60.14
Best AIC  ARIMA: (1, 0, 1) RMSE= 20.93 AIC= 179.01, TWh-2050= -89.01
Graph_formatting produced error at (0, 1, 0) or (1, 0, 1)
```




United States Oil (TWh) -Transformed
ARIMA Best RMSE (0, 1, 0)
Annual Change of United States Oil (TWh) -Historic
United States Oil (TWh) -ARIMA Prediction (0, 1, 0) On Transformation
TWh
Year
20
0
−20
−40
2000
2010
2020
2030
2040
2050


United States Oil (TWh) -Transformed
ARIMA Best RMSE (0, 1, 0) -Test Projection
United States Oil (TWh) -Historic
United States Oil (TWh) -Transformed
ARIMA Best RMSE (0, 1, 0) -Test Projection
IPCC Emissions 2030 Reduction Goal
IPCC Emissions 2050 Reduction Goal
TWh
Year
140
120
100
80
60
40
20
2000
2004
2008
2012
2016
2020



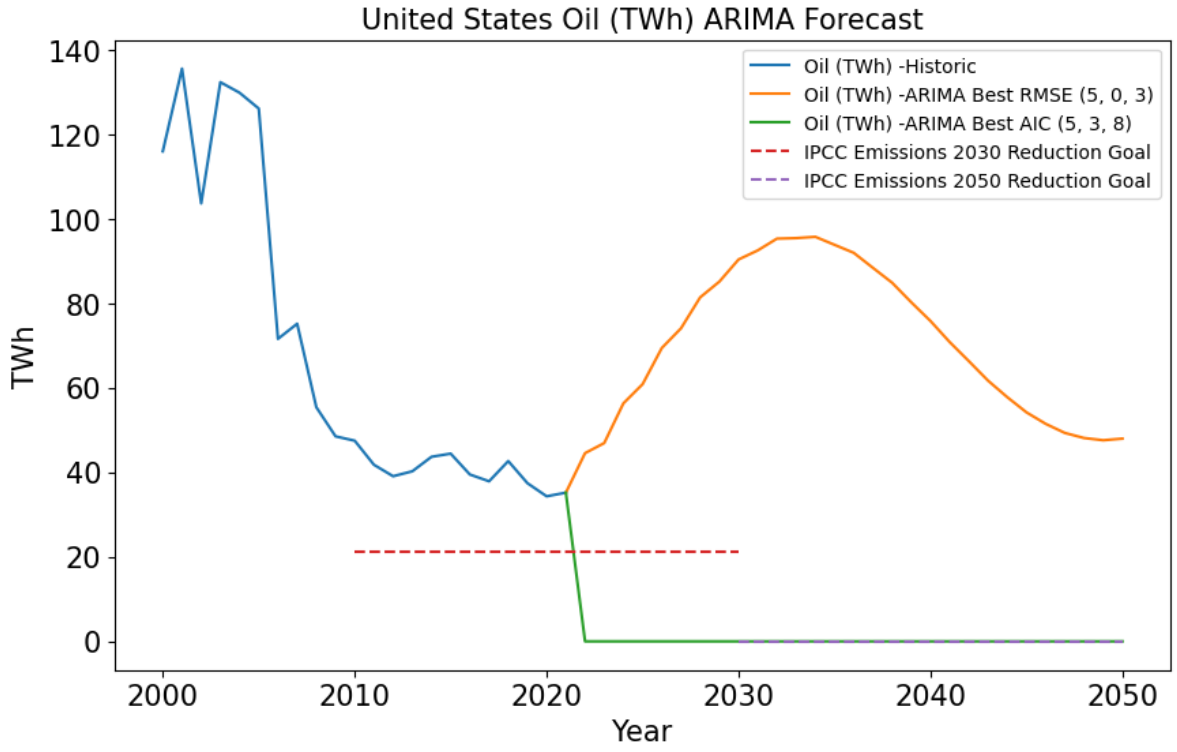


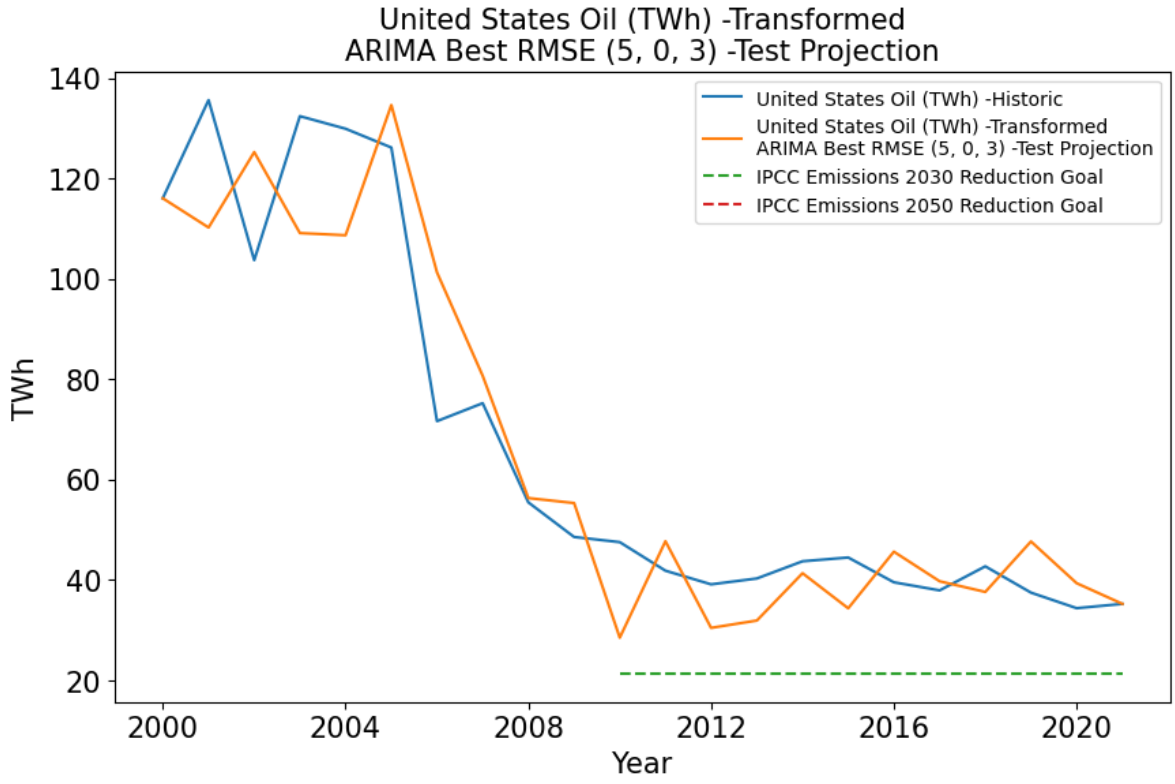
It looks as though Coal is declining so quick, it may be gone by 2040. Also, it has nearly already hit the 2030 IPCC goal, and it is likely to hit the IPCC 2040 goal.

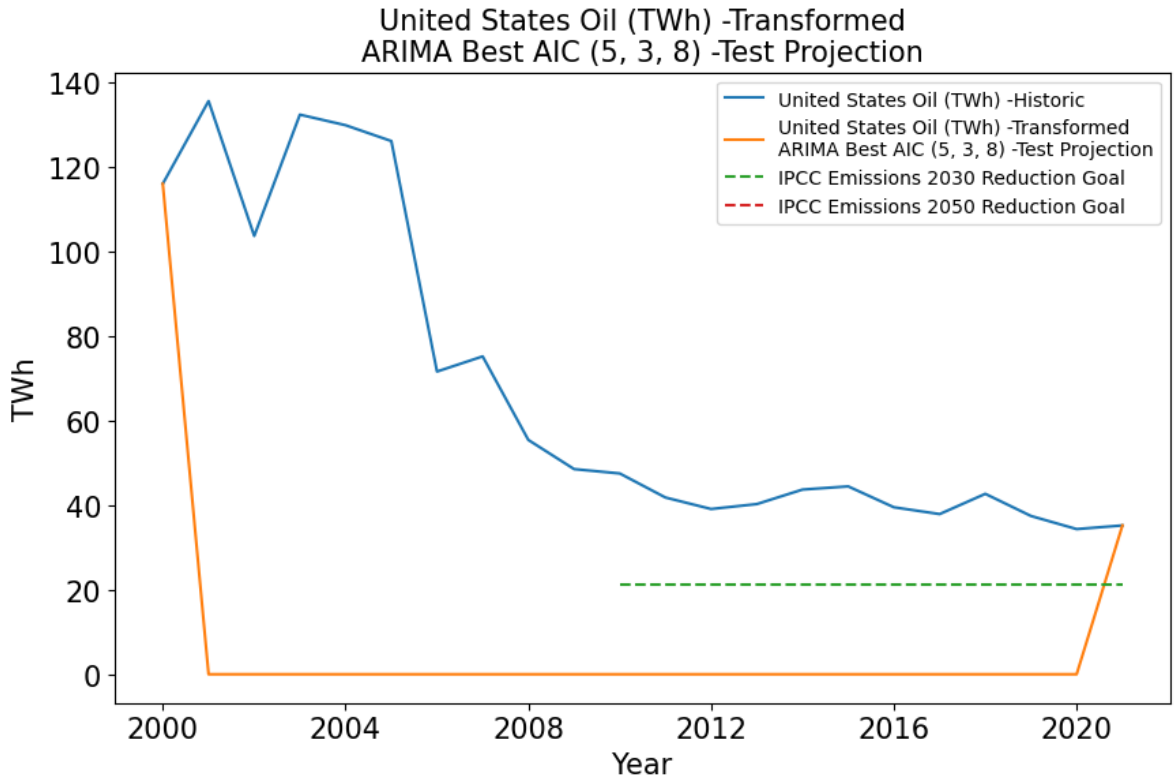
7.3.8 ARIMA Without Transformation

In [92]:
```
oil_rmse_cfg, oil_aic_cfg = arima_pdq_no_tran(
                                        model_data.iloc[:,2:3], 22, 50, s=14)
```

```
United States Oil (TWh) Grid Search:
RMSE ARIMA: (0, 0, 0), RMSE= 34.40, AIC= 224.82, TWh-2050= 67.19
RMSE ARIMA: (0, 0, 1), RMSE= 24.37, AIC= 212.29, TWh-2050= 66.83
RMSE ARIMA: (0, 0, 2), RMSE= 18.57, AIC= 200.38, TWh-2050= 67.13
RMSE ARIMA: (0, 0, 3), RMSE= 18.38, AIC= 202.17, TWh-2050= 66.94
RMSE ARIMA: (0, 0, 4), RMSE= 15.54, AIC= 197.50, TWh-2050= 69.35
RMSE ARIMA: (0, 0, 5), RMSE= 14.75, AIC= 196.62, TWh-2050= 67.86
RMSE ARIMA: (0, 0, 7), RMSE= 13.71, AIC= 197.95, TWh-2050= 67.31
RMSE ARIMA: (0, 0, 8), RMSE= 13.57, AIC= 198.84, TWh-2050= 67.10
AIC  ARIMA: (0, 1, 0), RMSE= 16.25, AIC= 179.67, TWh-2050= 35.20
AIC  ARIMA: (0, 2, 1), RMSE= 20.57, AIC= 176.14, TWh-2050= -76.35
AIC  ARIMA: (1, 2, 1), RMSE= 20.16, AIC= 175.94, TWh-2050= -80.24
AIC  ARIMA: (1, 2, 2), RMSE= 19.50, AIC= 175.18, TWh-2050= -88.68
RMSE ARIMA: (4, 0, 4), RMSE= 13.44, AIC= 197.77, TWh-2050= 46.07
RMSE ARIMA: (5, 0, 3), RMSE= 13.44, AIC= 197.83, TWh-2050= 47.99
AIC  ARIMA: (5, 3, 8), RMSE= 72.01, AIC= 28.00, TWh-2050= 0.00
Best RMSE ARIMA: (5, 0, 3) RMSE= 13.44 AIC= 197.83, TWh-2050= 47.99
Best AIC  ARIMA: (5, 3, 8) RMSE= 72.01 AIC= 28.00, TWh-2050= 0.00
```




United States Oil (TWh) -Transformed
ARIMA Best RMSE (5, 0, 3) -Test Projection
United States Oil (TWh) -Historic
United States Oil (TWh) -Transformed
ARIMA Best RMSE (5, 0, 3) -Test Projection
IPCC Emissions 2030 Reduction Goal
IPCC Emissions 2050 Reduction Goal
TWh
Year
140
120
100
80
60
40
20
2000
2004
2008
2012
2016
2020

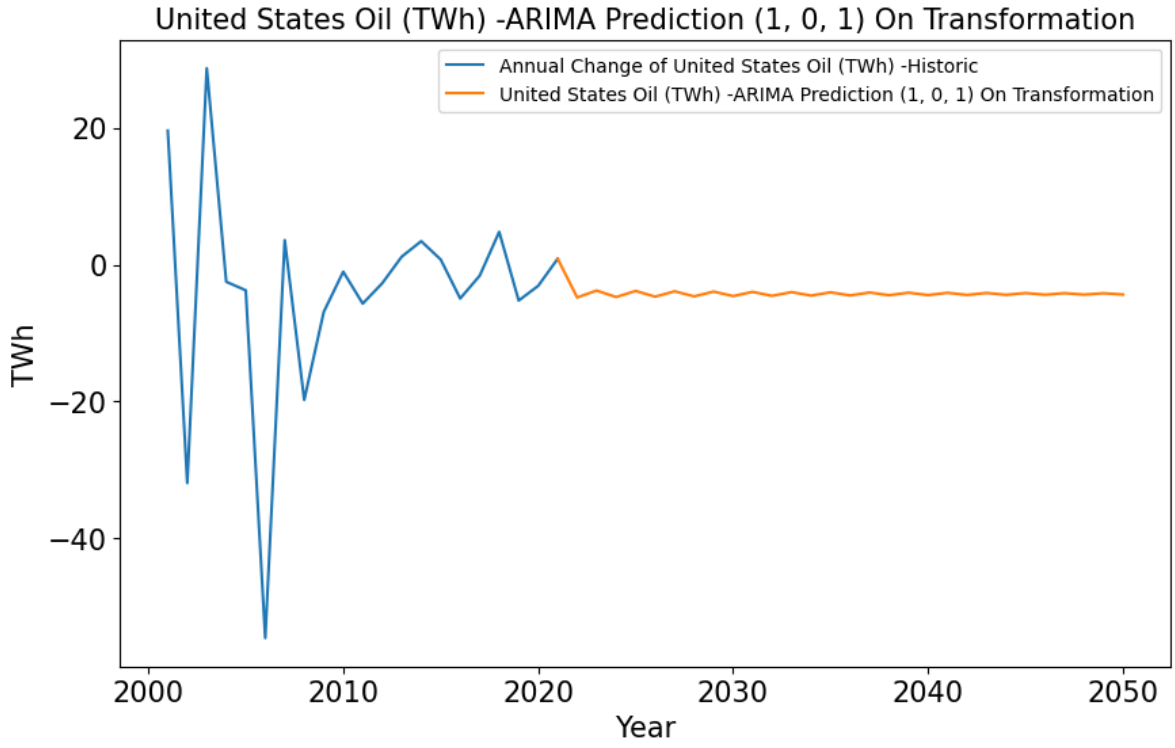

United States Oil (TWh) -Transformed
ARIMA Best AIC (5, 3, 8) -Test Projection
United States Oil (TWh) -Historic
United States Oil (TWh) -Transformed
ARIMA Best AIC (5, 3, 8) -Test Projection
IPCC Emissions 2030 Reduction Goal
IPCC Emissions 2050 Reduction Goal
TWh
Year
140
120
100
80
60
40
20
0
2000
2004
2008
2012
2016
2020

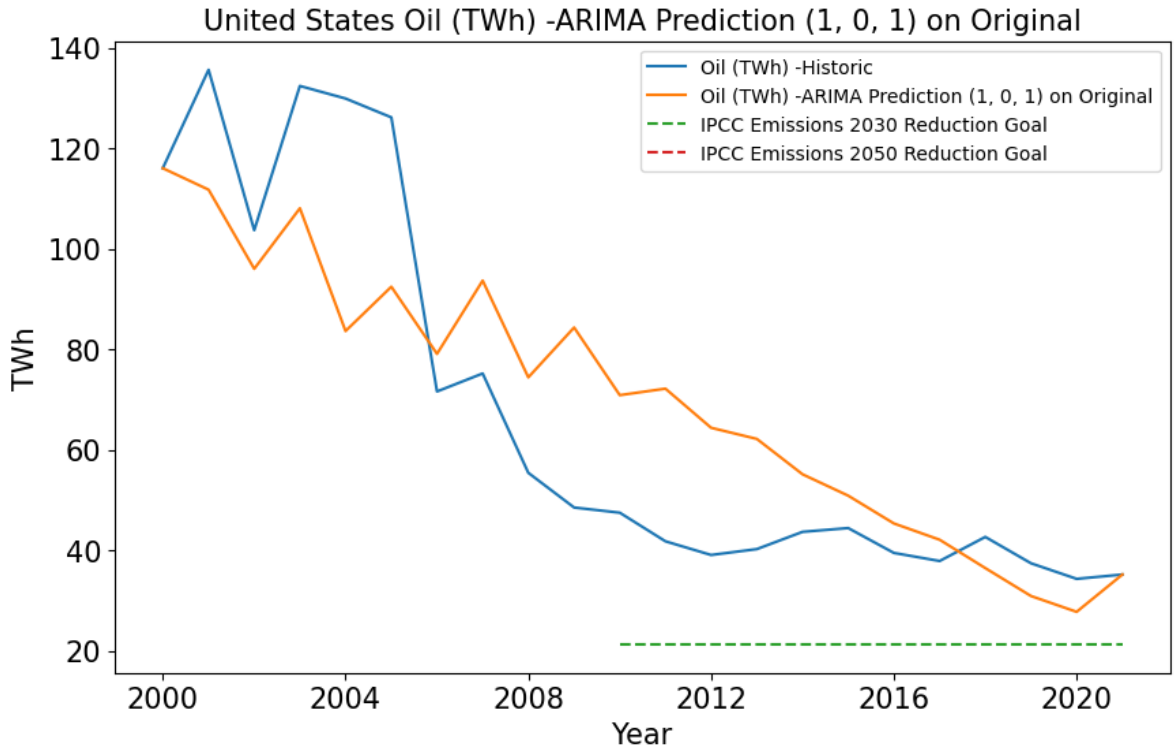
7.3.9 Model Selection

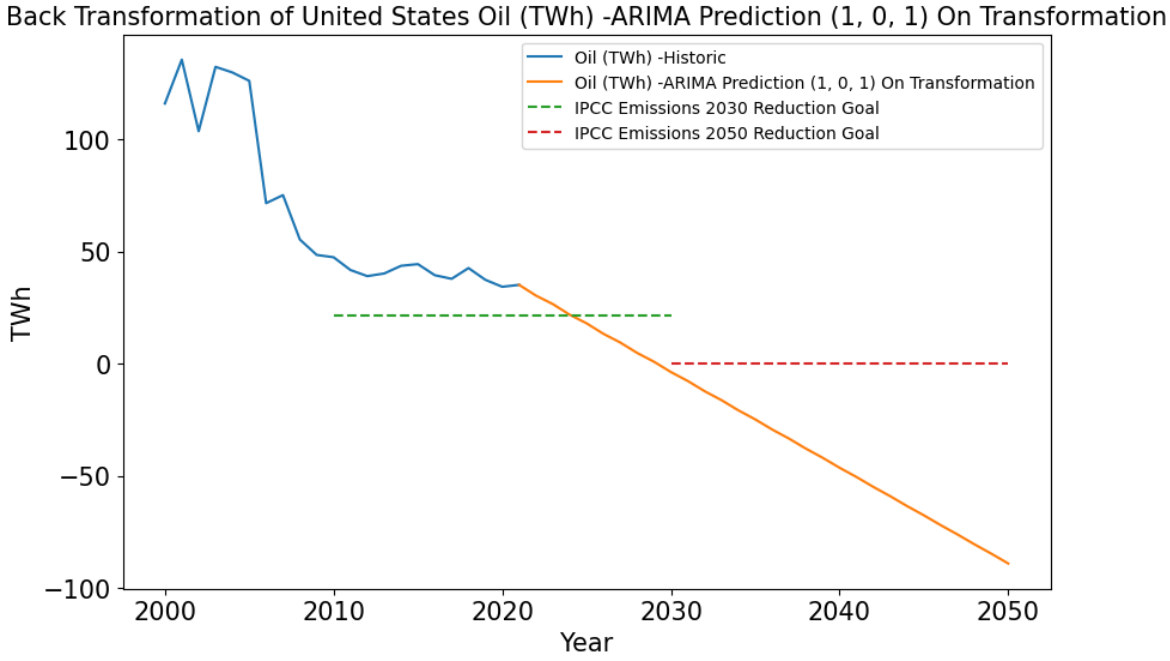
In [93]:
```
#Deleteme
stored_oil = selected_models.copy()
selected_models = stored_oil.copy()
```

In [94]:
```
selected_models = stored_oil.copy()
df_oil = model_data_t.iloc[:,4:6]

#                         Model,       df_o,  pdq,    tran,            n,PDQs
selected_models.append(['ARIMA_tran', df_oil,(1,0,1),'inv_difference',0,None])

m = 2
model_summary(selected_models[m][1], selected_models[m][2], 22, 50, 14,
              selected_models[m][3], selected_models[m][4])
```

```
ARIMA: (1, 0, 1), RMSE=20.93, AIC=179.01
********************************************************************************
United States Oil (TWh) model results.

                                      SARIMAX Results
==========================================================================================
Dep. Variable:     Annual Change of United States Oil (TWh)   No. Observations:                   22
Model:                                   ARIMA(1, 0, 1)   Log Likelihood                -85.504
Date:                                  Sat, 29 Apr 2023   AIC                            179.008
Time:                                          06:18:52   BIC                            183.372
Sample:                                      01-01-2000   HQIC                           180.036
                                           - 01-01-2021
Covariance Type:                                    opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const         -4.2724      3.291     -1.298      0.194     -10.722       2.177
ar.L1         -0.9392      0.116     -8.113      0.000      -1.166      -0.712
ma.L1          0.6933      0.327      2.122      0.034       0.053       1.334
sigma2       194.9214     85.379      2.283      0.022      27.581     362.262
===================================================================================
Ljung-Box (L1) (Q):                   0.15   Jarque-Bera (JB):                 6.82
Prob(Q):                              0.69   Prob(JB):                         0.03
Heteroskedasticity (H):               0.06   Skew:                            -1.07
Prob(H) (two-sided):                  0.00   Kurtosis:                         4.70
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```


United States Oil (TWh) -ARIMA Prediction (1, 0, 1) On Transformation
Annual Change of United States Oil (TWh) -Historic
United States Oil (TWh) -ARIMA Prediction (1, 0, 1) On Transformation
TWh
Year
20
0
−20
−40
2000
2010
2020
2030
2040
2050


United States Oil (TWh) -ARIMA Prediction (1, 0, 1) on Original
Oil (TWh) -Historic
Oil (TWh) -ARIMA Prediction (1, 0, 1) on Original
IPCC Emissions 2030 Reduction Goal
IPCC Emissions 2050 Reduction Goal
TWh
Year
140
120
100
80
60
40
20
2000
2004
2008
2012
2016
2020



It's clear these oil plants will be shut down soon. I think this graph is reasonable. It's also consistent. If I have no d term, these graphs largly look the same no matter the p or q.

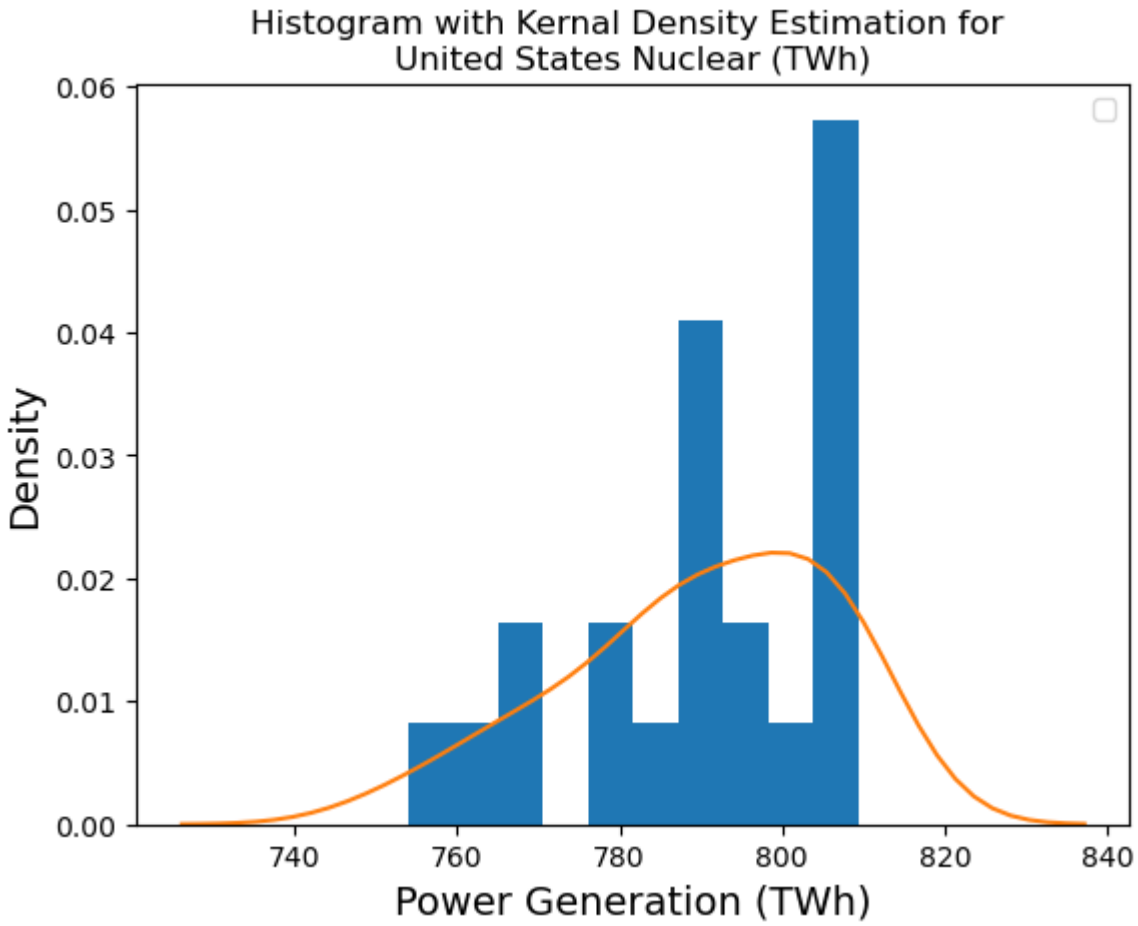
7.4 Nuclear

7.4.1 Distribution Investigation

In [95]:
```
stored_model_data = model_data.copy()
stored_model_data_t = model_data_t.copy()
```

In [96]:
```
model_data = stored_model_data.copy()
model_data_t = stored_model_data_t.copy()
```

In [97]:
```
hist(model_data.iloc[:,3:4])
```



In [98]:
```
stats_block = s_block(model_data.iloc[:,3:4], stats_block)
stats_block
```

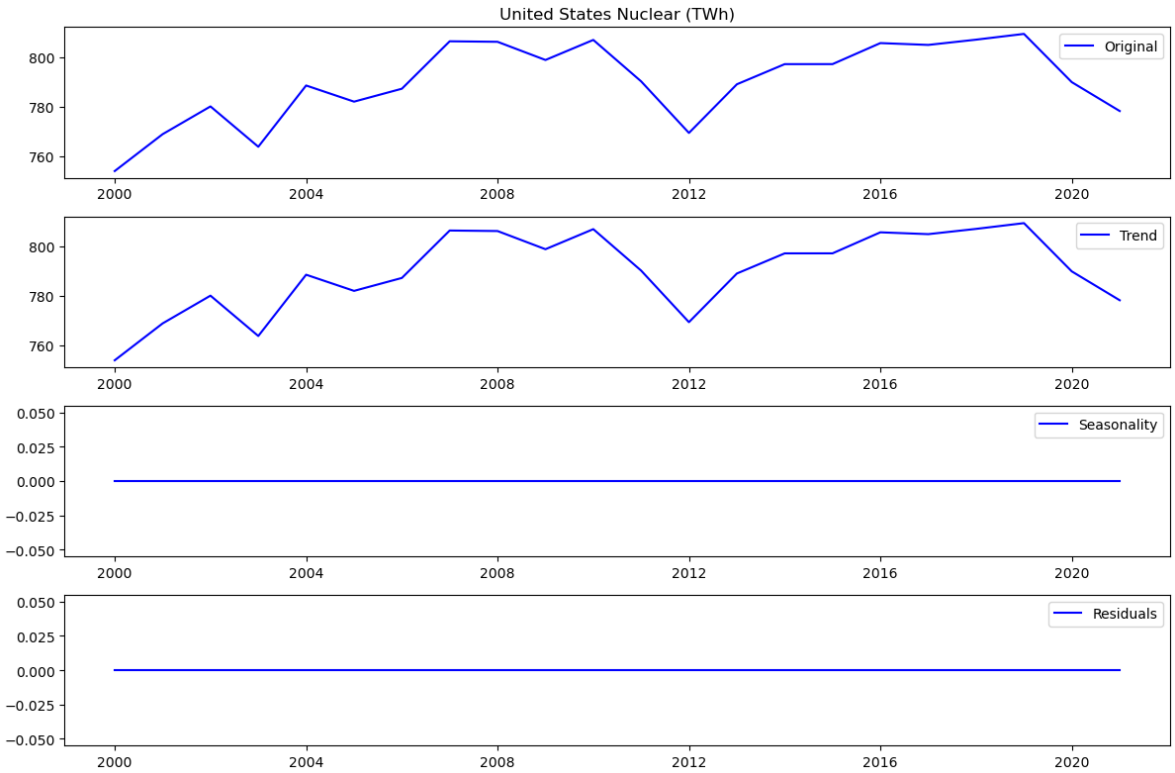
Out[98]:

	Dataset	Mean	Median	Standard Deviation	Skew	Fisher Kurtosis
0	United States Coal (TWh)	1,607.17	1,744.66	399.15	-0.68	-0.90
1	United States Gas (TWh)	1,060.39	1,000.70	325.82	0.29	-1.21
2	United States Oil (TWh)	67.20	45.97	36.59	0.89	-0.92
3	United States Nuclear (TWh)	790.04	790.04	15.55	-0.64	-0.53

This has a slightly negative skew distribution with wide and flat tails.

7.4.2 Decomposing The Data

In [99]:
```
decomp_graph(model_data.iloc[:,3:4])
```

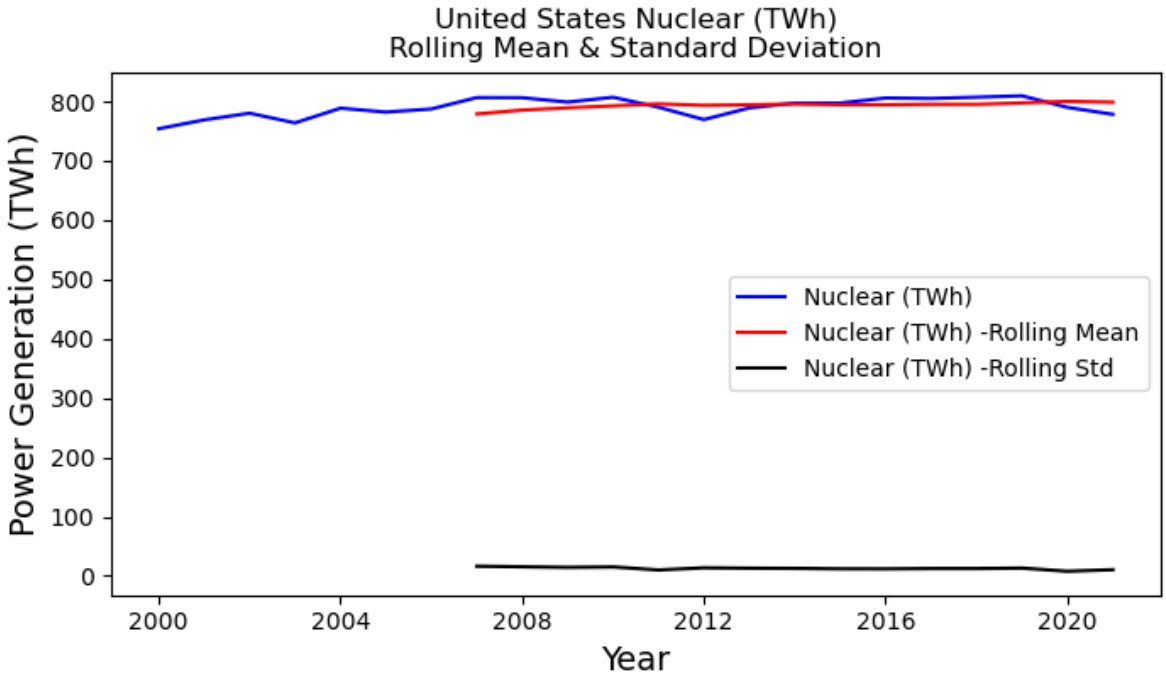


No seasonality or residuals. I'll check for stationarity.

7.4.3 Checking for Stationarity and Flattening

In [100]:
```
pd.set_option('display.max_rows', 30)
stationarity_check(model_data.iloc[:,3:4], s=14)
```

```
United States Nuclear (TWh)
Results of Dickey-Fuller test:

Test Statistic                -1.69
p-value                        0.44
#Lags Used                     8.00
Number of Observations Used   13.00
Critical Value (1%)           -4.07
Critical Value (5%)           -3.13
Critical Value (10%)          -2.70
dtype: float64
```



This is not closer to stationary than the other graphs I've looked at. After playing around with transformations, I found that if I subtracted the rolling average of eight values, I can get it stationary.

In [101]:
```
stored = model_data_t.copy()
```

In [102]:
```
model_data_t = stored.copy()
t = subtract_roll_mean(model_data.iloc[:,3:4], n=8)
model_data_t.insert(7,t.columns[0],t)

model_data_t.iloc[:,6:8].head(10)
```

Out[102]:

Year	United States Nuclear (TWh)	Subtract Rolling Mean of United States Nuclear (TWh)
2000-01-01	753.89	NaN
2001-01-01	768.83	NaN
2002-01-01	780.06	NaN
2003-01-01	763.73	NaN
2004-01-01	788.53	NaN
2005-01-01	781.99	NaN
2006-01-01	787.22	NaN
2007-01-01	806.42	27.59
2008-01-01	806.21	20.84
2009-01-01	798.85	9.72

Before I move forward, I want to make sure I can transform this data back after modeling.

In [103]:
```
test = model_data_t.iloc[:,6:8].copy()

test['Results'] = inv_subtract_roll_mean(test, n=8)
test
```

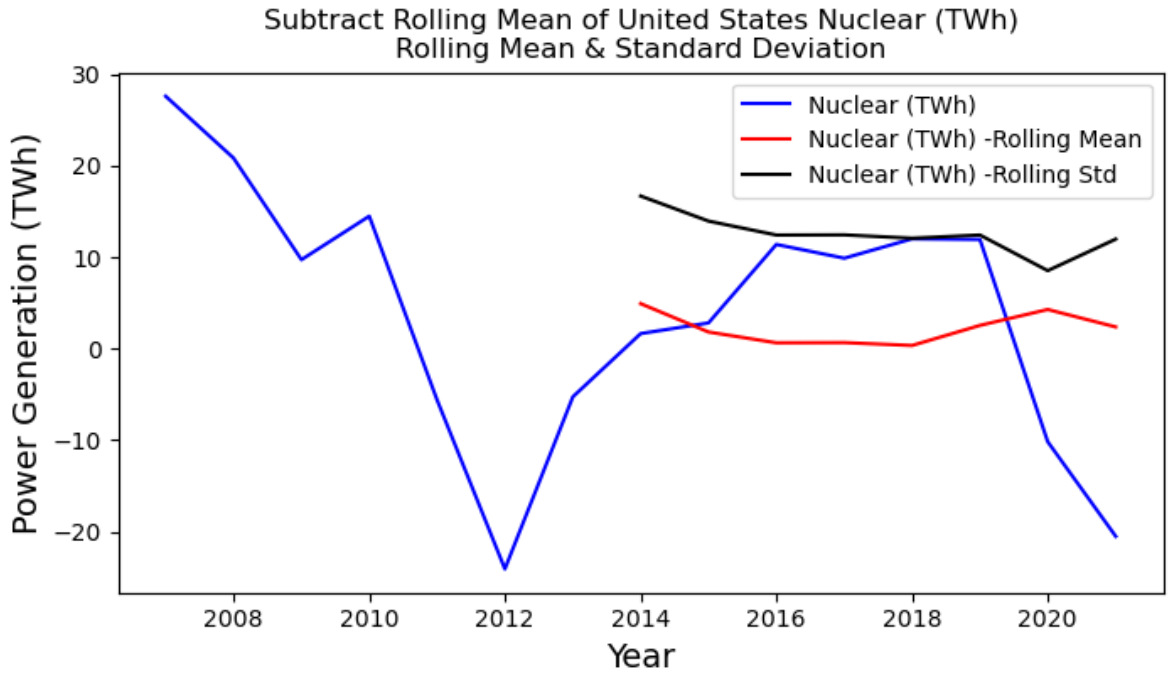
Out[103]:

Year	United States Nuclear (TWh)	Subtract Rolling Mean of United States Nuclear (TWh)	Results
2000-01-01	753.89	NaN	753.89
2001-01-01	768.83	NaN	768.83
2002-01-01	780.06	NaN	780.06
2003-01-01	763.73	NaN	763.73
2004-01-01	788.53	NaN	788.53
2005-01-01	781.99	NaN	781.99
2006-01-01	787.22	NaN	787.22
2007-01-01	806.42	27.59	806.42
2008-01-01	806.21	20.84	806.21
2009-01-01	798.85	9.72	798.85
2010-01-01	806.97	14.48	806.97
2011-01-01	790.20	-5.60	790.20
2012-01-01	769.33	-24.07	769.33
2013-01-01	789.02	-5.26	789.02
2014-01-01	797.17	1.65	797.17
2015-01-01	797.18	2.81	797.18
2016-01-01	805.69	11.39	805.69
2017-01-01	804.95	9.89	804.95
2018-01-01	807.08	12.00	807.08
2019-01-01	809.41	11.93	809.41
2020-01-01	789.88	-10.17	789.88
2021-01-01	778.19	-20.50	778.19

Transferring back is a possibility. Therefore, I can move forward. I'll again check for stationarity.

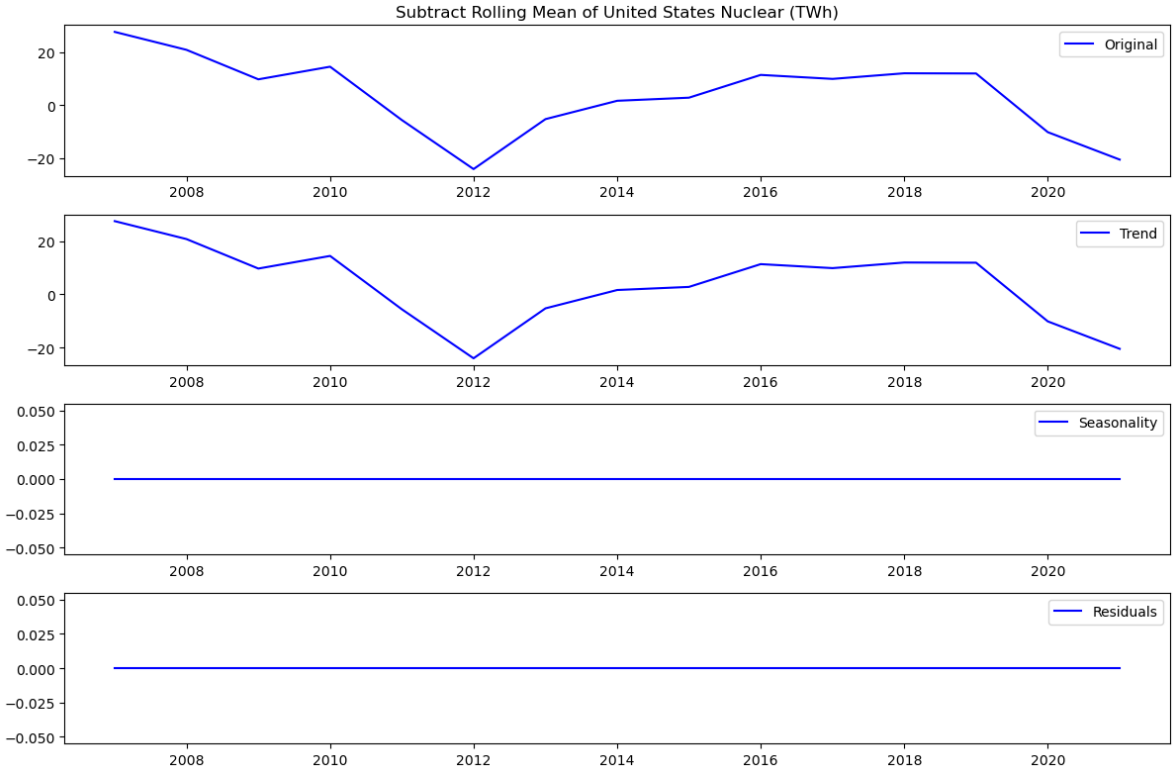
In [104]:
```
stationarity_check(model_data_t.iloc[:,7:8], s=39)
```

```
Subtract Rolling Mean of United States Nuclear (TWh)
Results of Dickey-Fuller test:

Test Statistic                -3.05
p-value                        0.03
#Lags Used                     5.00
Number of Observations Used    9.00
Critical Value (1%)           -4.47
Critical Value (5%)           -3.29
Critical Value (10%)          -2.77
dtype: float64
```



My p-value is less than .05. Stationarity achieved. I'll look at the decomposition graph again.

In [105]:
```
decomp_graph(model_data_t.iloc[:,7:8].dropna())
```

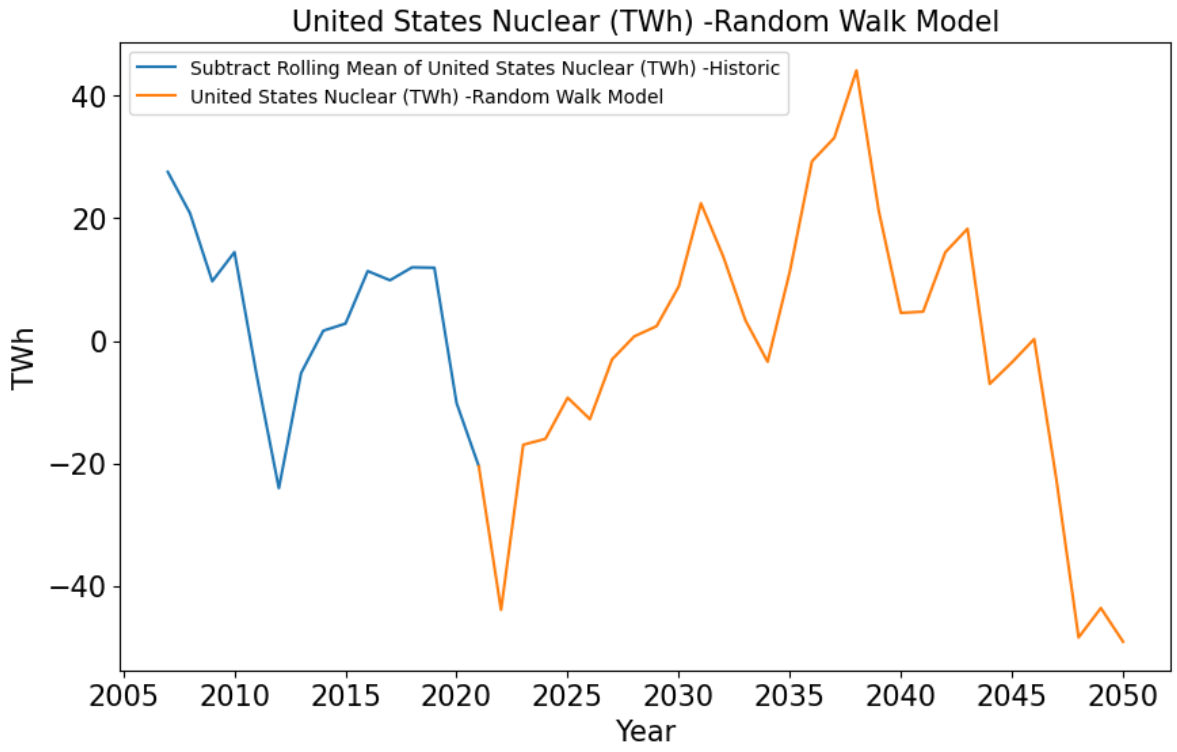


Looks like there's still a trend line, but it does pass stationarity, so I'm moving forward to the baseline random walk model.

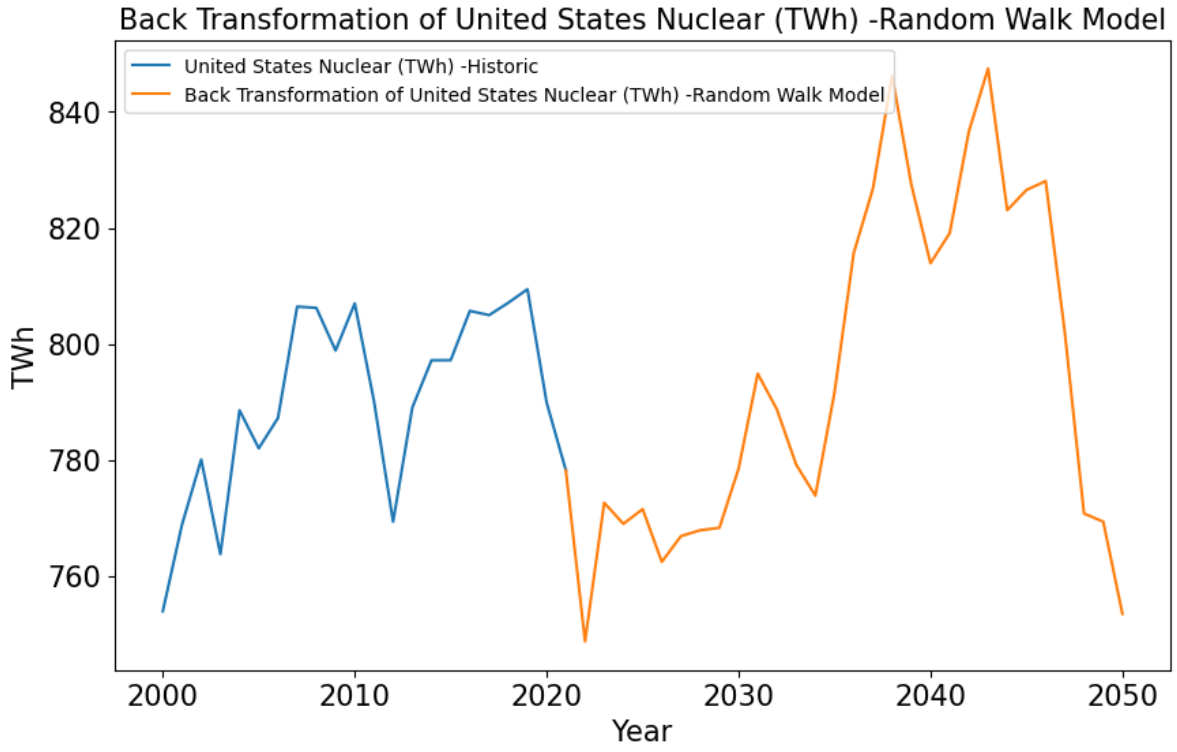
7.4.4 Random Walk Model

In [106]:
```
pd.set_option('display.max_rows',None)
y_hat_proj = random_walk(model_data_t.iloc[:,7:8])
y_hat_proj.columns = [model_data.columns[3]+' -Random Walk Model']

# Plot the transformed Random Walk
pred_graph(model_data_t.iloc[:,7:8], y_hat_proj.iloc[:,0:1])
```

```
Random Walk with Standard Deviation of Transformed Data set: 14.08
```



In [107]:
```
# Prepare the DataFrame for Back Transformation
ranplot = model_data_t.iloc[:,6:8].copy()
ranplot.rename(columns={str(ranplot.columns[1]): str(y_hat_proj.columns[0])},
               inplace=True)

ranplot = pd.concat([ranplot,y_hat_proj],axis=0)

# Perform Back Transformation
y_hat_proj = inv_subtract_roll_mean(ranplot, n=8)

#Plot the new graph
pred_graph(model_data_t.iloc[:,6:7], y_hat_proj.iloc[22:,0:1])
```

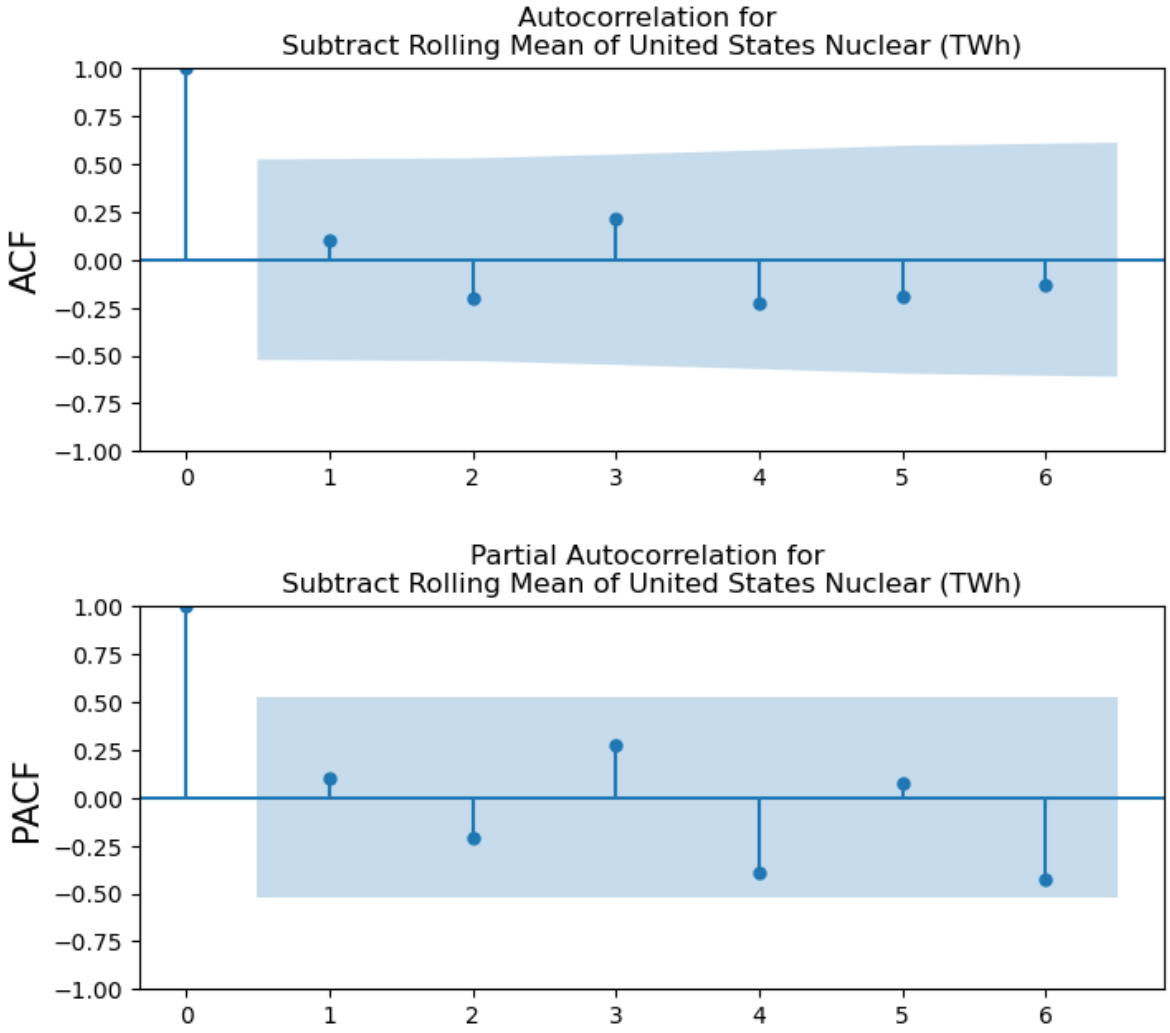


This model is as unreliable as it's name would imply, random walk. Again, of the transformed data oscillates about the zero x-axis, then the model is going to unrealistic when transformed back.

Now I'll move forward with an ARMA model. First, I need to evaluate the ACF and PACF plots to find the best Autoregressive and Moving Average terms.

7.4.5 Evaluating Initial AR and MA Values with ACF and PACF Plots

In [108]:
```
plotacf(model_data_t.iloc[:,7:8], lags = 6)
plotpacf(model_data_t.iloc[:,7:8], lags = 6)
```

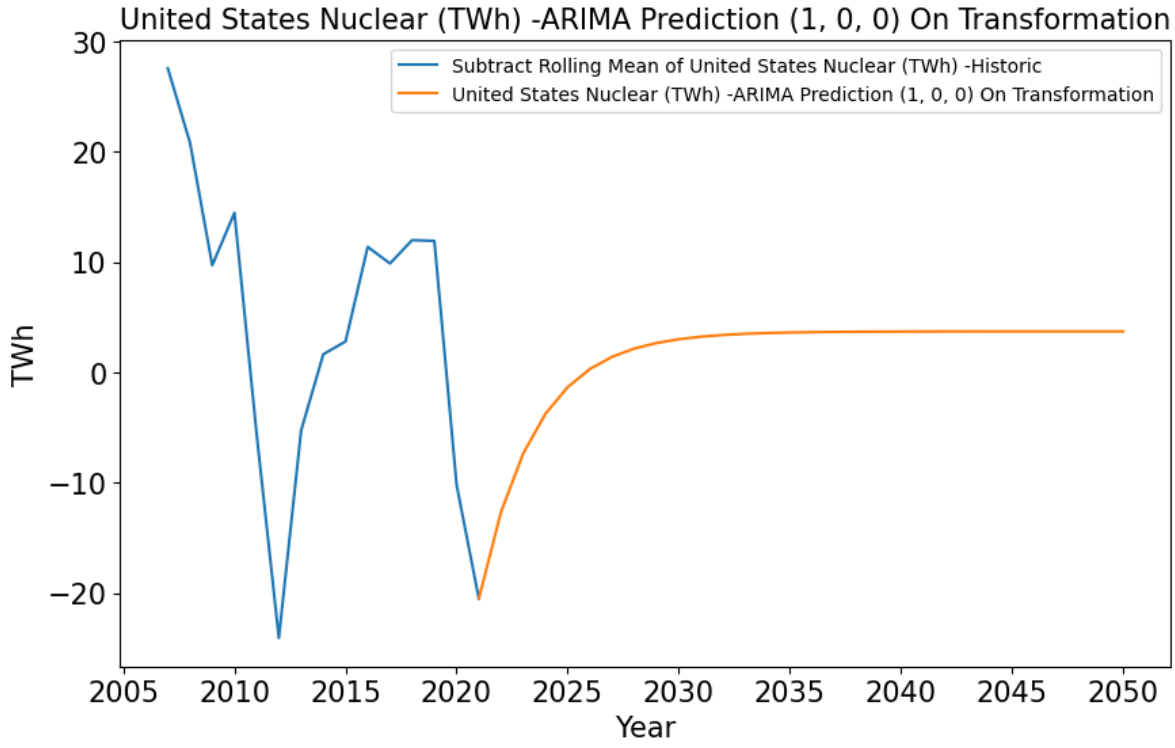


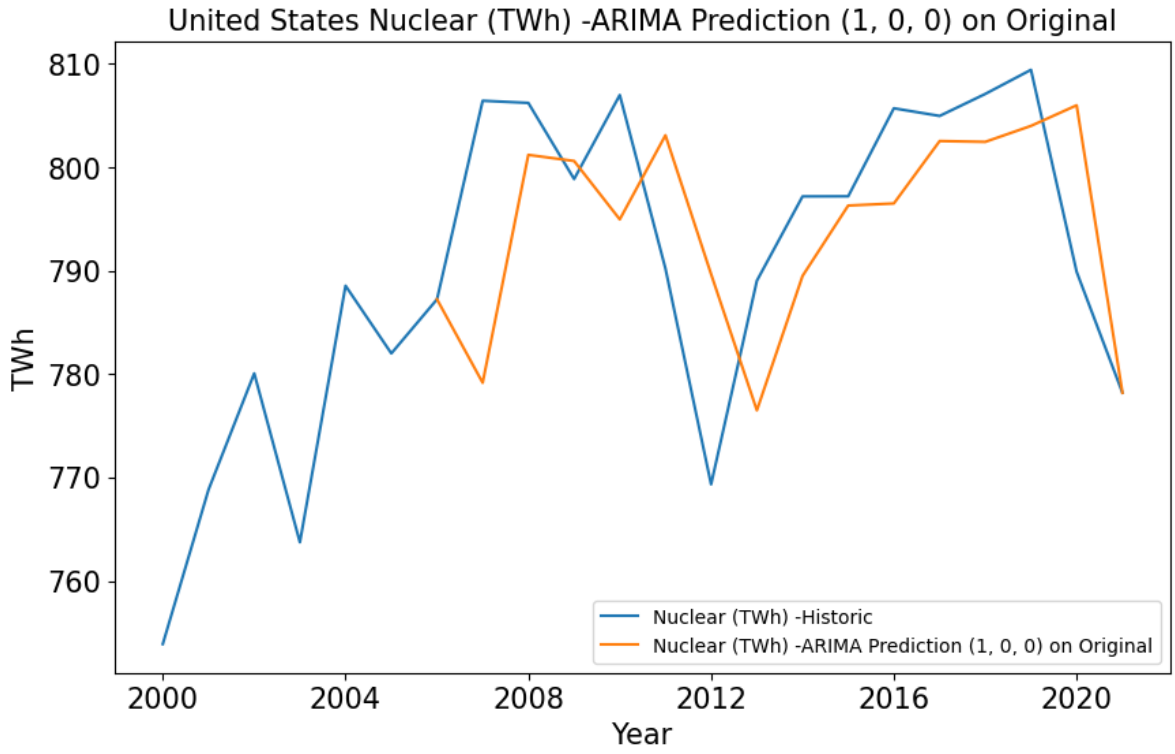
Neither ACF or PACF breaches at all. PACF gets close at 4 and 6. I'll try (1,0,0), (4,0,0), and (6,0,0) for my initial ARMA models.

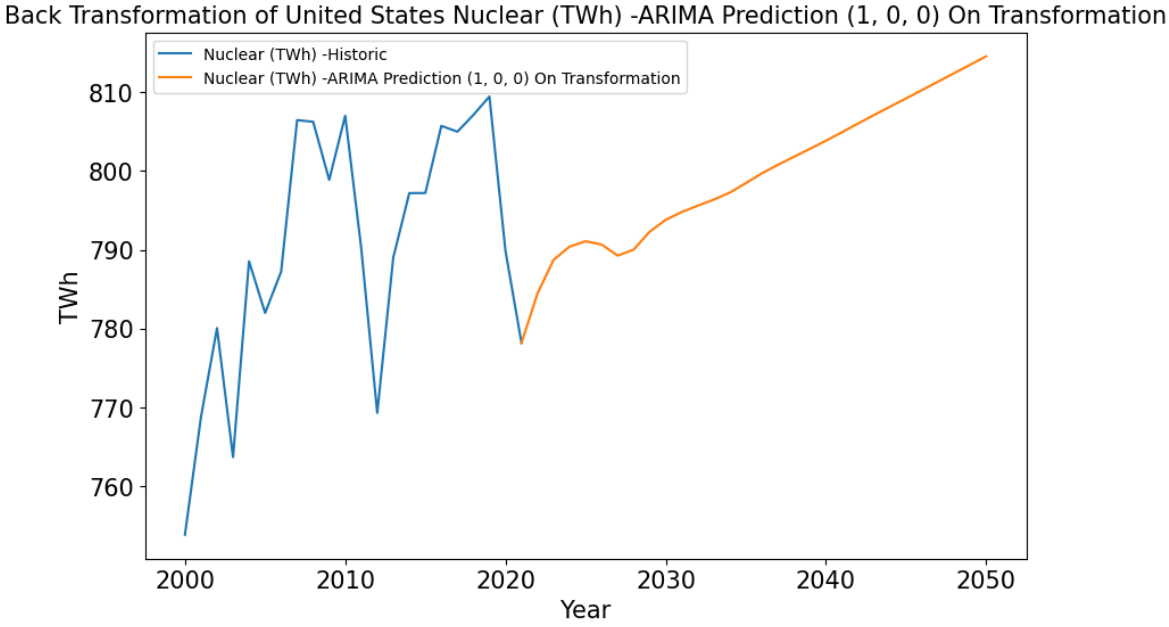
7.4.6 ARMA Model

In [109]:
```
order = (1,0,0)
model_summary(model_data_t.iloc[:,6:8], order, 22, 50, 14,
          tran = 'inv_subtract_roll_mean', n=8)
```

```
ARIMA: (1, 0, 0), RMSE=10.67, AIC=122.01
**********************************************************************
**
United States Nuclear (TWh) model results.

                                               SARIMAX Results
==========================================================================
==================================
Dep. Variable:     Subtract Rolling Mean of United States Nuclear (TWh)   No.
Observations:                   22
Model:                                             ARIMA(1, 0, 0)   Log
Likelihood                -58.007
Date:                                           Sat, 29 Apr 2023   AIC
122.013
Time:                                                  06:18:56   BIC
125.286
Sample:                                              01-01-2000   HQI
C                          122.784
                                                   - 01-01-2021
Covariance Type:                                             opg
==========================================================================
=
                 coef    std err          z      P>|z|      [0.025      0.97
5]
--------------------------------------------------------------------------
-
const          3.7276      7.731      0.482      0.630     -11.425      18.88
1
ar.L1          0.6757      0.261      2.585      0.010       0.163       1.18
8
sigma2       128.4692     62.556      2.054      0.040       5.861     251.07
7
==========================================================================
======
Ljung-Box (L1) (Q):                   0.65   Jarque-Bera (JB):
1.74
Prob(Q):                              0.42   Prob(JB):
0.42
Heteroskedasticity (H):                inf   Skew:
-0.68
Prob(H) (two-sided):                  0.00   Kurtosis:
3.23
==========================================================================
======

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (comple
x-step).
```


United States Nuclear (TWh) -ARIMA Prediction (1, 0, 0) On Transformation
Subtract Rolling Mean of United States Nuclear (TWh) -Historic
United States Nuclear (TWh) -ARIMA Prediction (1, 0, 0) On Transformation
TWh
Year
30
20
10
0
−10
−20
2005
2010
2015
2020
2025
2030
2035
2040
2045
2050


United States Nuclear (TWh) -ARIMA Prediction (1, 0, 0) on Original
Nuclear (TWh) -Historic
Nuclear (TWh) -ARIMA Prediction (1, 0, 0) on Original
TWh
Year
810
800
790
780
770
760
2000
2004
2008
2012
2016
2020



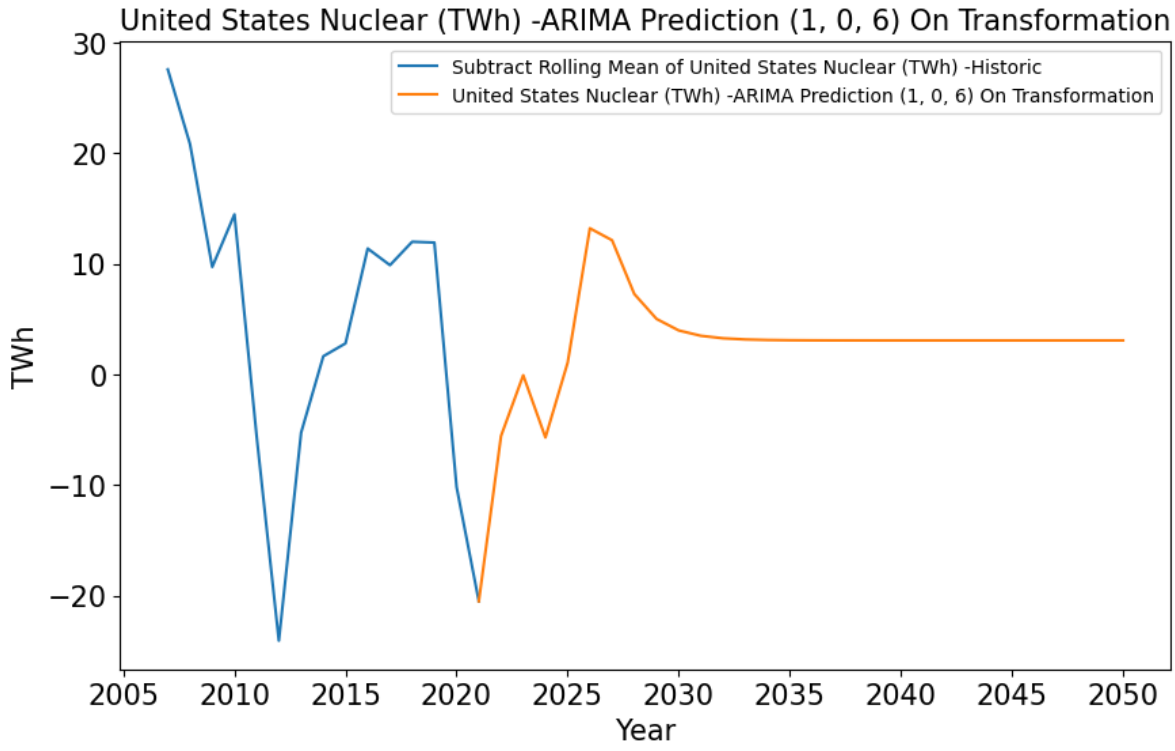
In [110]:
```
#order = (0,0,4)
# model_summary(model_data_t.iloc[:,6:8], order, 22, 50, 0,
#         tran = 'inv_subtract_roll_mean', n=8)
```

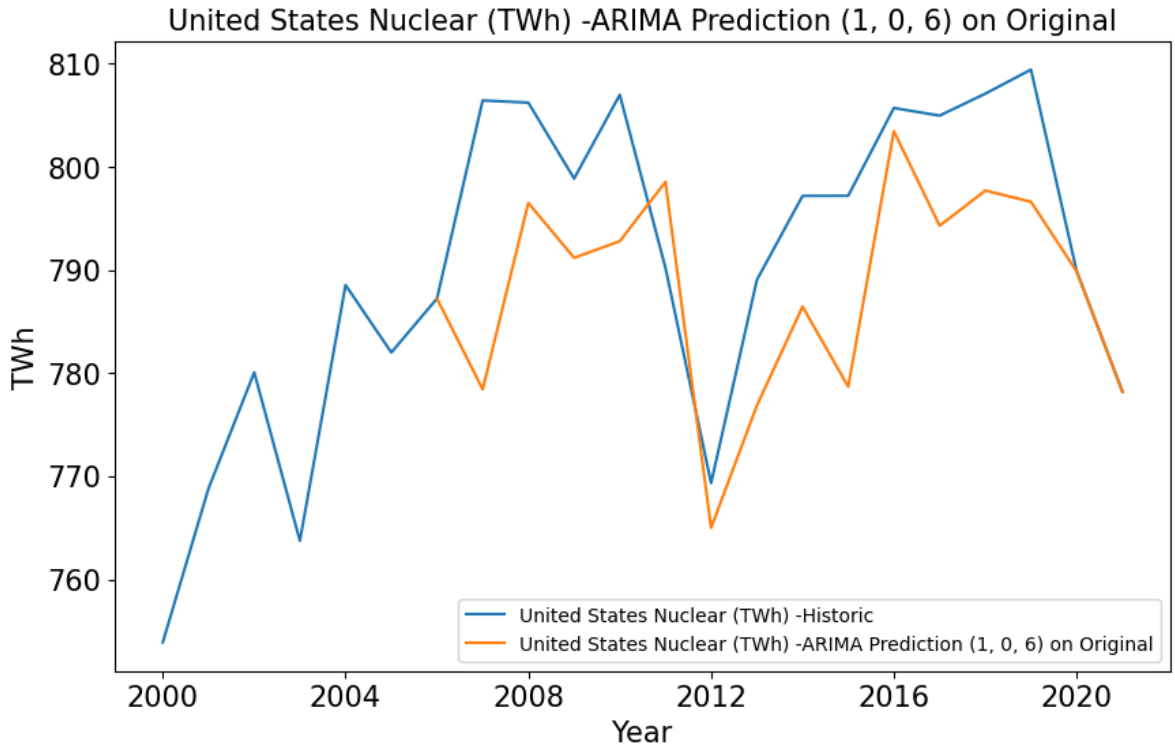
The combination above creates an error state. Sometimes this happens. All I can do is try a different model.

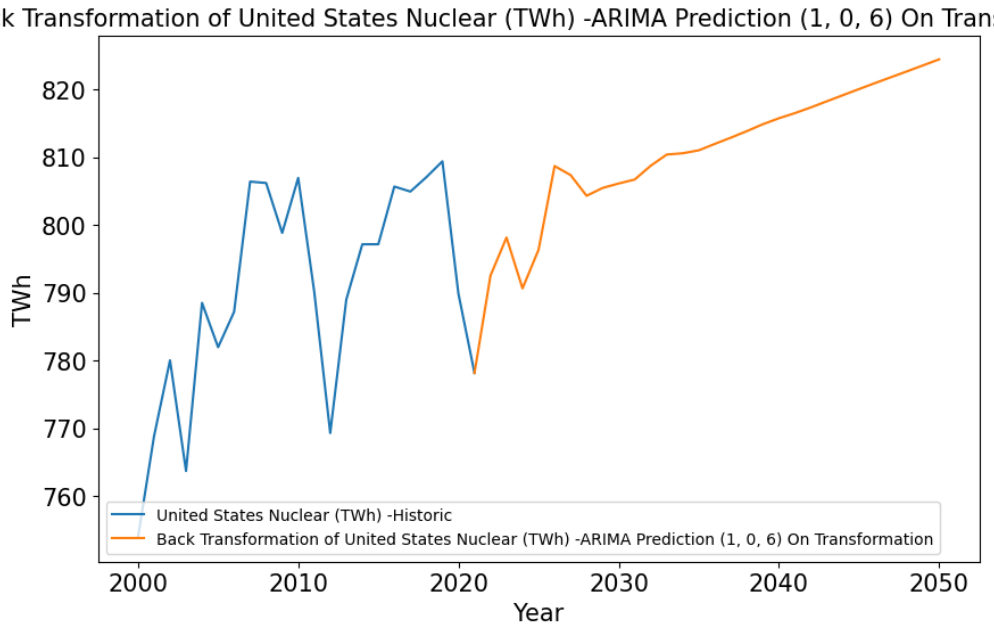
In [111]:
```
order = (1,0,6)
model_summary(model_data_t.iloc[:,6:8], order, 22, 50, 0,
          tran = 'inv_subtract_roll_mean', n=8)
```

```
ARIMA: (1, 0, 6), RMSE=10.09, AIC=121.58
**********************************************************************
**
United States Nuclear (TWh) model results.

                                               SARIMAX Results
==========================================================================
==================================
Dep. Variable:     Subtract Rolling Mean of United States Nuclear (TWh)   No.
Observations:                   22
Model:                                              ARIMA(1, 0, 6)   Log
Likelihood                -51.791
Date:                                           Sat, 29 Apr 2023   AIC
121.582
Time:                                                  06:18:57   BIC
131.402
Sample:                                                01-01-2000   HQI
C                          123.895
                                                     - 01-01-2021
Covariance Type:                                               opg
==========================================================================
=
                 coef    std err          z      P>|z|      [0.025      0.97
5]
--------------------------------------------------------------------------
-
const          3.0790      7.122      0.432      0.666     -10.880      17.03
8
ar.L1          0.4637      1.304      0.356      0.722      -2.093       3.02
0
ma.L1          0.8855    752.152      0.001      0.999   -1473.305    1475.07
6
ma.L2         -0.5436    743.277     -0.001      0.999   -1457.340    1456.25
3
ma.L3         -0.0020    566.821  -3.55e-06      1.000   -1110.951    1110.94
7
ma.L4          0.5208    610.705      0.001      0.999   -1196.439    1197.48
0
ma.L5         -0.8829    737.915     -0.001      0.999   -1447.171    1445.40
5
ma.L6         -0.9761    680.472     -0.001      0.999   -1334.677    1332.72
5
sigma2        29.6922   2.07e+04      0.001      0.999   -4.05e+04    4.05e+0
4
==========================================================================
======
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):
0.86
Prob(Q):                              0.97   Prob(JB):
0.65
Heteroskedasticity (H):                inf   Skew:
-0.24
Prob(H) (two-sided):                  0.00   Kurtosis:
3.84
==========================================================================
======
```

```
Warnings:
[1] Covariance matrix calculated using the outer product of gradients (comple
x-step).
```





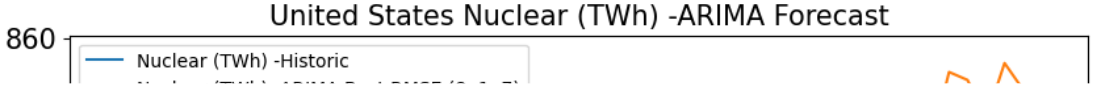


I cannot explain why the two models diverge one spot earlier than I expected. Otherwise, both of these models look convincing. They show a steady incline in Nuclear.

7.4.7 ARIMA Model and Grid Search

In [112]:
```
oil_rmse_cfg, oil_aic_cfg = arima_pdq(
            model_data_t.iloc[:,6:8], s=14, tran='inv_subtract_roll_mean', n=8
```

```
RMSE ARIMA: (0, 0, 0), RMSE= 13.96, AIC= 125.90, TWh-2050= 829.39
RMSE ARIMA: (0, 0, 1), RMSE= 9.20, AIC= 116.95, TWh-2050= 827.23
RMSE ARIMA: (0, 0, 2), RMSE= 9.00, AIC= 116.89, TWh-2050= 829.24
RMSE ARIMA: (0, 1, 6), RMSE= 8.97, AIC= 129.29, TWh-2050= 807.03
RMSE ARIMA: (1, 0, 1), RMSE= 8.82, AIC= 117.70, TWh-2050= 825.18
RMSE ARIMA: (1, 0, 2), RMSE= 8.75, AIC= 117.80, TWh-2050= 828.47
AIC  ARIMA: (4, 0, 0), RMSE= 15.15, AIC= 114.71, TWh-2050= 821.42
AIC  ARIMA: (4, 0, 2), RMSE= 17.33, AIC= 113.50, TWh-2050= 826.75
AIC  ARIMA: (6, 0, 0), RMSE= 15.62, AIC= 111.79, TWh-2050= 823.61
RMSE ARIMA: (6, 1, 5), RMSE= 8.46, AIC= 126.80, TWh-2050= 780.52
RMSE ARIMA: (6, 1, 6), RMSE= 8.32, AIC= 127.91, TWh-2050= 786.19
RMSE ARIMA: (8, 1, 6), RMSE= 8.20, AIC= 130.11, TWh-2050= 800.95
RMSE ARIMA: (8, 1, 7), RMSE= 8.18, AIC= 131.61, TWh-2050= 832.07
Best RMSE ARIMA: (8, 1, 7) RMSE= 8.18 AIC= 131.61, TWh-2050= 832.07
Best AIC  ARIMA: (6, 0, 0) RMSE= 15.62 AIC= 111.79, TWh-2050= 823.61
Graph_formatting produced error at (8, 1, 7) or (6, 0, 0)
```

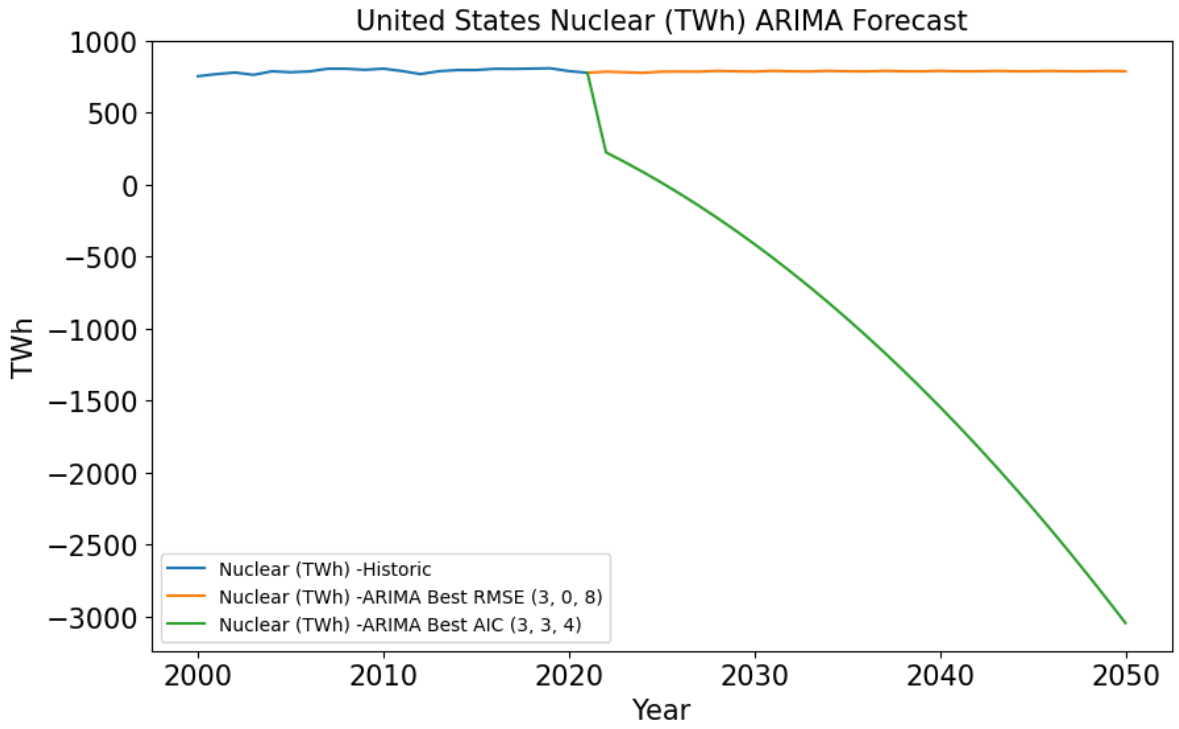


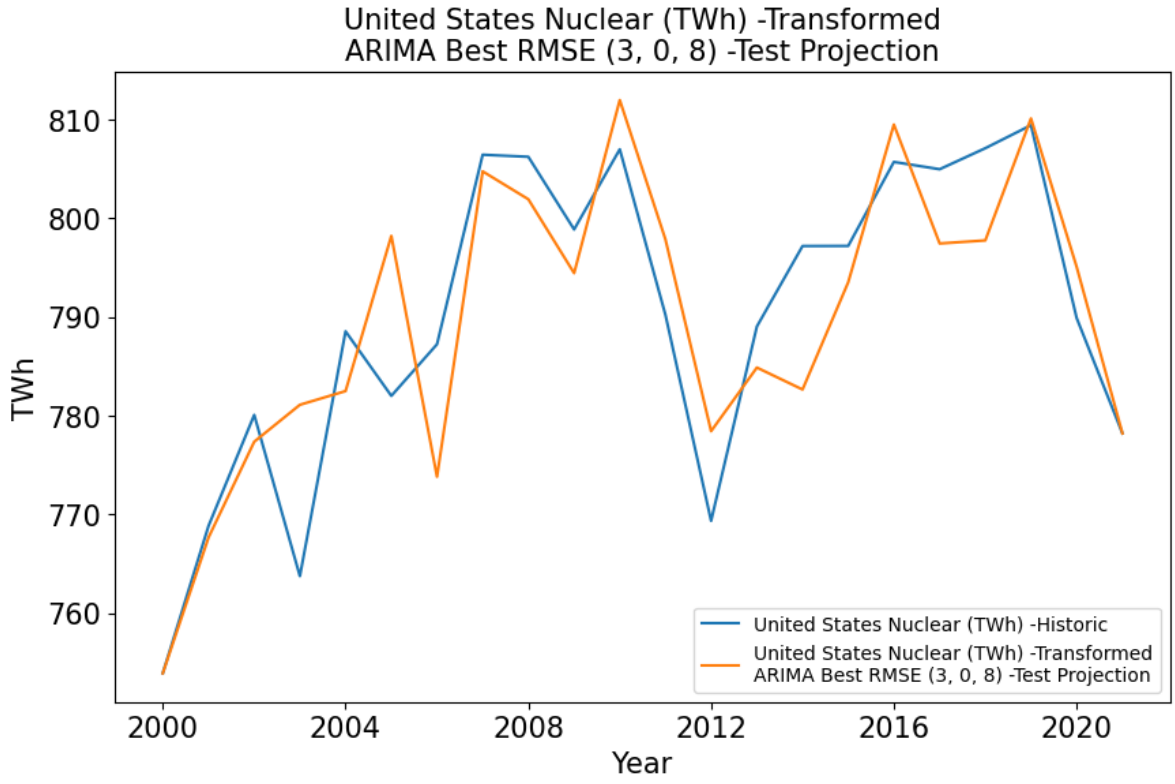
This model is reasonable, but it looks as though it added a bit of seasonality, which is curious. More research needs to be done into why nuclear tanked in 2012 and 2020/2021. It likely has something to do with the fact that nuclear is the most expensive kind of power generation.

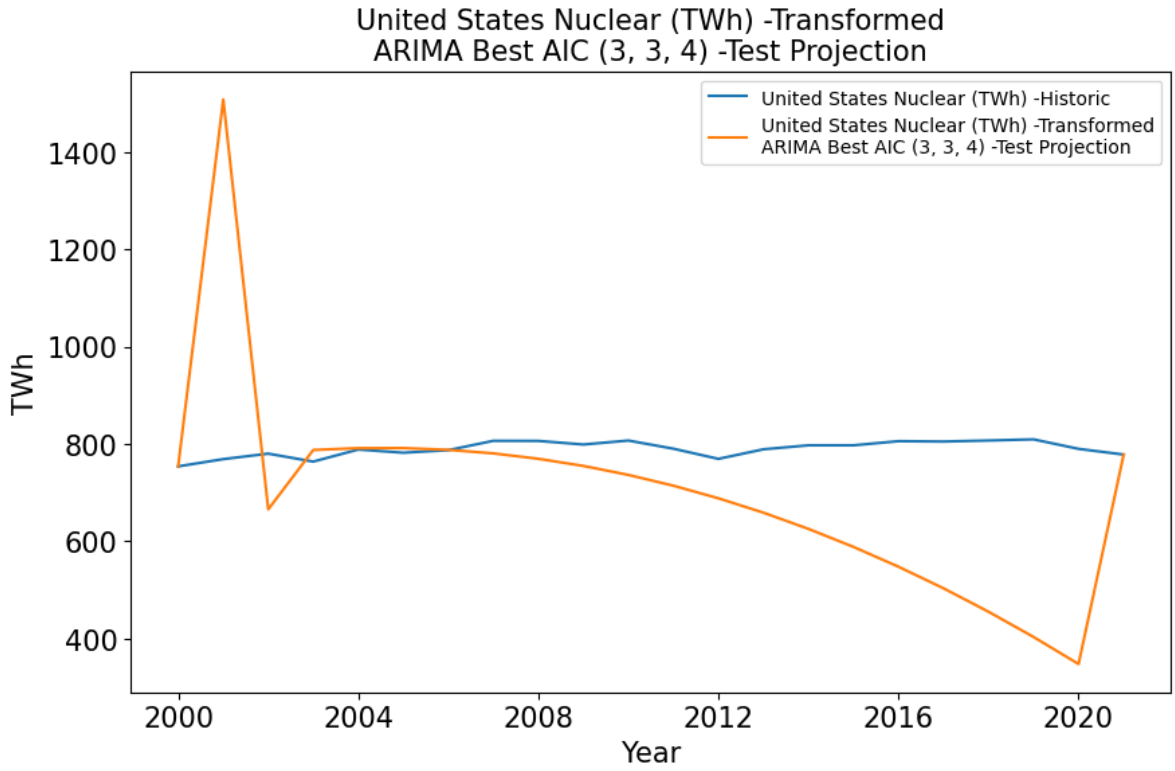
7.4.8 ARIMA Without Transformation

In [113]:
```
oil_rmse_cfg, oil_aic_cfg = arima_pdq_no_tran(
                                        model_data.iloc[:,3:4], 22, 50, s=14)
```

```
United States Nuclear (TWh) Grid Search:
RMSE ARIMA: (0, 0, 0), RMSE= 13.27, AIC= 187.18, TWh-2050= 790.04
RMSE ARIMA: (0, 0, 1), RMSE= 10.57, AIC= 179.14, TWh-2050= 789.38
RMSE ARIMA: (0, 0, 3), RMSE= 9.47, AIC= 178.85, TWh-2050= 787.49
RMSE ARIMA: (0, 0, 4), RMSE= 9.42, AIC= 180.68, TWh-2050= 787.27
RMSE ARIMA: (0, 0, 7), RMSE= 8.91, AIC= 184.07, TWh-2050= 789.05
AIC  ARIMA: (0, 1, 0), RMSE= 12.48, AIC= 169.40, TWh-2050= 778.19
AIC  ARIMA: (0, 2, 1), RMSE= 78.36, AIC= 167.24, TWh-2050= 764.11
AIC  ARIMA: (0, 3, 2), RMSE= 160.37, AIC= 166.55, TWh-2050= 372.29
RMSE ARIMA: (1, 0, 7), RMSE= 8.85, AIC= 185.18, TWh-2050= 789.78
AIC  ARIMA: (2, 3, 1), RMSE= 160.35, AIC= 165.78, TWh-2050= 49.40
RMSE ARIMA: (3, 0, 5), RMSE= 8.84, AIC= 186.27, TWh-2050= 789.69
RMSE ARIMA: (3, 0, 7), RMSE= 8.39, AIC= 187.87, TWh-2050= 784.92
RMSE ARIMA: (3, 0, 8), RMSE= 8.05, AIC= 186.74, TWh-2050= 788.50
AIC  ARIMA: (3, 3, 4), RMSE= 244.28, AIC= 16.00, TWh-2050= -3043.86
Best RMSE ARIMA: (3, 0, 8) RMSE= 8.05 AIC= 186.74, TWh-2050= 788.50
Best AIC  ARIMA: (3, 3, 4) RMSE= 244.28 AIC= 16.00, TWh-2050= -3043.86
```




United States Nuclear (TWh) -Transformed
ARIMA Best RMSE (3, 0, 8) -Test Projection
TWh
Year
United States Nuclear (TWh) -Historic
United States Nuclear (TWh) -Transformed
ARIMA Best RMSE (3, 0, 8) -Test Projection


United States Nuclear (TWh) -Transformed
ARIMA Best AIC (3, 3, 4) -Test Projection
TWh
Year
United States Nuclear (TWh) -Historic
United States Nuclear (TWh) -Transformed
ARIMA Best AIC (3, 3, 4) -Test Projection

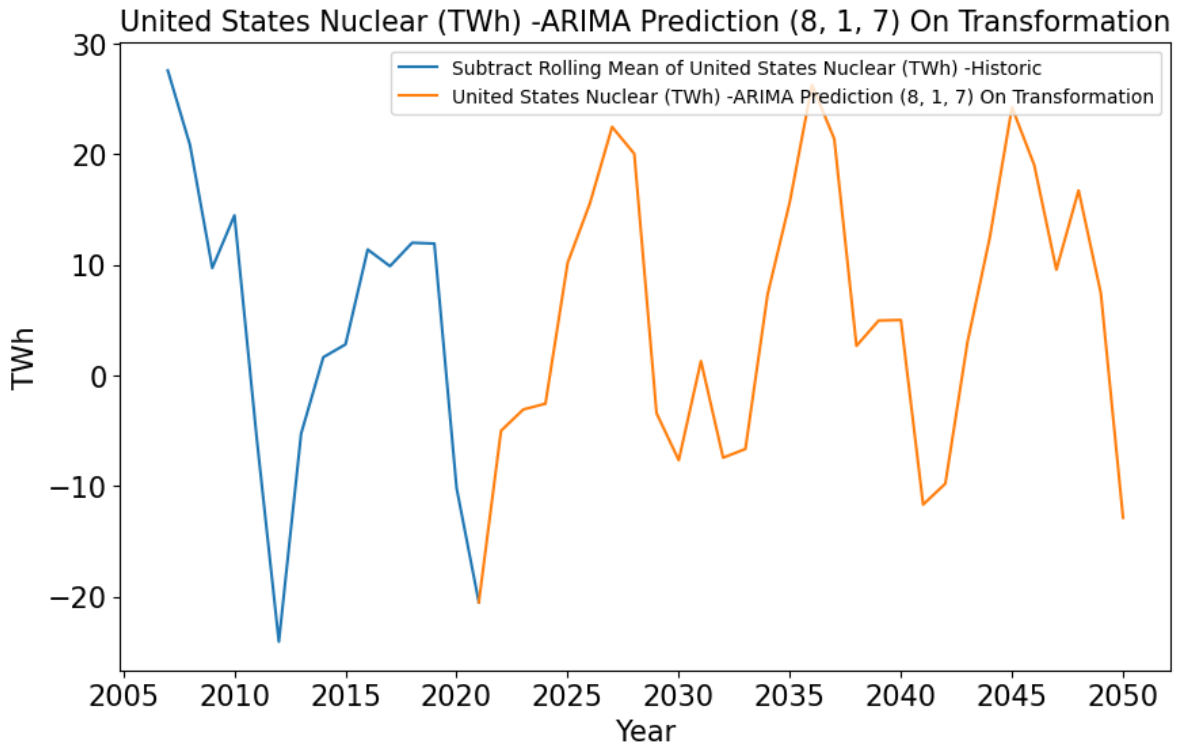
7.4.9 Model Selection

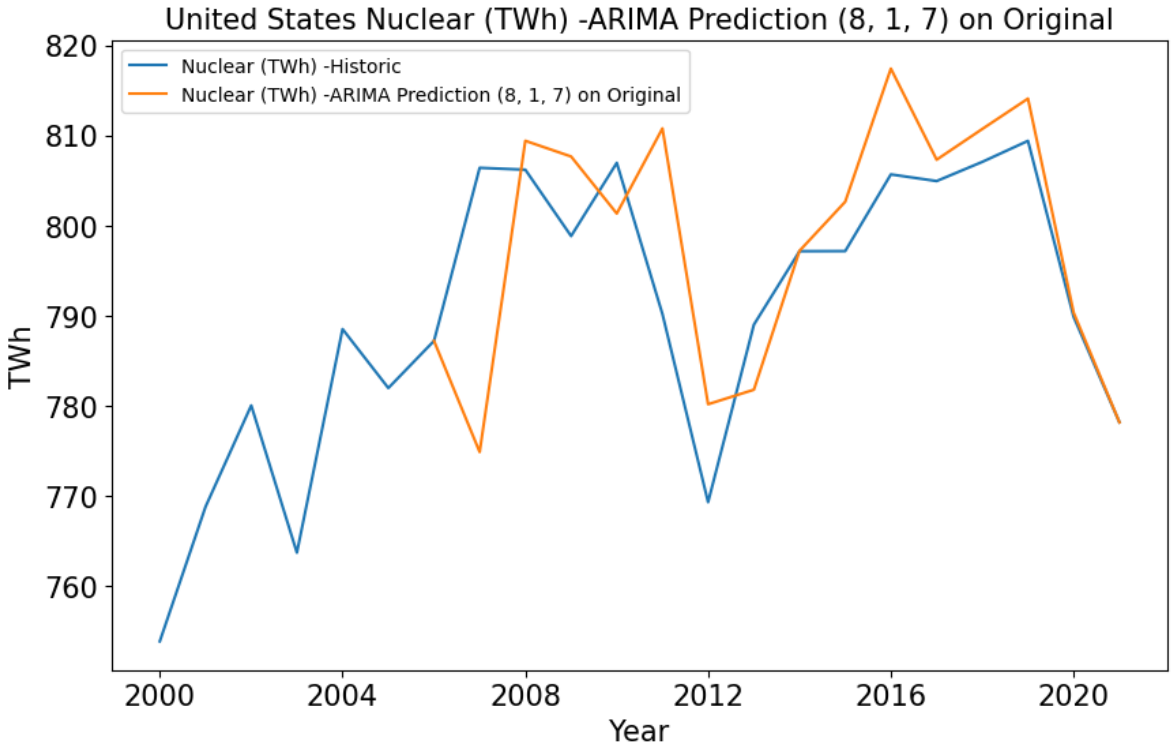
In [114]:
```
#Deleteme
stored_nuc = selected_models.copy()
```

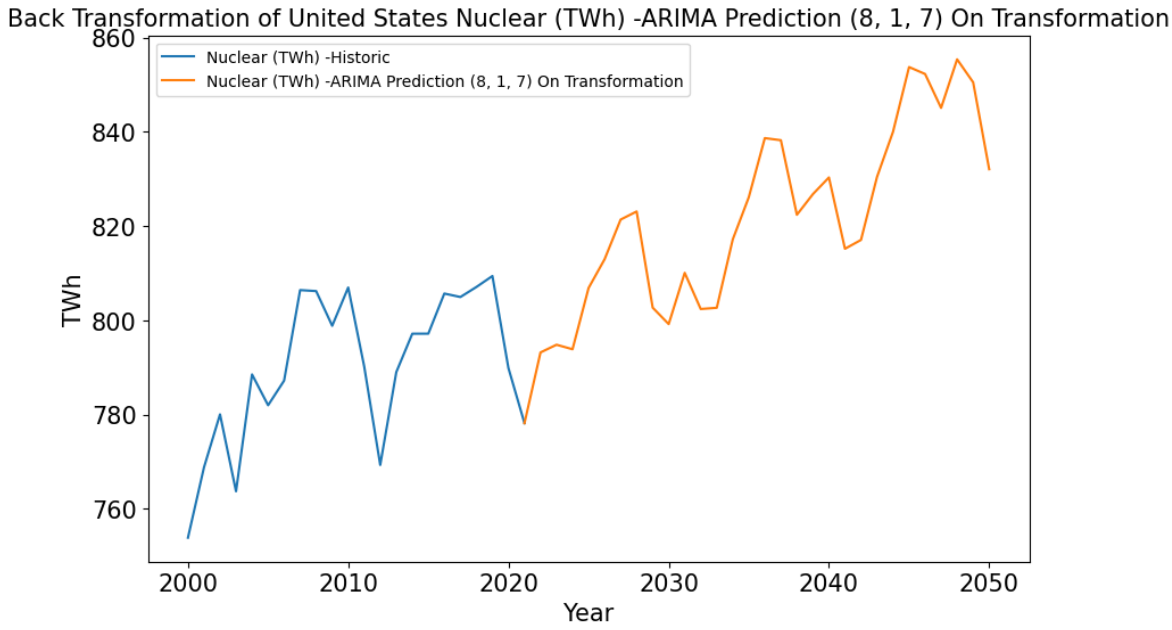
In [115]:
```
selected_models = stored_nuc.copy() #deleteme
df_nuclear = model_data_t.iloc[:,6:8]

#                        Model,       df_o,       pdq,
selected_models.append(['ARIMA_tran', df_nuclear, (8,1,7),
                        'inv_subtract_roll_mean', 8, None])

m = 3
model_summary(selected_models[m][1], selected_models[m][2], 22, 50, 14,
              selected_models[m][3], selected_models[m][4])
```

```
ARIMA: (8, 1, 7), RMSE=8.18, AIC=131.61
**********************************************************************
**
United States Nuclear (TWh) model results.

                                                SARIMAX Results
==========================================================================
==================================
Dep. Variable:     Subtract Rolling Mean of United States Nuclear (TWh)   No.
Observations:                   22
Model:                                            ARIMA(8, 1, 7)   Log
Likelihood                -49.805
Date:                                          Sat, 29 Apr 2023   AIC
131.611
Time:                                                 06:20:14   BIC
148.323
Sample:                                               01-01-2000   HQI
C                          135.238
                                                    - 01-01-2021
Covariance Type:                                              opg
==========================================================================
=
                 coef    std err          z      P>|z|      [0.025      0.97
5]
--------------------------------------------------------------------------
-
ar.L1          0.6116      9.083      0.067      0.946     -17.190      18.41
3
ar.L2         -0.4274     26.928     -0.016      0.987     -53.205      52.35
0
ar.L3          0.2743     19.039      0.014      0.989     -37.042      37.59
1
ar.L4         -0.0662     44.365     -0.001      0.999     -87.020      86.88
7
ar.L5         -0.3445     29.066     -0.012      0.991     -57.312      56.62
3
ar.L6          0.2546     30.609      0.008      0.993     -59.738      60.24
7
ar.L7         -0.7205     25.292     -0.028      0.977     -50.291      48.85
0
ar.L8          0.8496     29.409      0.029      0.977     -56.791      58.49
0
ma.L1         -0.8896   1393.594     -0.001      0.999   -2732.283    2730.50
4
ma.L2          1.1299   1130.859      0.001      0.999   -2215.312    2217.57
2
ma.L3         -0.1450    526.792     -0.000      1.000   -1032.639    1032.34
9
ma.L4         -0.2493    309.358     -0.001      0.999    -606.580     606.08
1
ma.L5          1.1876    788.785      0.002      0.999   -1544.803    1547.17
8
ma.L6         -0.8440   1109.984     -0.001      0.999   -2176.372    2174.68
4
ma.L7          0.8829    341.764      0.003      0.998    -668.963     670.72
9
```

```
sigma2         2.6268    544.573      0.005      0.996    -1064.717    1069.97
1
==========================================================================
======
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):
1.27
Prob(Q):                              1.00   Prob(JB):
0.53
Heteroskedasticity (H):            8064.30   Skew:
-0.60
Prob(H) (two-sided):                  0.00   Kurtosis:
3.09
==========================================================================
======

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (comple
x-step).
[2] Covariance matrix is singular or near-singular, with condition number 6.4
e+18. Standard errors may be unstable.
```







This model has the best RMSE and is reasonable. Though I don't think nuclear has such a set seasonal component, projects are that it will rise in the future.

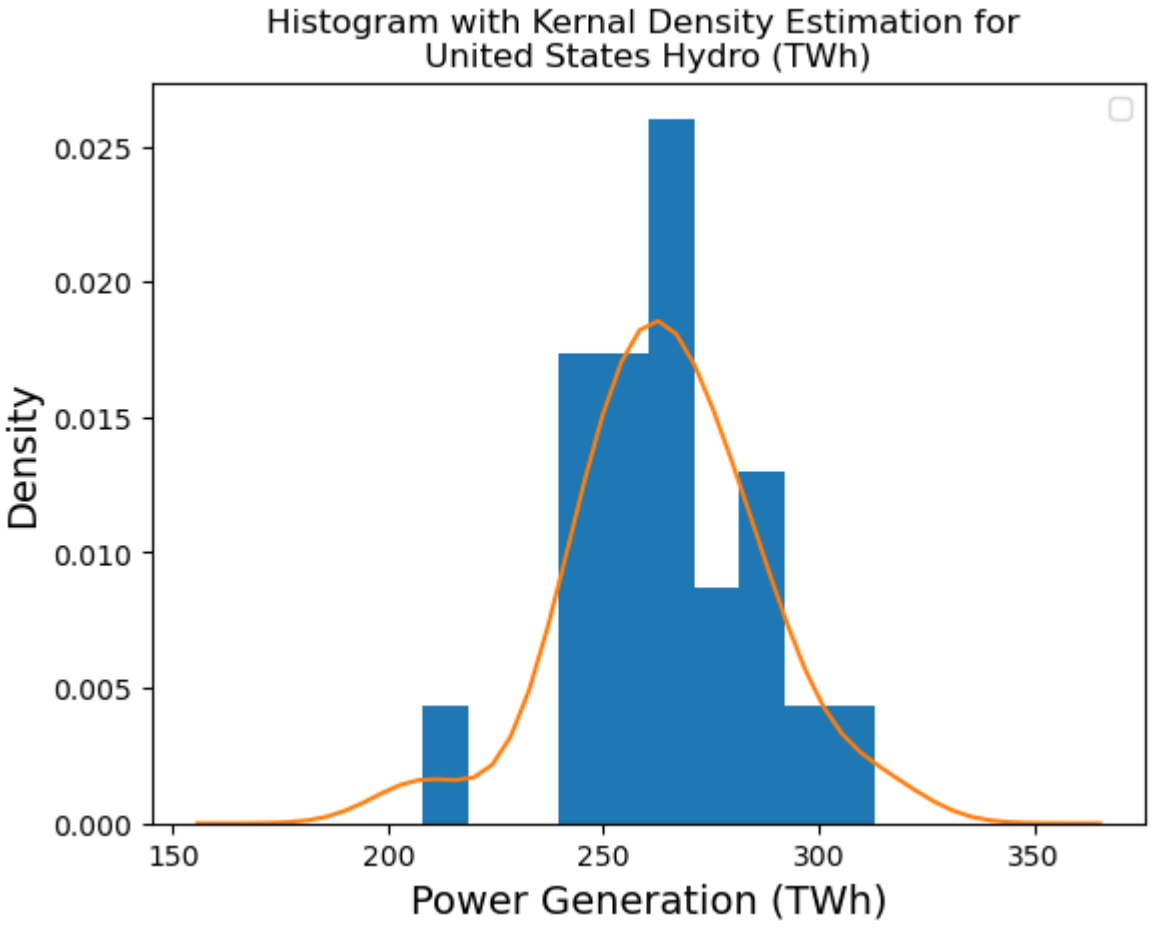
7.5 Hydro

7.5.1 Distribution Investigation

In [116]:
```
stored_model_data = model_data.copy()
stored_model_data_t = model_data_t.copy()
```

In [117]:
```
model_data = stored_model_data.copy()
model_data_t = stored_model_data_t.copy()
```

In [118]:
```
hist(model_data.iloc[:,4:5])
```



In [119]:
```
stats_block = s_block(model_data.iloc[:,4:5], stats_block)
stats_block
```

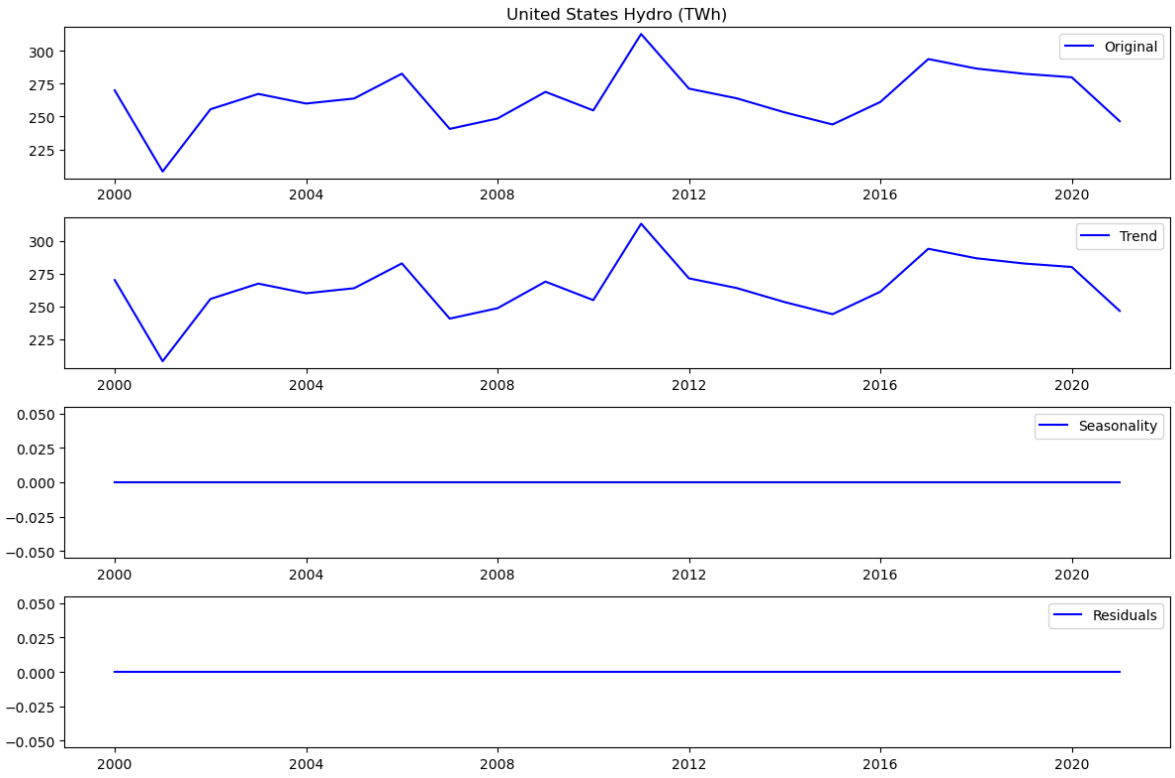
Out[119]:

	Dataset	Mean	Median	Standard Deviation	Skew	Fisher Kurtosis
0	United States Coal (TWh)	1,607.17	1,744.66	399.15	-0.68	-0.90
1	United States Gas (TWh)	1,060.39	1,000.70	325.82	0.29	-1.21
2	United States Oil (TWh)	67.20	45.97	36.59	0.89	-0.92
3	United States Nuclear (TWh)	790.04	790.04	15.55	-0.64	-0.53
4	United States Hydro (TWh)	264.36	263.82	21.08	-0.22	1.03

This is very close to normal distribution, with taller middle and thinner tails.

7.5.2 Decomposing The Data

In [120]:
```
decomp_graph(model_data.iloc[:,4:5])
```

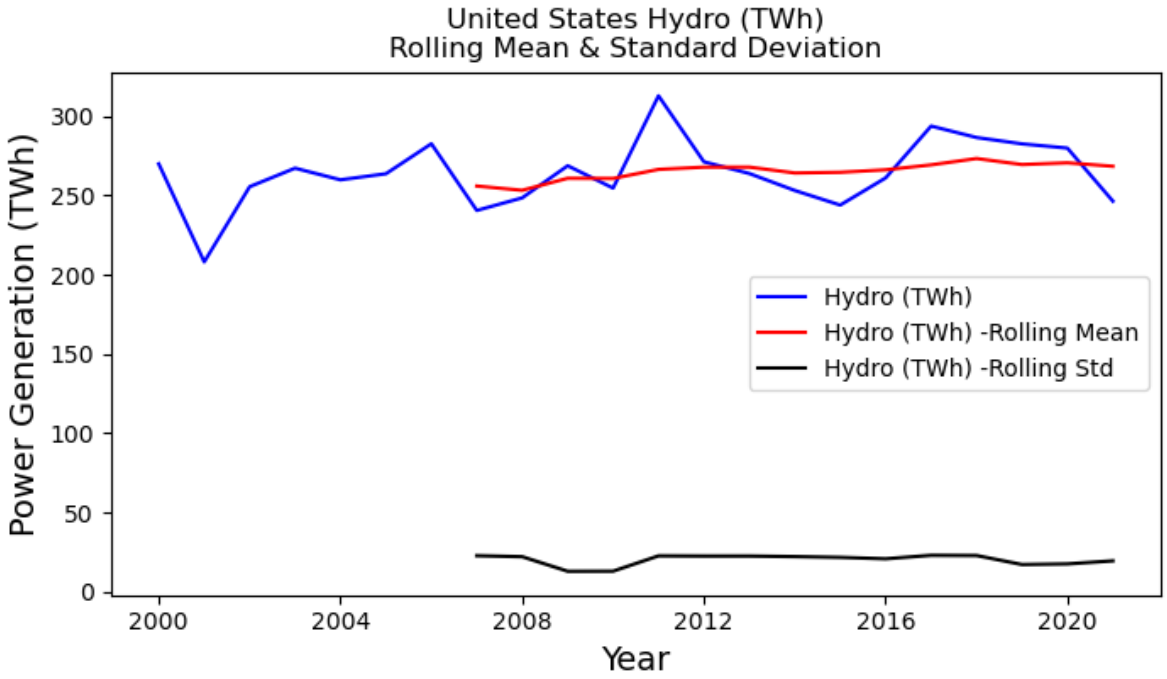


No seasonality or residuals. I'll check for stationarity.

7.5.3 Checking for Stationarity and Flattening

In [121]:
```
pd.set_option('display.max_rows', 30)
stationarity_check(model_data.iloc[:,4:5], s=14)
```

```
United States Hydro (TWh)
Results of Dickey-Fuller test:

Test Statistic                -2.62
p-value                        0.09
#Lags Used                     8.00
Number of Observations Used   13.00
Critical Value (1%)           -4.07
Critical Value (5%)           -3.13
Critical Value (10%)          -2.70
dtype: float64
```



This is close to stationary. After playing around with transformations, I found that if I subtracted the rolling average of four values, I can get it stationary.

In [122]:
```
stored = model_data_t.copy()
```

In [123]:
```
model_data_t = stored.copy()
t = subtract_roll_mean((model_data.iloc[:,4:5]), n=4)
model_data_t.insert(9,t.columns[0],t)

model_data_t.iloc[:,8:10].head(10)
```

Out[123]:

Year	United States Hydro (TWh)	Subtract Rolling Mean of United States Hydro (TWh)
2000-01-01	270.03	NaN
2001-01-01	208.14	NaN
2002-01-01	255.59	NaN
2003-01-01	267.27	17.01
2004-01-01	259.93	12.20
2005-01-01	263.76	2.12
2006-01-01	282.69	14.28
2007-01-01	240.61	-21.14
2008-01-01	248.54	-10.36
2009-01-01	268.82	8.66

Before I move forward, I want to make sure I can transform this data back after modeling.

In [124]:
```
test = model_data_t.iloc[:,8:10].copy()

test['Results'] = inv_subtract_roll_mean(test, n=4)
test
```

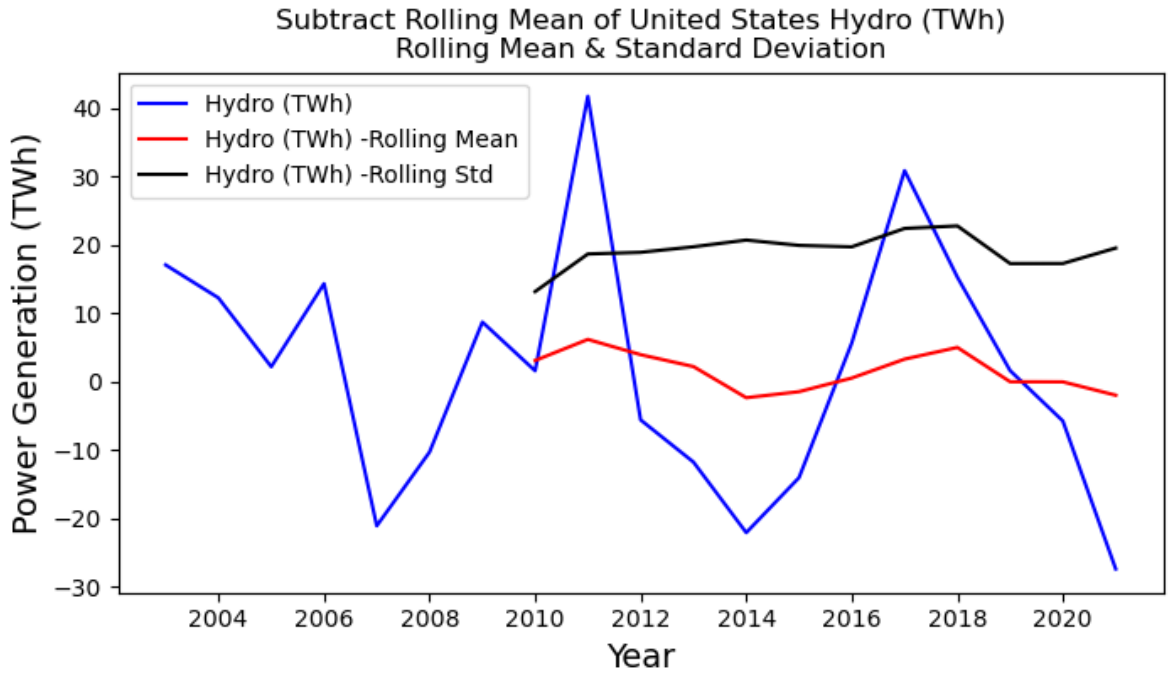
Out[124]:

Year	United States Hydro (TWh)	Subtract Rolling Mean of United States Hydro (TWh)	Results
2000-01-01	270.03	NaN	270.03
2001-01-01	208.14	NaN	208.14
2002-01-01	255.59	NaN	255.59
2003-01-01	267.27	17.01	267.27
2004-01-01	259.93	12.20	259.93
2005-01-01	263.76	2.12	263.76
2006-01-01	282.69	14.28	282.69
2007-01-01	240.61	-21.14	240.61
2008-01-01	248.54	-10.36	248.54
2009-01-01	268.82	8.66	268.82
2010-01-01	254.70	1.53	254.70
2011-01-01	312.93	41.68	312.93
2012-01-01	271.29	-5.64	271.29
2013-01-01	263.88	-11.82	263.88
2014-01-01	253.19	-22.13	253.19
2015-01-01	243.99	-14.10	243.99
2016-01-01	261.13	5.58	261.13
2017-01-01	293.84	30.80	293.84
2018-01-01	286.62	15.23	286.62
2019-01-01	282.61	1.56	282.61
2020-01-01	279.95	-5.81	279.95
2021-01-01	246.47	-27.44	246.47

Transferring back is a possibility. Therefore, I can move forward. I'll again check for stationarity.

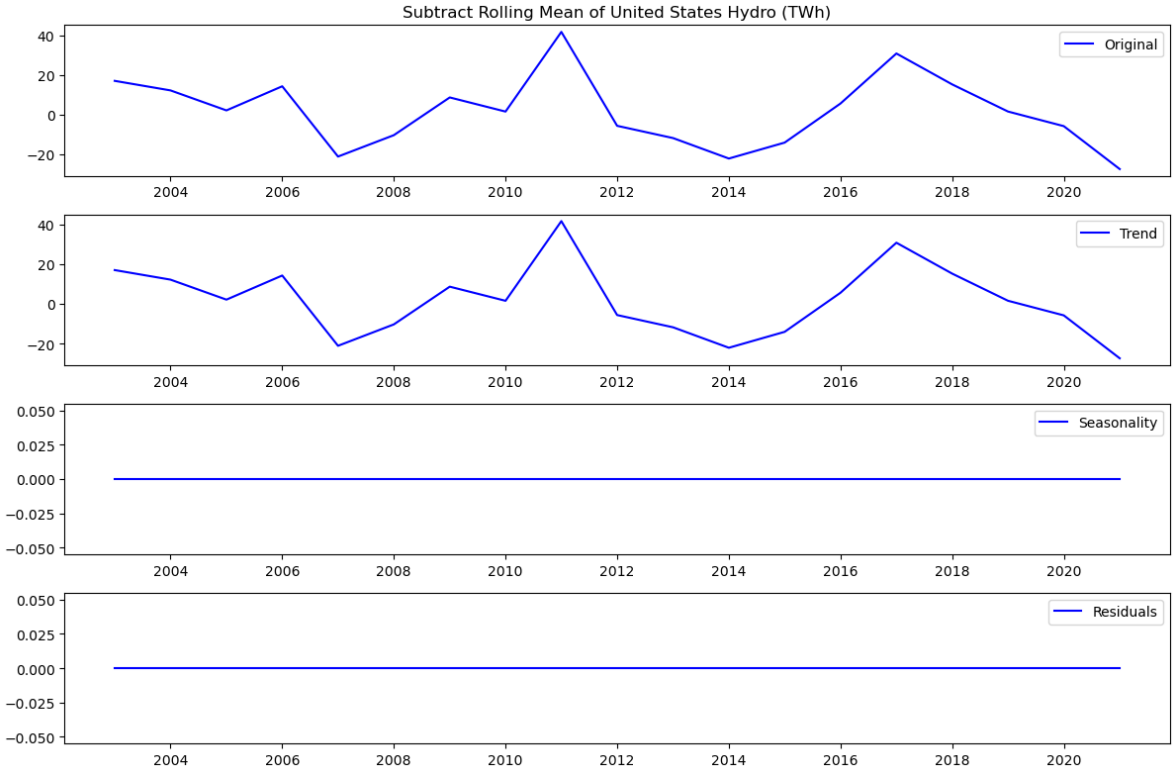
In [125]:
```
stationarity_check(model_data_t.iloc[:,9:10], s=39)
```

```
Subtract Rolling Mean of United States Hydro (TWh)
Results of Dickey-Fuller test:

Test Statistic                -7.83
p-value                        0.00
#Lags Used                     3.00
Number of Observations Used   15.00
Critical Value (1%)           -3.96
Critical Value (5%)           -3.08
Critical Value (10%)          -2.68
dtype: float64
```



My p-value is less than .05. Stationarity achieved. I'll look at the decomposition graph again.

In [126]:
```
decomp_graph(model_data_t.iloc[:,9:10].dropna())
```

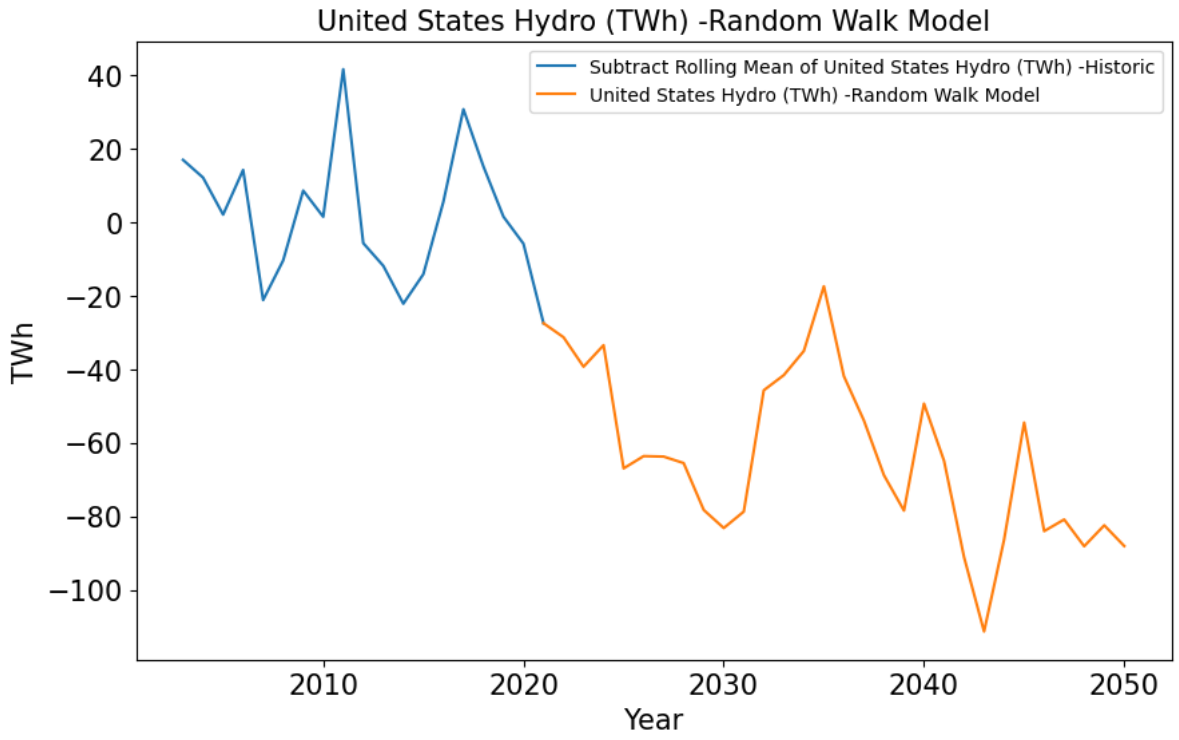


Looks like there's still a trend line, but it does pass stationarity, so I'm moving forward to the baseline random walk model.

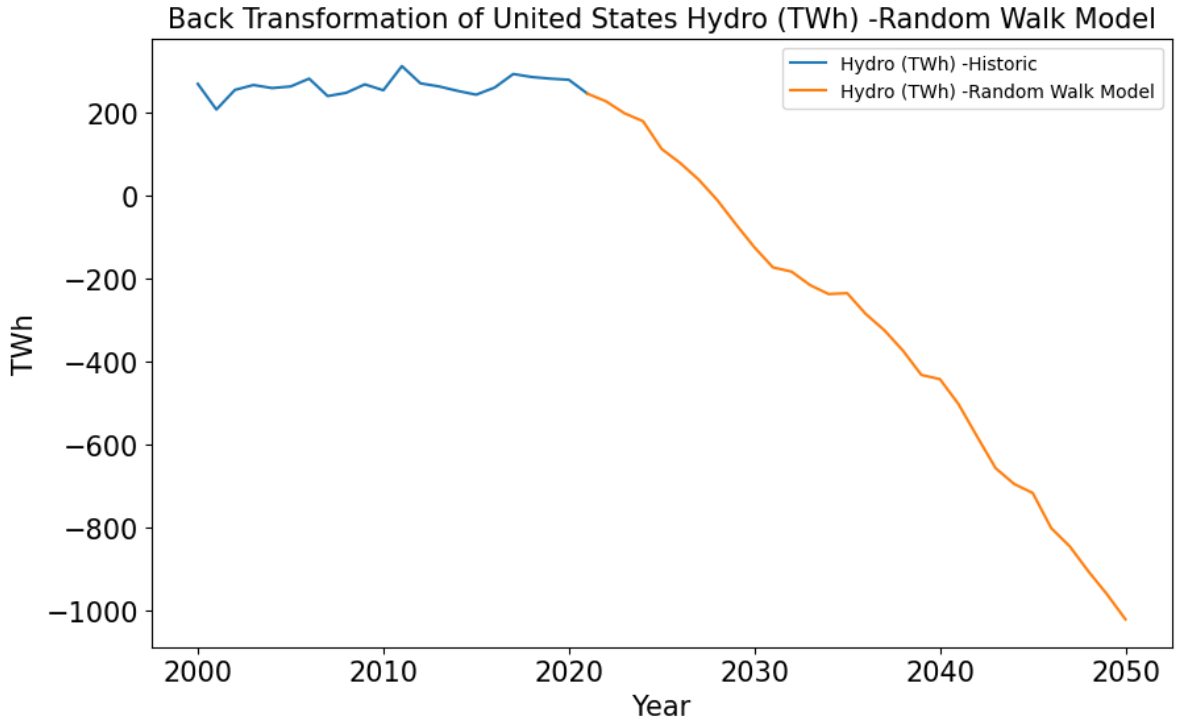
7.5.4 Random Walk Model

In [127]:
```
pd.set_option('display.max_rows',None)
y_hat_proj = random_walk(model_data_t.iloc[:,9:10])
y_hat_proj.columns = [model_data.columns[4]+' -Random Walk Model']

# Plot the transformed Random Walk
pred_graph(model_data_t.iloc[:,9:10], y_hat_proj.iloc[:,0:1])
```

```
Random Walk with Standard Deviation of Transformed Data set: 17.50
```



In [128]:
```
# Prepare the DataFrame for Back Transformation
ranplot = model_data_t.iloc[:,8:10].copy()
ranplot.rename(columns={str(ranplot.columns[1]): str(y_hat_proj.columns[0])},
               inplace=True)

ranplot = pd.concat([ranplot,y_hat_proj],axis=0)

# Perform Back Transformation
y_hat_proj = inv_subtract_roll_mean(ranplot, n=4)

#Plot the new graph
pred_graph(model_data_t.iloc[:,8:9], y_hat_proj.iloc[22:,0:1], 14)
```

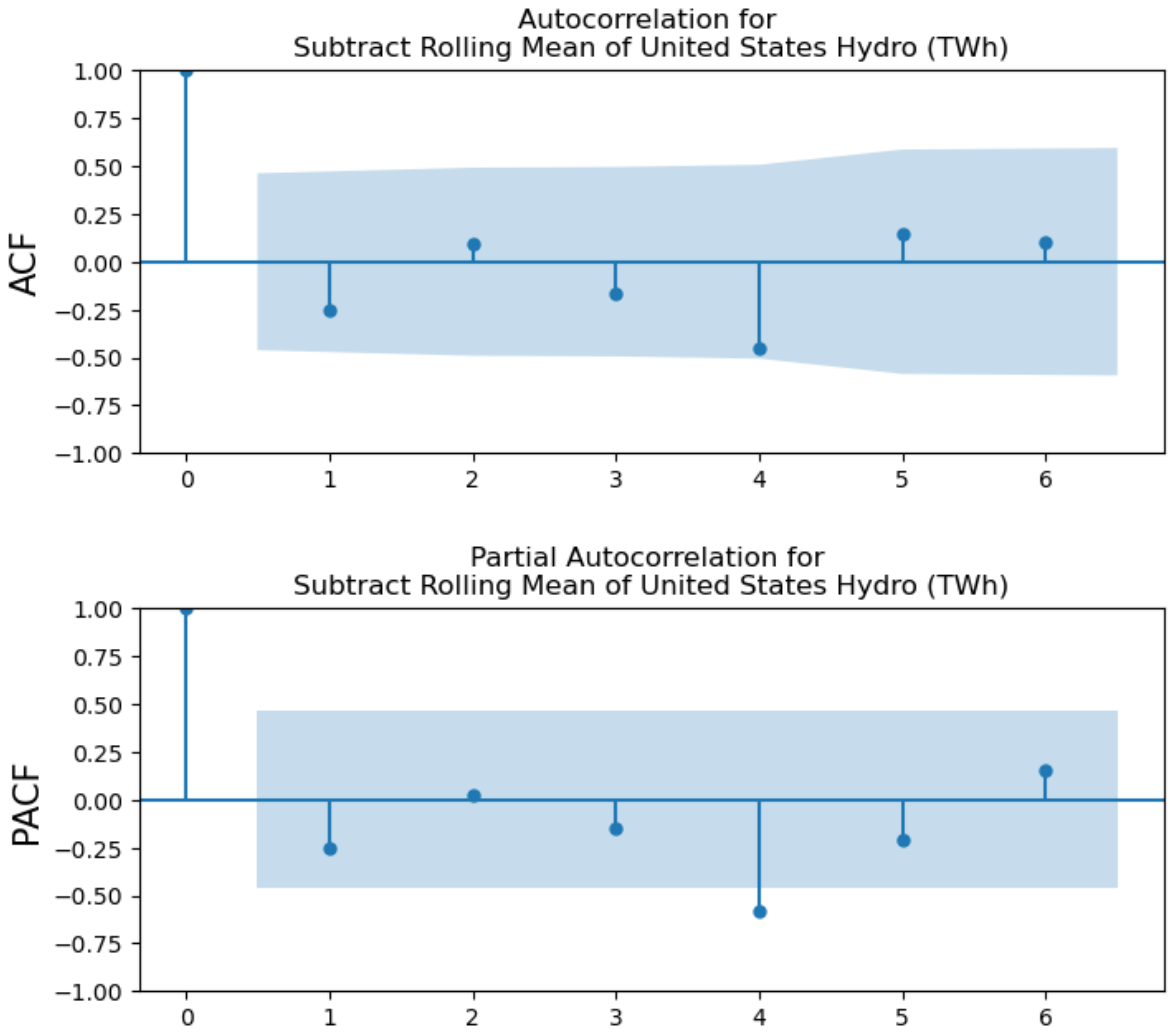


This model is as unreliable as it's name would imply, random walk. A recurring theme is that if the transformed data is along the zero x-axis, random walk will simply not work.

Now I'll move forward with an ARMA model. First, I need to evaluate the ACF and PACF plots to find the best Autoregressive and Moving Average terms.

7.5.5 Evaluating Initial AR and MA Values with ACF and PACF Plots

In [129]:
```
plotacf(model_data_t.iloc[:,9:10], lags = 6)
plotpacf(model_data_t.iloc[:,9:10], lags = 6)
```

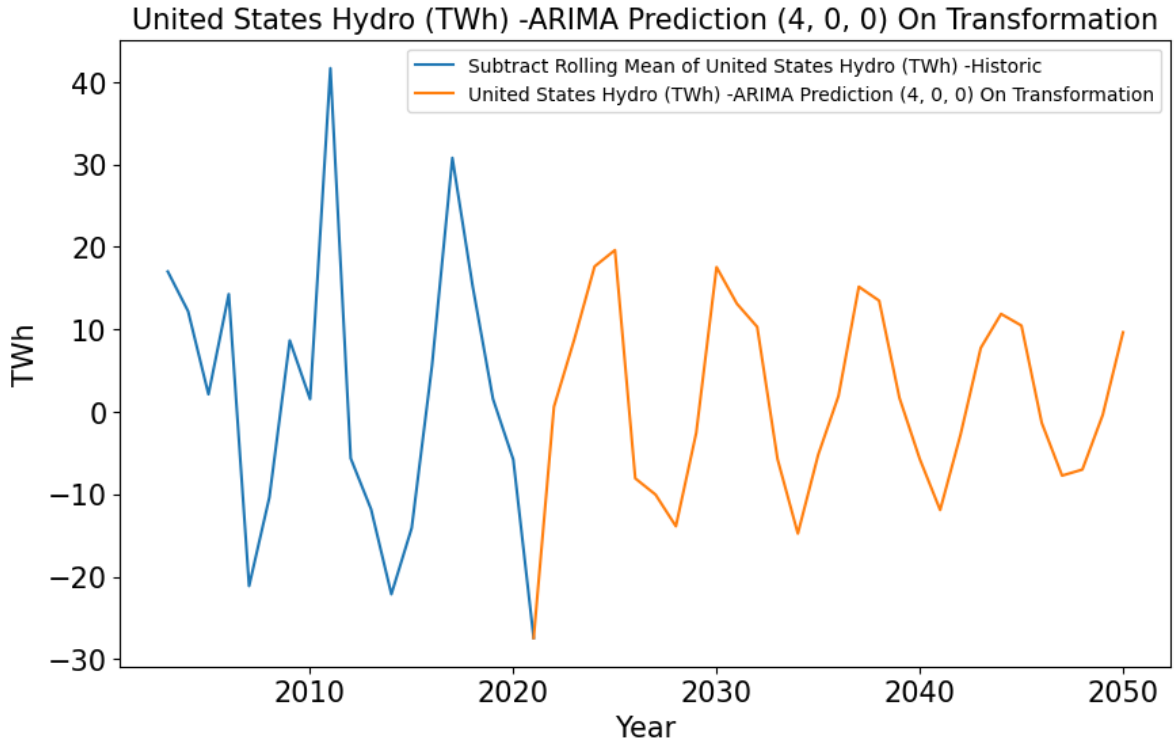


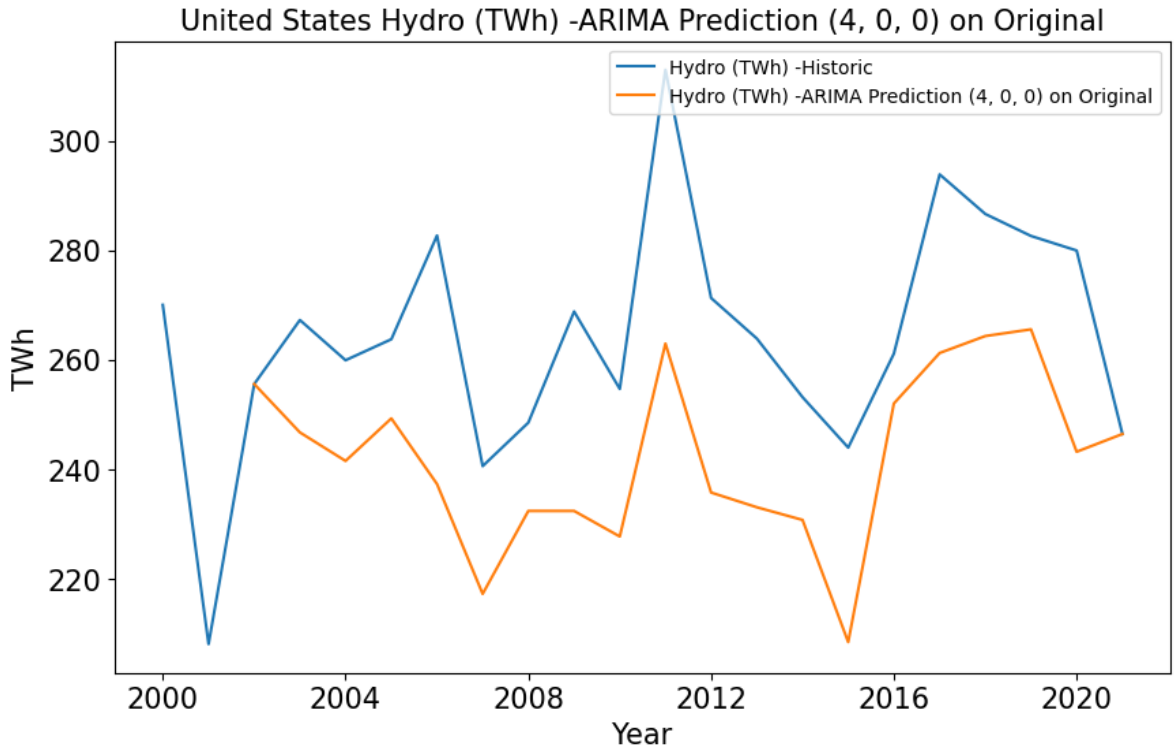
The ACF nearly breaches at 4, and PACF definitely breaches at 4. I will try both (4,0,0) and (4,0,4) for my ARMA model.

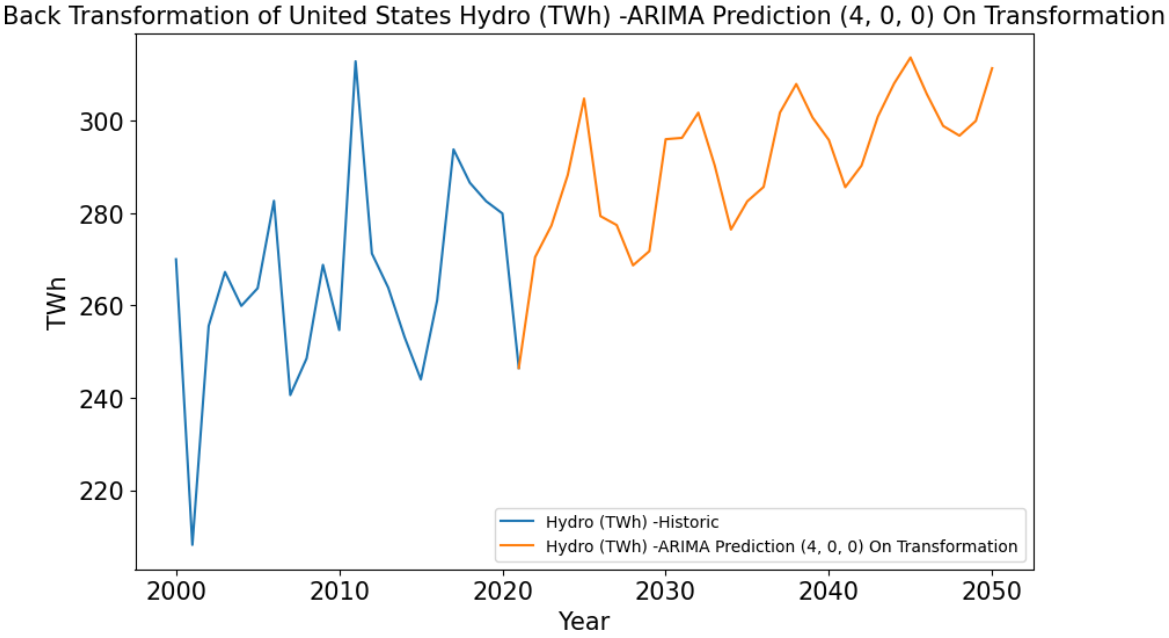
7.5.6 ARMA Model

In [130]:
```
order = (4,0,0)
model_summary(model_data_t.iloc[:,8:10], order, 22, 50, 14,
          tran = 'inv_subtract_roll_mean', n=4)
```

```
ARIMA: (4, 0, 0), RMSE=29.53, AIC=150.73
********************************************************************************
**
United States Hydro (TWh) model results.

                                               SARIMAX Results
================================================================================
==================================
Dep. Variable:     Subtract Rolling Mean of United States Hydro (TWh)   No. O
bservations:                    22
Model:                                            ARIMA(4, 0, 0)   Log L
ikelihood                 -69.367
Date:                                          Sat, 29 Apr 2023   AIC
150.734
Time:                                                  06:20:17   BIC
157.280
Sample:                                              01-01-2000   HQIC
152.276
                                                   - 01-01-2021
Covariance Type:                                             opg
================================================================================
=
                 coef    std err          z      P>|z|      [0.025      0.97
5]
--------------------------------------------------------------------------------
-
const          1.6325      1.189      1.373      0.170      -0.697       3.96
2
ar.L1         -0.3076      0.214     -1.439      0.150      -0.727       0.11
1
ar.L2         -0.1266      0.224     -0.566      0.571      -0.565       0.31
2
ar.L3         -0.4157      0.108     -3.849      0.000      -0.627      -0.20
4
ar.L4         -0.8044      0.213     -3.781      0.000      -1.221      -0.38
7
sigma2        66.7680     49.450      1.350      0.177     -30.152     163.68
9
================================================================================
======
Ljung-Box (L1) (Q):                   1.16   Jarque-Bera (JB):
0.00
Prob(Q):                              0.28   Prob(JB):
1.00
Heteroskedasticity (H):               1.83   Skew:
0.00
Prob(H) (two-sided):                  0.44   Kurtosis:
3.03
================================================================================
======

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (comple
x-step).
```


United States Hydro (TWh) -ARIMA Prediction (4, 0, 0) On Transformation
Subtract Rolling Mean of United States Hydro (TWh) -Historic
United States Hydro (TWh) -ARIMA Prediction (4, 0, 0) On Transformation
TWh
Year
40
30
20
10
0
−10
−20
−30
2010
2020
2030
2040
2050

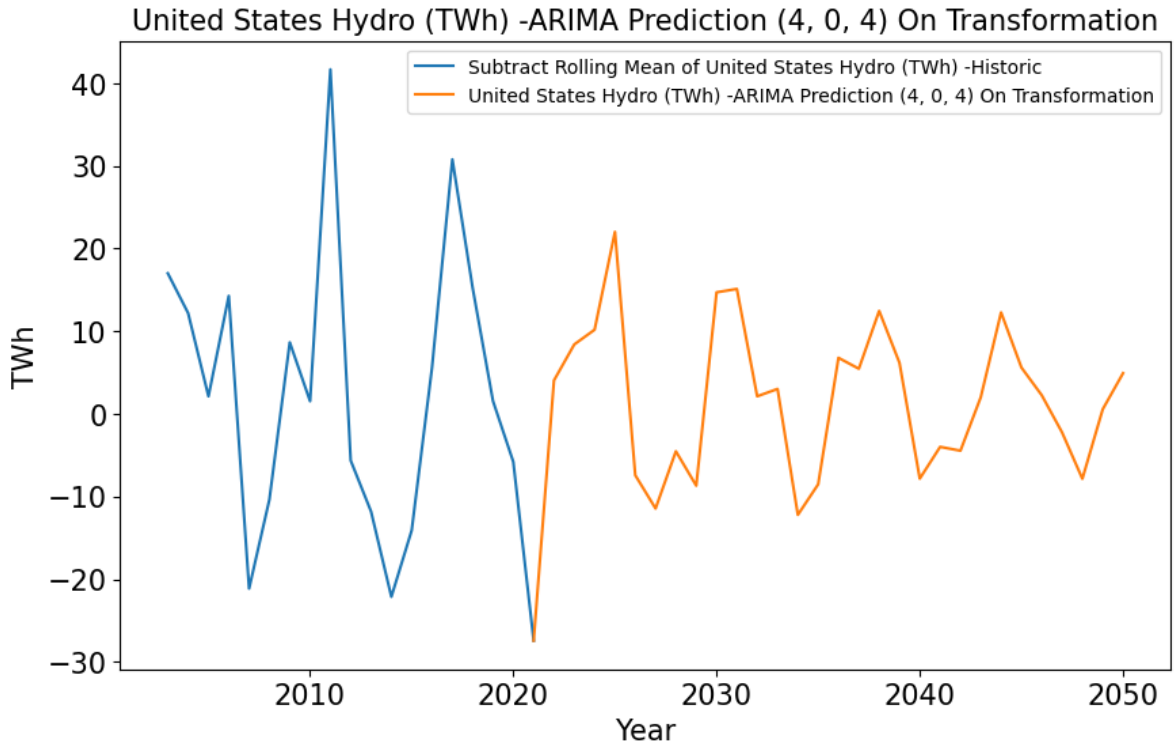

United States Hydro (TWh) -ARIMA Prediction (4, 0, 0) on Original
Hydro (TWh) -Historic
Hydro (TWh) -ARIMA Prediction (4, 0, 0) on Original
TWh
Year
300
280
260
240
220
2000
2004
2008
2012
2016
2020

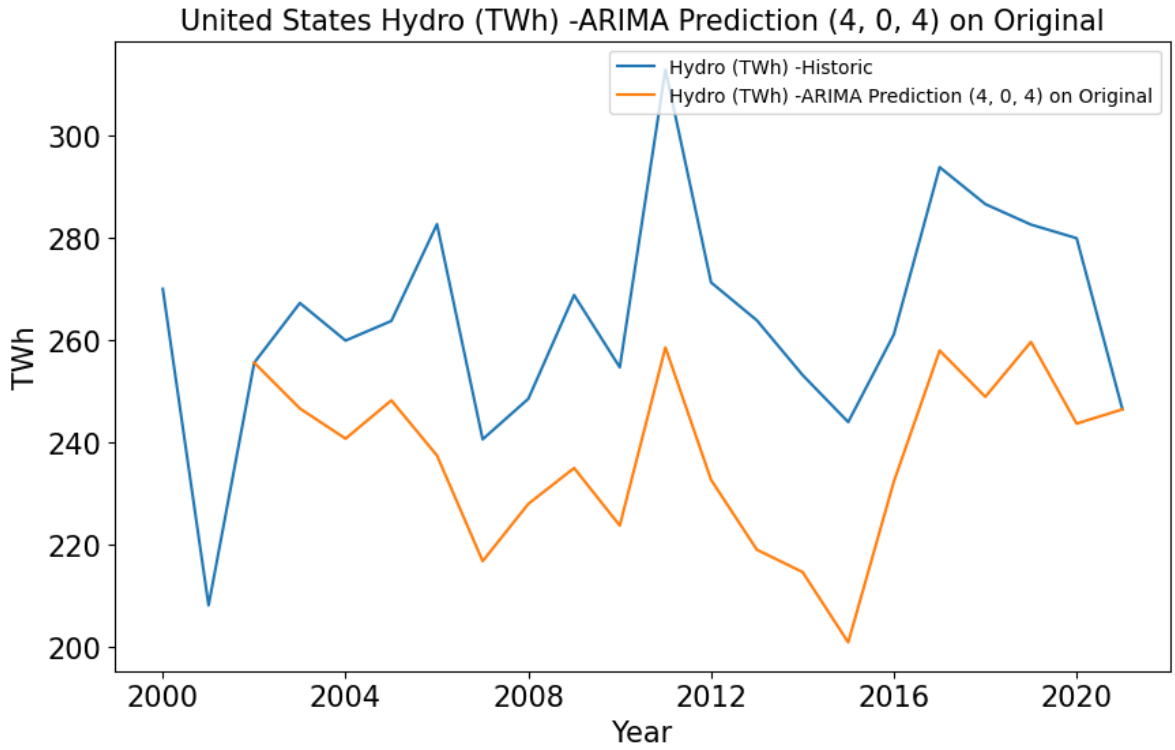

Back Transformation of United States Hydro (TWh) -ARIMA Prediction (4, 0, 0) On Transformation
TWh
Year
300
280
260
240
220
2000
2010
2020
2030
2040
2050
Hydro (TWh) -Historic
Hydro (TWh) -ARIMA Prediction (4, 0, 0) On Transformation

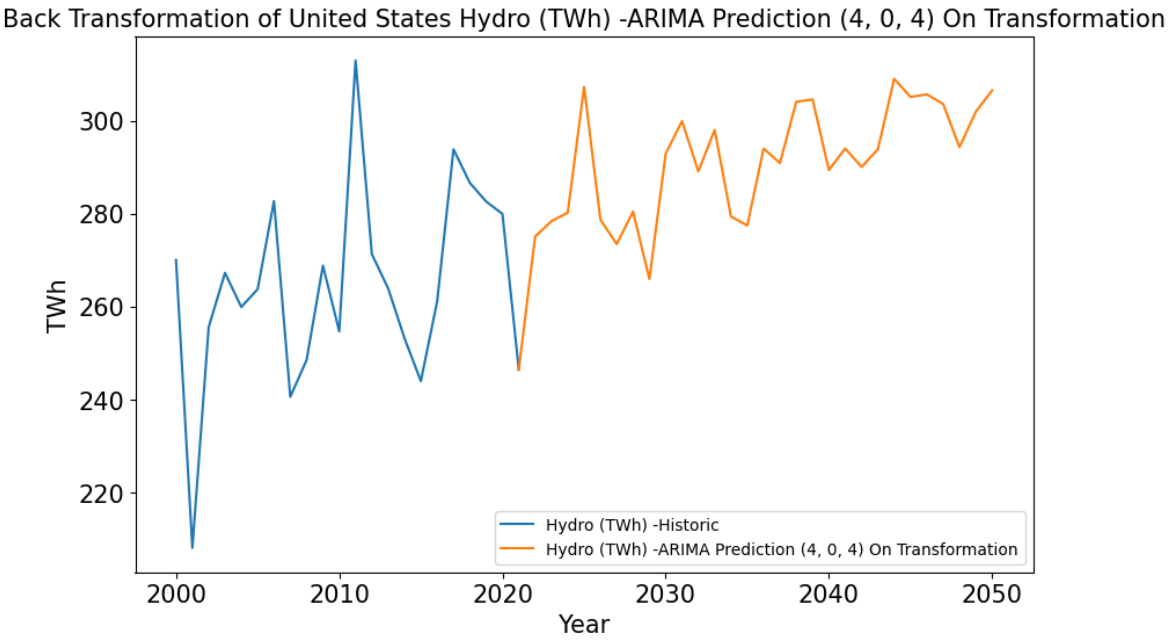
In [131]:
```
order = (4,0,4)
model_summary(model_data_t.iloc[:,8:10], order, 22, 50, 14,
          tran = 'inv_subtract_roll_mean', n=4)
```

```
ARIMA: (4, 0, 4), RMSE=35.06, AIC=151.98
**********************************************************************
**
United States Hydro (TWh) model results.

                                           SARIMAX Results
==========================================================================
==================================
Dep. Variable:     Subtract Rolling Mean of United States Hydro (TWh)   No. O
bservations:                   22
Model:                                          ARIMA(4, 0, 4)   Log L
ikelihood                 -65.992
Date:                                        Sat, 29 Apr 2023   AIC
151.984
Time:                                               06:20:18   BIC
162.895
Sample:                                           01-01-2000   HQIC
154.554
                                                - 01-01-2021
Covariance Type:                                         opg
==========================================================================
=
                 coef    std err          z      P>|z|      [0.025      0.97
5]
--------------------------------------------------------------------------
-
const          1.5372      0.711      2.163      0.031       0.145       2.93
0
ar.L1         -0.1331      0.546     -0.244      0.807      -1.204       0.93
7
ar.L2         -0.3386      0.440     -0.769      0.442      -1.201       0.52
4
ar.L3         -0.1778      0.414     -0.430      0.667      -0.989       0.63
3
ar.L4         -0.8310      0.297     -2.801      0.005      -1.413      -0.25
0
ma.L1         -0.4397      3.848     -0.114      0.909      -7.981       7.10
2
ma.L2          0.2897      4.293      0.067      0.946      -8.125       8.70
4
ma.L3         -0.9756      7.952     -0.123      0.902     -16.561      14.60
9
ma.L4          0.2189      1.839      0.119      0.905      -3.385       3.82
3
sigma2        33.8022    226.790      0.149      0.882    -410.699     478.30
3
==========================================================================
======
Ljung-Box (L1) (Q):                   0.03   Jarque-Bera (JB):
6.23
Prob(Q):                              0.86   Prob(JB):
0.04
Heteroskedasticity (H):               1.40   Skew:
0.77
Prob(H) (two-sided):                  0.67   Kurtosis:
5.11
```

```
==========================================================================
======

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (comple
x-step).
```





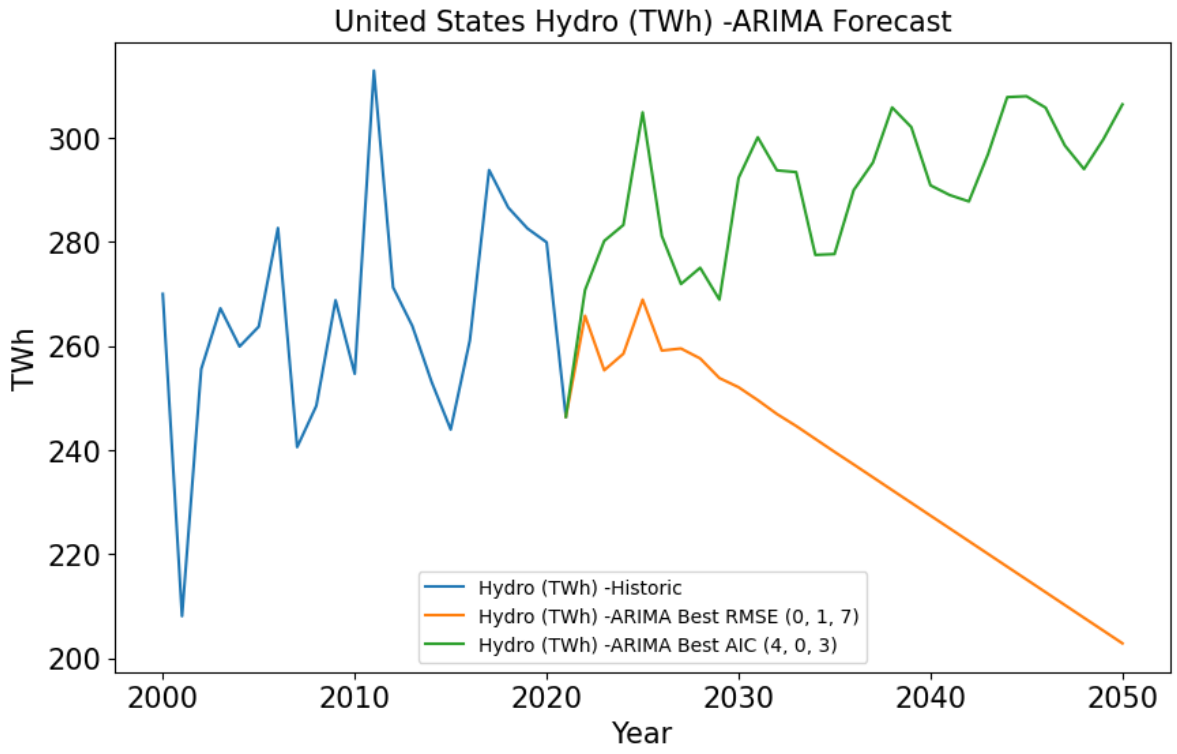


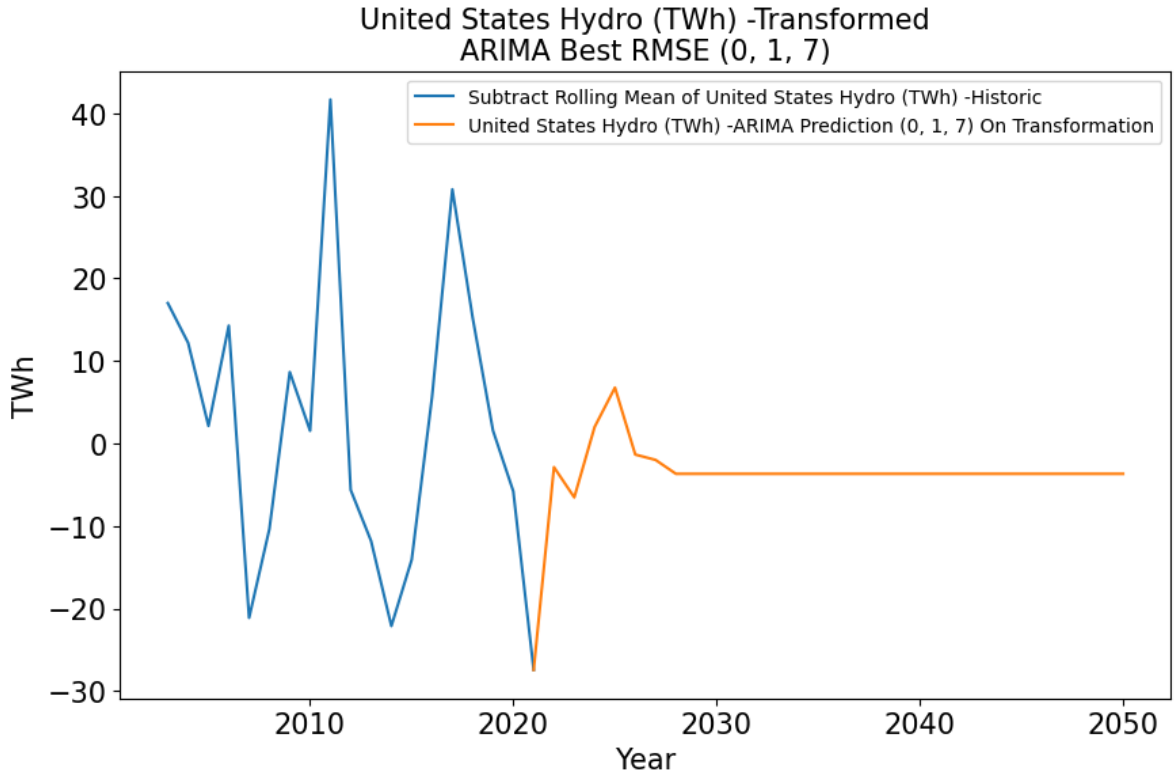
Both of these models look reasonable. In reality, we won't see much change in Hydro, just a change in how much the rivers flow. No further infrastructure is really possible in the United States.

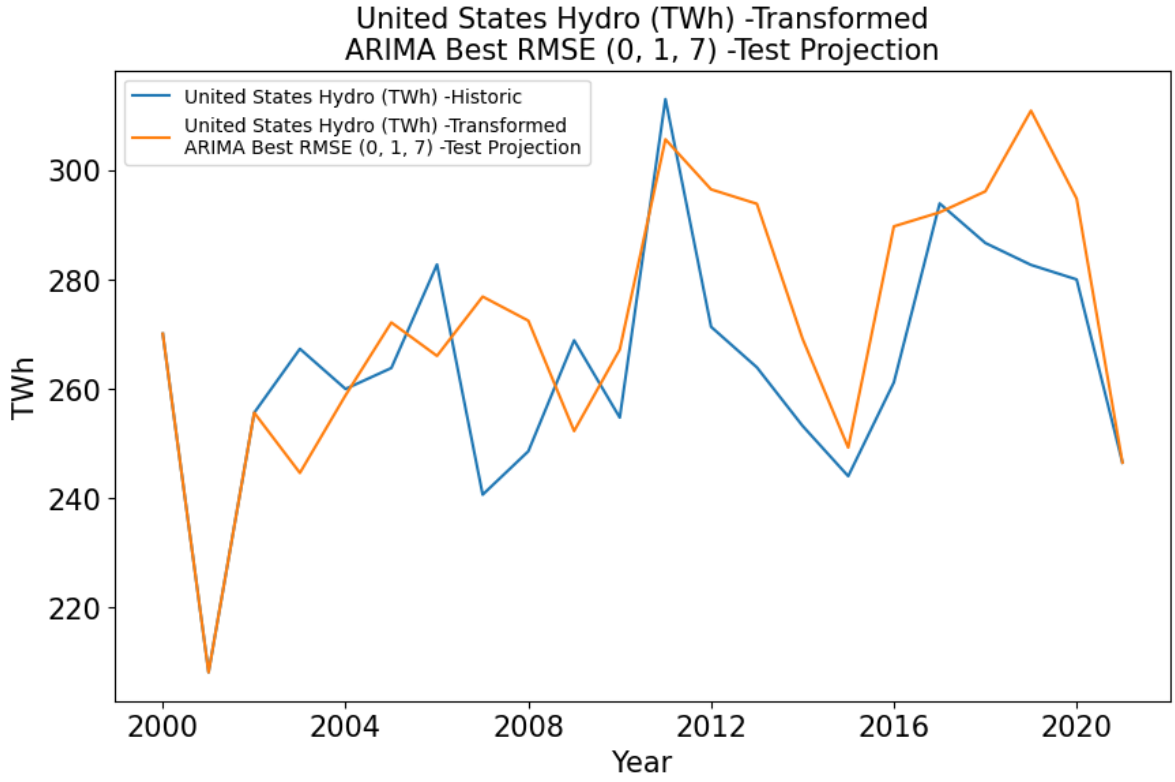
7.5.7 ARIMA Model and Grid Search

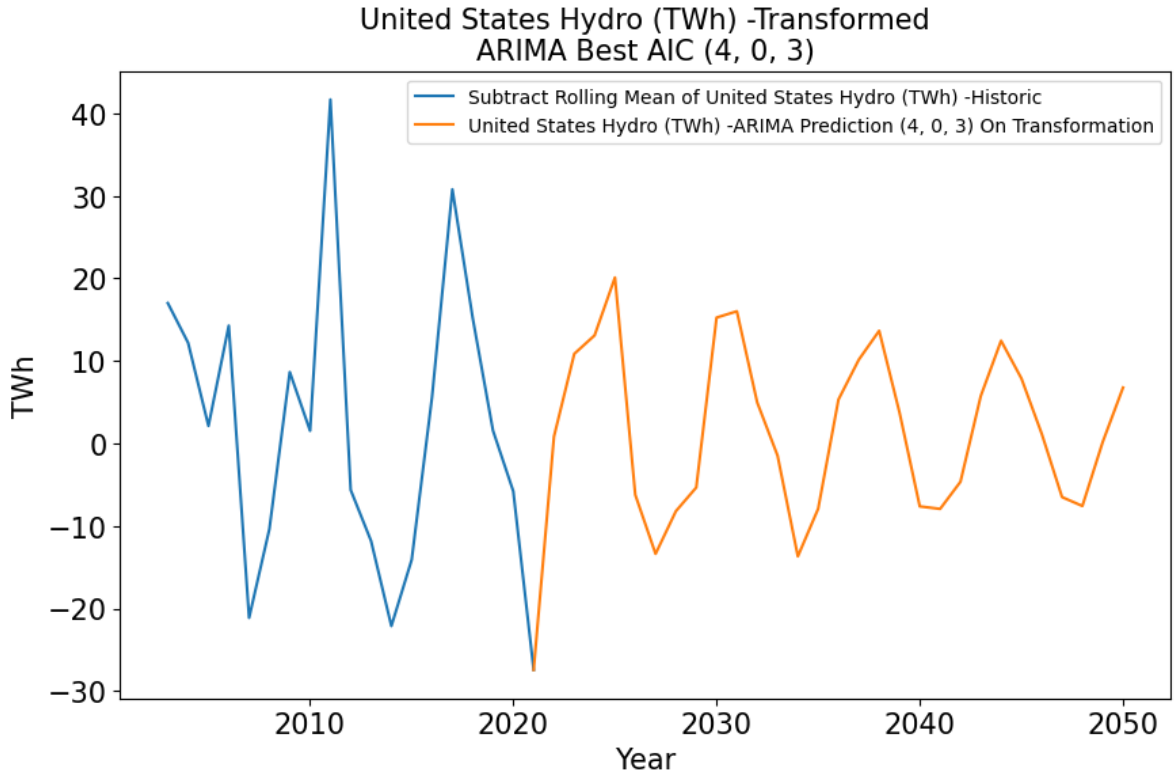
In [132]:
```
oil_rmse_cfg, oil_aic_cfg = arima_pdq(
        model_data_t.iloc[:,8:10], s=14, tran='inv_subtract_roll_mean', n=4)
```

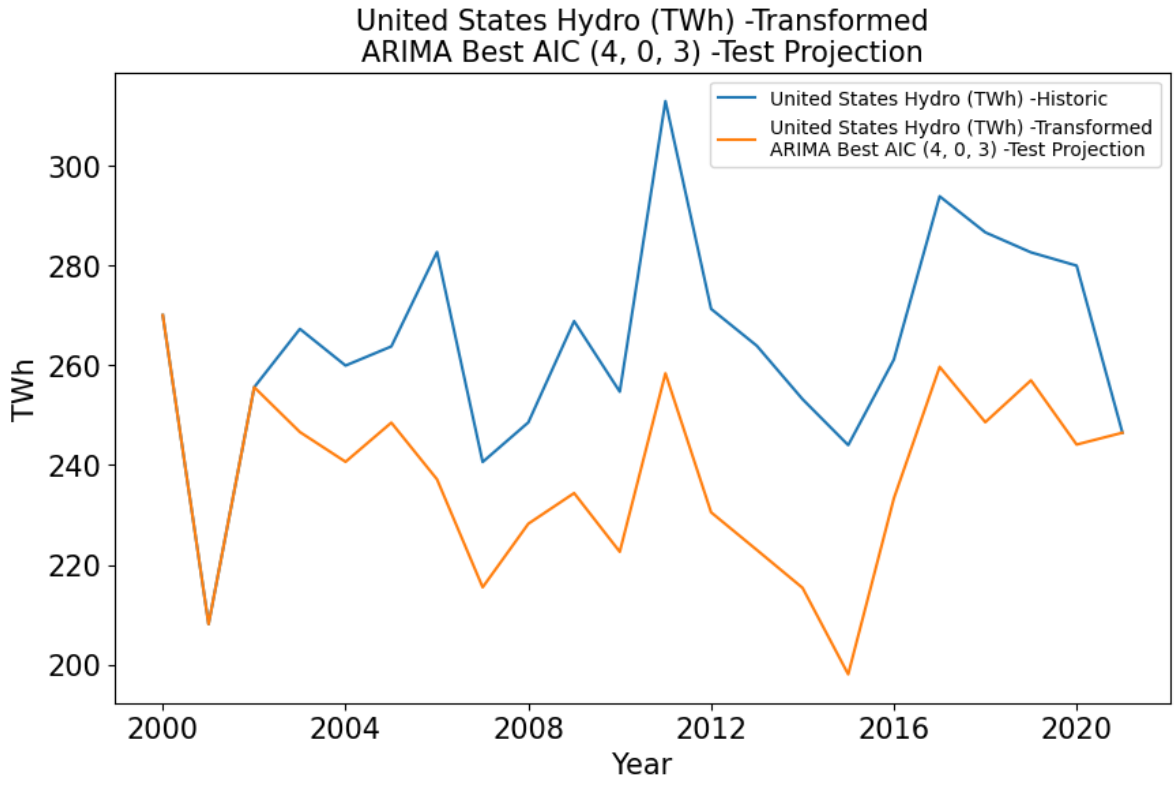
```
RMSE ARIMA: (0, 0, 0), RMSE= 22.64, AIC= 166.69, TWh-2050= 297.18
RMSE ARIMA: (0, 0, 1), RMSE= 21.77, AIC= 167.57, TWh-2050= 288.79
AIC  ARIMA: (0, 0, 2), RMSE= 33.69, AIC= 165.63, TWh-2050= 309.30
AIC  ARIMA: (0, 0, 3), RMSE= 31.35, AIC= 162.43, TWh-2050= 310.63
AIC  ARIMA: (0, 0, 4), RMSE= 25.56, AIC= 160.93, TWh-2050= 306.74
AIC  ARIMA: (0, 0, 7), RMSE= 31.84, AIC= 159.85, TWh-2050= 308.04
RMSE ARIMA: (0, 1, 4), RMSE= 21.01, AIC= 175.84, TWh-2050= 268.92
RMSE ARIMA: (0, 1, 7), RMSE= 19.54, AIC= 175.11, TWh-2050= 202.91
AIC  ARIMA: (2, 0, 2), RMSE= 21.56, AIC= 158.01, TWh-2050= 308.86
AIC  ARIMA: (2, 0, 3), RMSE= 35.57, AIC= 152.50, TWh-2050= 315.92
AIC  ARIMA: (4, 0, 0), RMSE= 29.53, AIC= 150.73, TWh-2050= 311.43
AIC  ARIMA: (4, 0, 1), RMSE= 31.85, AIC= 150.71, TWh-2050= 304.16
AIC  ARIMA: (4, 0, 3), RMSE= 35.21, AIC= 149.84, TWh-2050= 306.44
Best RMSE ARIMA: (0, 1, 7) RMSE= 19.54 AIC= 175.11, TWh-2050= 202.91
Best AIC  ARIMA: (4, 0, 3) RMSE= 35.21 AIC= 149.84, TWh-2050= 306.44
Graph_formatting produced error at (0, 1, 7) or (4, 0, 3)
```




United States Hydro (TWh) -Transformed
ARIMA Best RMSE (0, 1, 7)
Subtract Rolling Mean of United States Hydro (TWh) -Historic
United States Hydro (TWh) -ARIMA Prediction (0, 1, 7) On Transformation
TWh
Year
40
30
20
10
0
−10
−20
−30
2010
2020
2030
2040
2050


United States Hydro (TWh) -Transformed
ARIMA Best RMSE (0, 1, 7) -Test Projection
United States Hydro (TWh) -Historic
United States Hydro (TWh) -Transformed
ARIMA Best RMSE (0, 1, 7) -Test Projection
TWh
Year
300
280
260
240
220
2000
2004
2008
2012
2016
2020



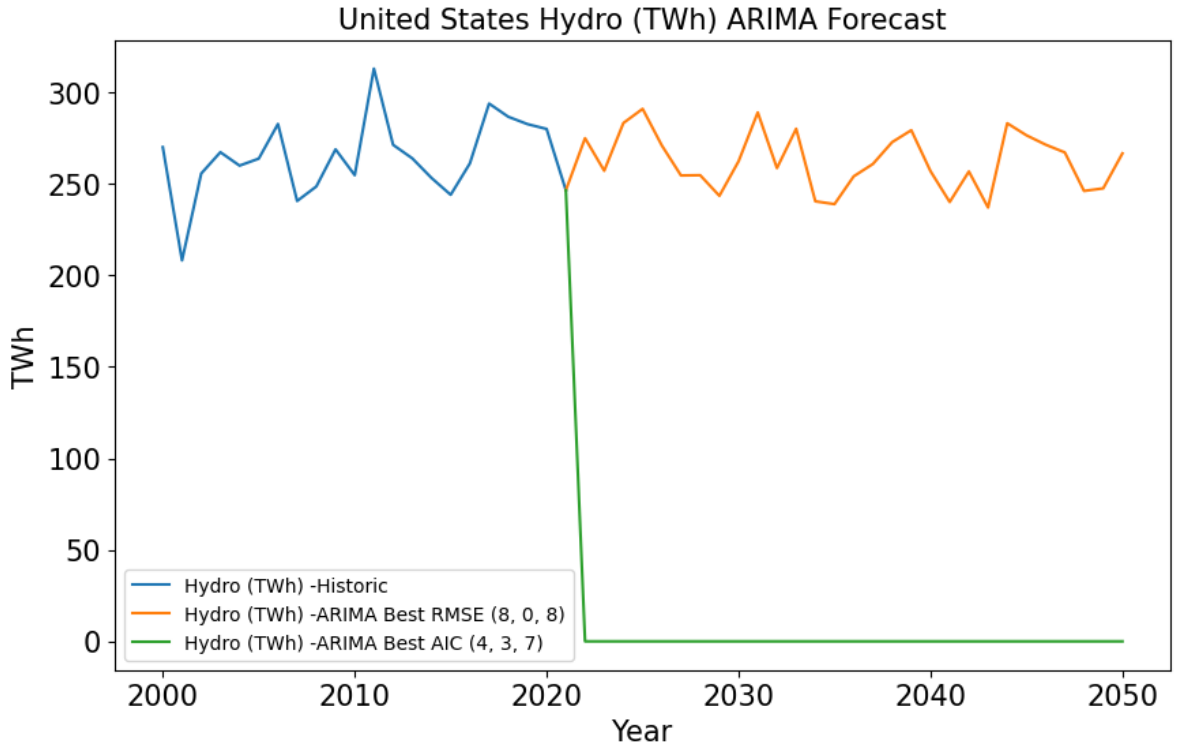


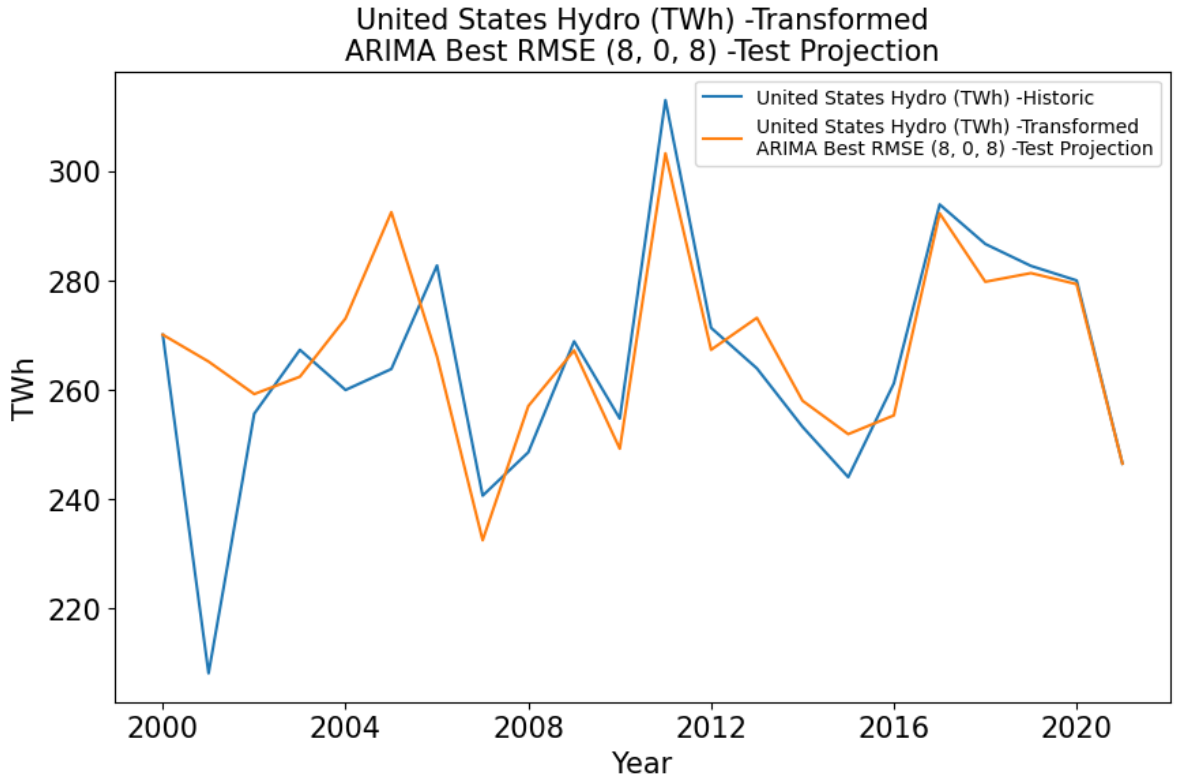
I plotted the best RMSE, and it was a flat line. Neither graph is convincing. I'll have to try using one of the ARMA models above or an non-transformation model.

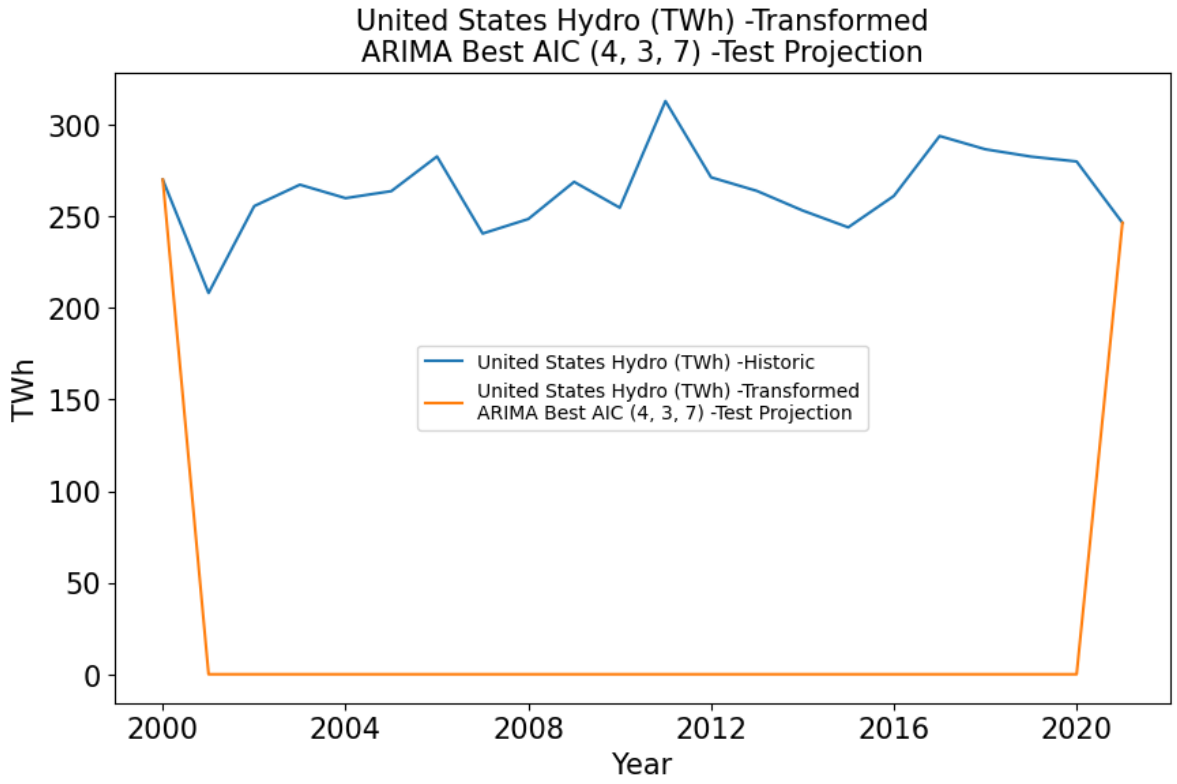
7.5.8 ARIMA Without Transformation

In [133]:
```
oil_rmse_cfg, oil_aic_cfg = arima_pdq_no_tran(
                                        model_data.iloc[:,4:5], 22, 50, s=14)
```

```
United States Hydro (TWh) Grid Search:
RMSE ARIMA: (0, 0, 0), RMSE= 20.70, AIC= 200.57, TWh-2050= 264.36
RMSE ARIMA: (0, 0, 1), RMSE= 20.48, AIC= 202.26, TWh-2050= 264.32
RMSE ARIMA: (0, 0, 4), RMSE= 19.01, AIC= 201.86, TWh-2050= 264.72
RMSE ARIMA: (0, 0, 5), RMSE= 18.91, AIC= 203.66, TWh-2050= 264.73
RMSE ARIMA: (0, 0, 7), RMSE= 18.01, AIC= 206.16, TWh-2050= 263.86
AIC  ARIMA: (0, 1, 1), RMSE= 23.17, AIC= 195.44, TWh-2050= 267.77
AIC  ARIMA: (0, 2, 2), RMSE= 52.01, AIC= 193.60, TWh-2050= 275.32
RMSE ARIMA: (1, 0, 7), RMSE= 17.78, AIC= 207.18, TWh-2050= 263.70
RMSE ARIMA: (1, 0, 8), RMSE= 17.62, AIC= 209.06, TWh-2050= 263.78
RMSE ARIMA: (2, 0, 5), RMSE= 17.54, AIC= 203.88, TWh-2050= 264.72
RMSE ARIMA: (2, 0, 7), RMSE= 17.45, AIC= 208.78, TWh-2050= 263.86
RMSE ARIMA: (2, 0, 8), RMSE= 17.27, AIC= 210.44, TWh-2050= 263.08
RMSE ARIMA: (3, 0, 7), RMSE= 16.68, AIC= 205.54, TWh-2050= 265.57
RMSE ARIMA: (4, 0, 7), RMSE= 16.68, AIC= 205.04, TWh-2050= 246.42
AIC  ARIMA: (4, 1, 2), RMSE= 19.47, AIC= 191.01, TWh-2050= 273.14
AIC  ARIMA: (4, 2, 3), RMSE= 51.13, AIC= 189.67, TWh-2050= 313.00
AIC  ARIMA: (4, 3, 7), RMSE= 253.49, AIC= 24.00, TWh-2050= 0.00
RMSE ARIMA: (5, 0, 7), RMSE= 16.37, AIC= 207.99, TWh-2050= 255.05
RMSE ARIMA: (5, 0, 8), RMSE= 16.06, AIC= 209.14, TWh-2050= 266.84
RMSE ARIMA: (7, 0, 8), RMSE= 15.63, AIC= 207.52, TWh-2050= 267.41
RMSE ARIMA: (8, 0, 7), RMSE= 15.63, AIC= 206.81, TWh-2050= 263.73
RMSE ARIMA: (8, 0, 8), RMSE= 15.21, AIC= 207.63, TWh-2050= 266.62
Best RMSE ARIMA: (8, 0, 8) RMSE= 15.21 AIC= 207.63, TWh-2050= 266.62
Best AIC  ARIMA: (4, 3, 7) RMSE= 253.49 AIC= 24.00, TWh-2050= 0.00
```




United States Hydro (TWh) -Transformed
ARIMA Best RMSE (8, 0, 8) -Test Projection
United States Hydro (TWh) -Historic
United States Hydro (TWh) -Transformed
ARIMA Best RMSE (8, 0, 8) -Test Projection
TWh
300
280
260
240
220
2000
2004
2008
2012
2016
2020
Year


United States Hydro (TWh) -Transformed
ARIMA Best AIC (4, 3, 7) -Test Projection
United States Hydro (TWh) -Historic
United States Hydro (TWh) -Transformed
ARIMA Best AIC (4, 3, 7) -Test Projection
TWh
300
250
200
150
100
50
0
2000
2004
2008
2012
2016
2020
Year

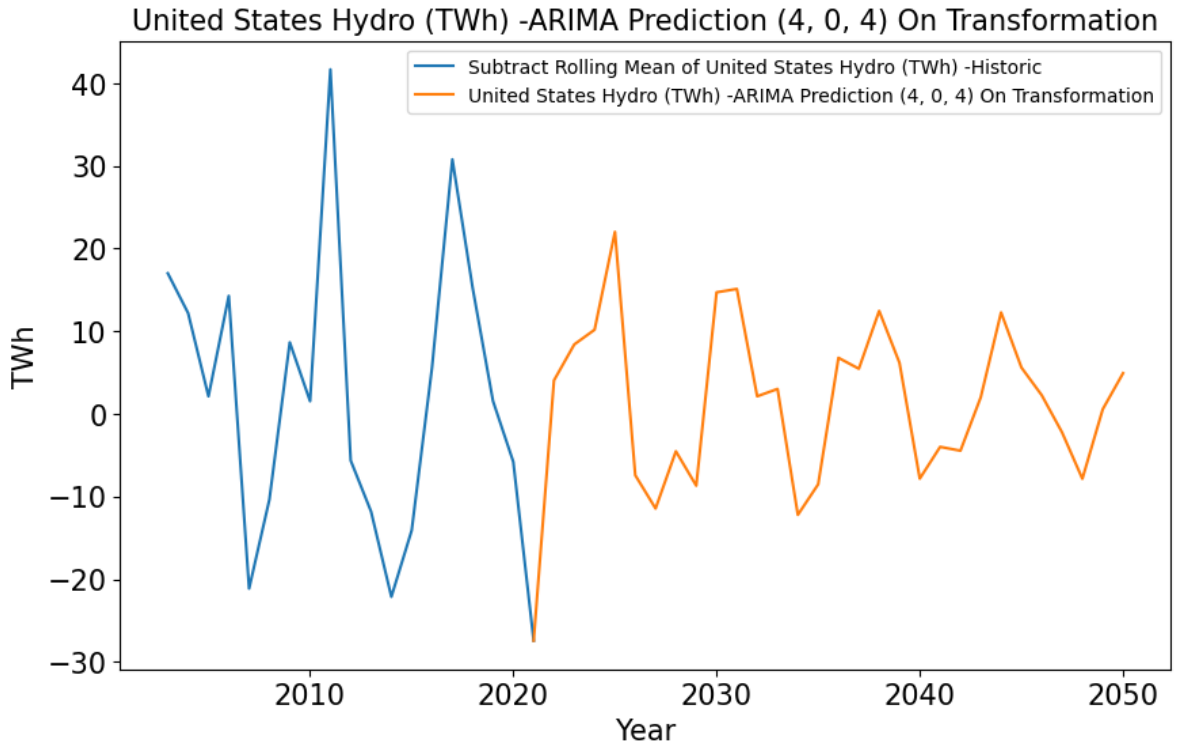
7.5.9 Model Selection

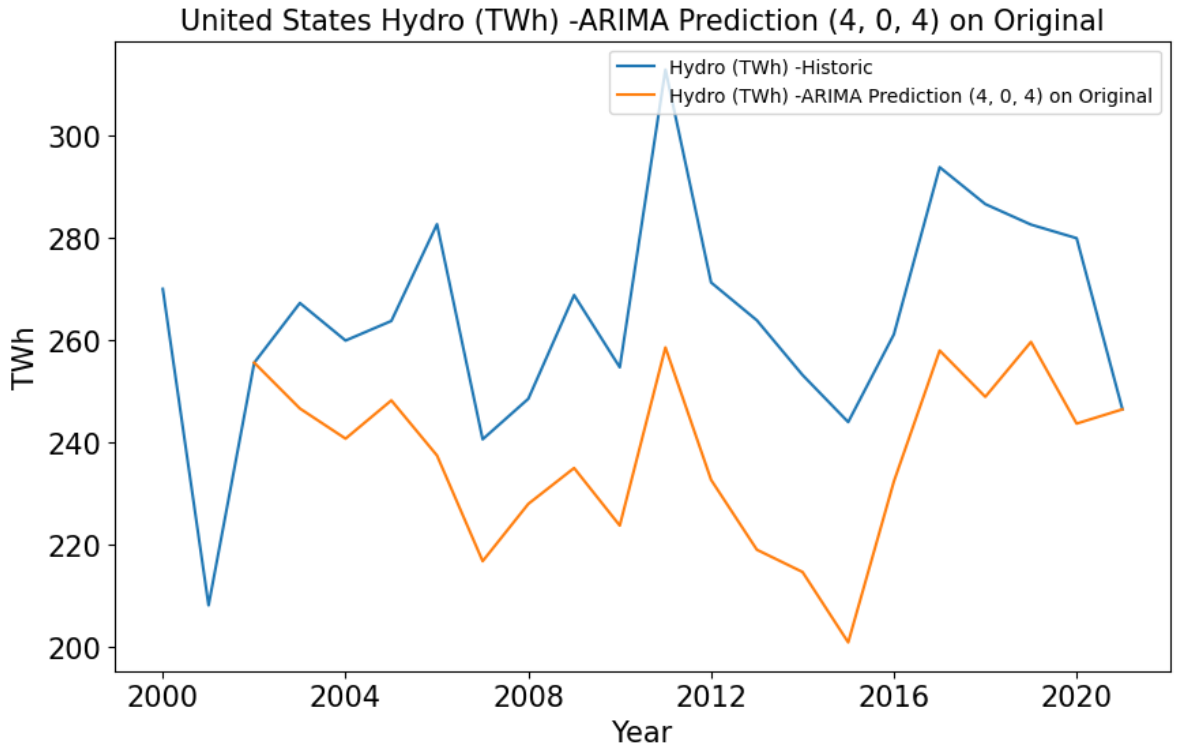
In [134]:
```
#Deleteme
stored_hyd = selected_models.copy()
```

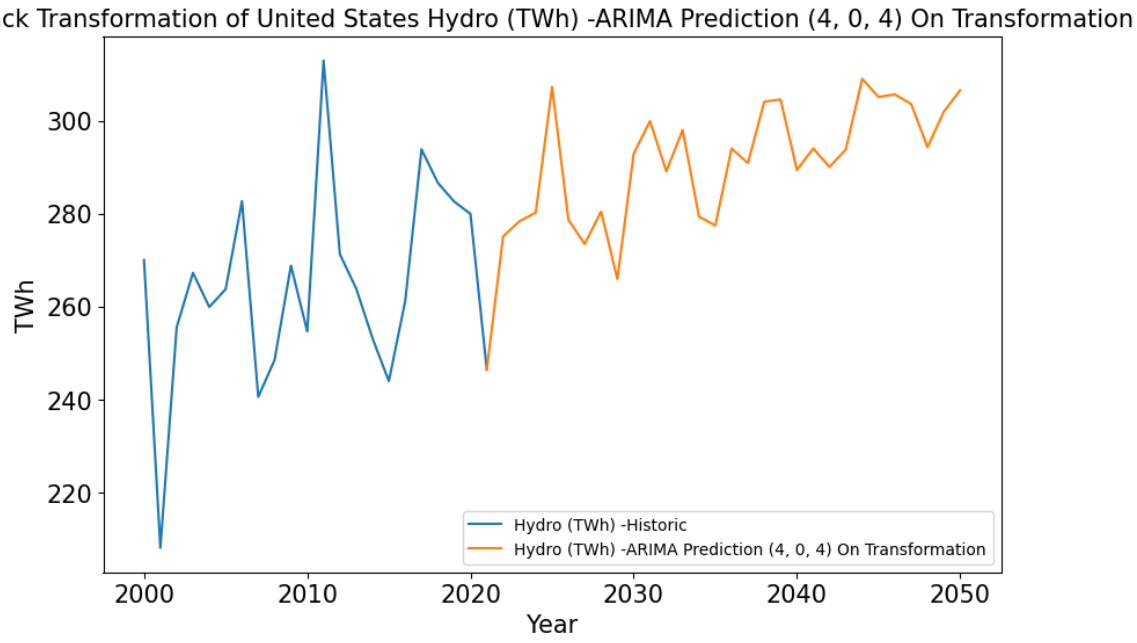
In [135]:
```
selected_models = stored_hyd.copy() #deleteme
df_hydro = model_data_t.iloc[:,8:10]

#                         Model,       df_o,     pdq
selected_models.append(['ARIMA_tran', df_hydro, (4,0,4),
                        'inv_subtract_roll_mean', 4, None])

m = 4
model_summary(selected_models[m][1], selected_models[m][2], 22, 50, 14,
              selected_models[m][3], selected_models[m][4])
```

```
ARIMA: (4, 0, 4), RMSE=35.06, AIC=151.98
**********************************************************************
**
United States Hydro (TWh) model results.

                                           SARIMAX Results
==========================================================================
==================================
Dep. Variable:     Subtract Rolling Mean of United States Hydro (TWh)   No. O
bservations:                   22
Model:                                          ARIMA(4, 0, 4)   Log L
ikelihood                 -65.992
Date:                                        Sat, 29 Apr 2023   AIC
151.984
Time:                                               06:21:33   BIC
162.895
Sample:                                           01-01-2000   HQIC
154.554
                                                - 01-01-2021
Covariance Type:                                         opg
==========================================================================
=
                 coef    std err          z      P>|z|      [0.025      0.97
5]
--------------------------------------------------------------------------
-
const          1.5372      0.711      2.163      0.031       0.145       2.93
0
ar.L1         -0.1331      0.546     -0.244      0.807      -1.204       0.93
7
ar.L2         -0.3386      0.440     -0.769      0.442      -1.201       0.52
4
ar.L3         -0.1778      0.414     -0.430      0.667      -0.989       0.63
3
ar.L4         -0.8310      0.297     -2.801      0.005      -1.413      -0.25
0
ma.L1         -0.4397      3.848     -0.114      0.909      -7.981       7.10
2
ma.L2          0.2897      4.293      0.067      0.946      -8.125       8.70
4
ma.L3         -0.9756      7.952     -0.123      0.902     -16.561      14.60
9
ma.L4          0.2189      1.839      0.119      0.905      -3.385       3.82
3
sigma2        33.8022    226.790      0.149      0.882    -410.699     478.30
3
==========================================================================
======
Ljung-Box (L1) (Q):                   0.03   Jarque-Bera (JB):
6.23
Prob(Q):                              0.86   Prob(JB):
0.04
Heteroskedasticity (H):               1.40   Skew:
0.77
Prob(H) (two-sided):                  0.67   Kurtosis:
5.11
```

```
==========================================================================
======

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (comple
x-step).
```







I decided to use (4,0,4) because it retains the seasonal element to hydro power. Hydro is dependent on the rain, and rain is not as reliable as the wind or sun. In dryer years, there is less rain. Also, this graph's RMSE, 35.06, is not dramatically different than the best RMSE with transformation, 19.34.

That being said, there is not a lot of growth available to hydro, because we simply don't want to build more dams, which create their own environmental problems.

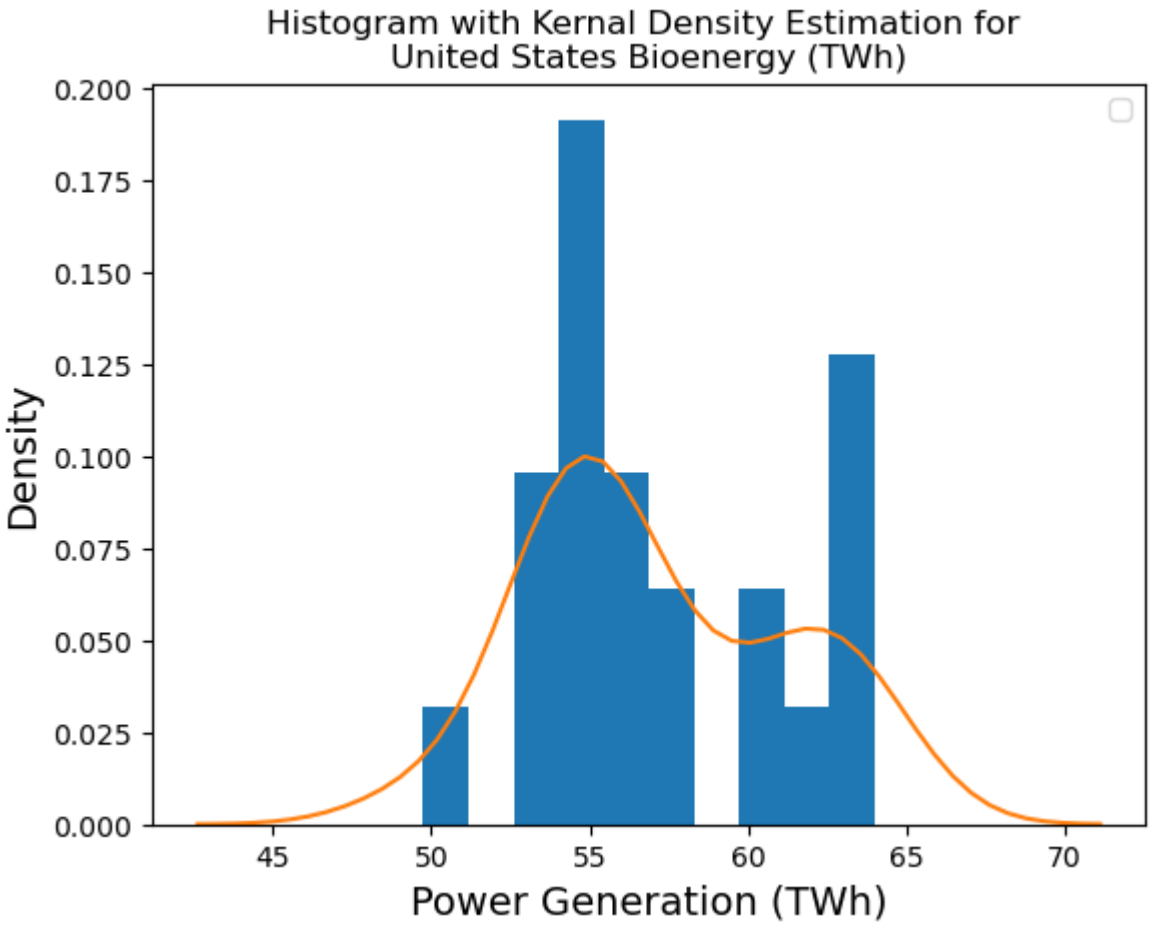
7.6 Bioenergy

7.6.1 Distribution Investigation

In [136]:
```
stored_model_data = model_data.copy()
stored_model_data_t = model_data_t.copy()
```

In [137]:
```
model_data = stored_model_data.copy()
model_data_t = stored_model_data_t.copy()
```

In [138]:
```
hist(model_data.iloc[:,5:6])
```



In [139]:
```
stats_block = s_block(model_data.iloc[:,5:6], stats_block)
stats_block
```

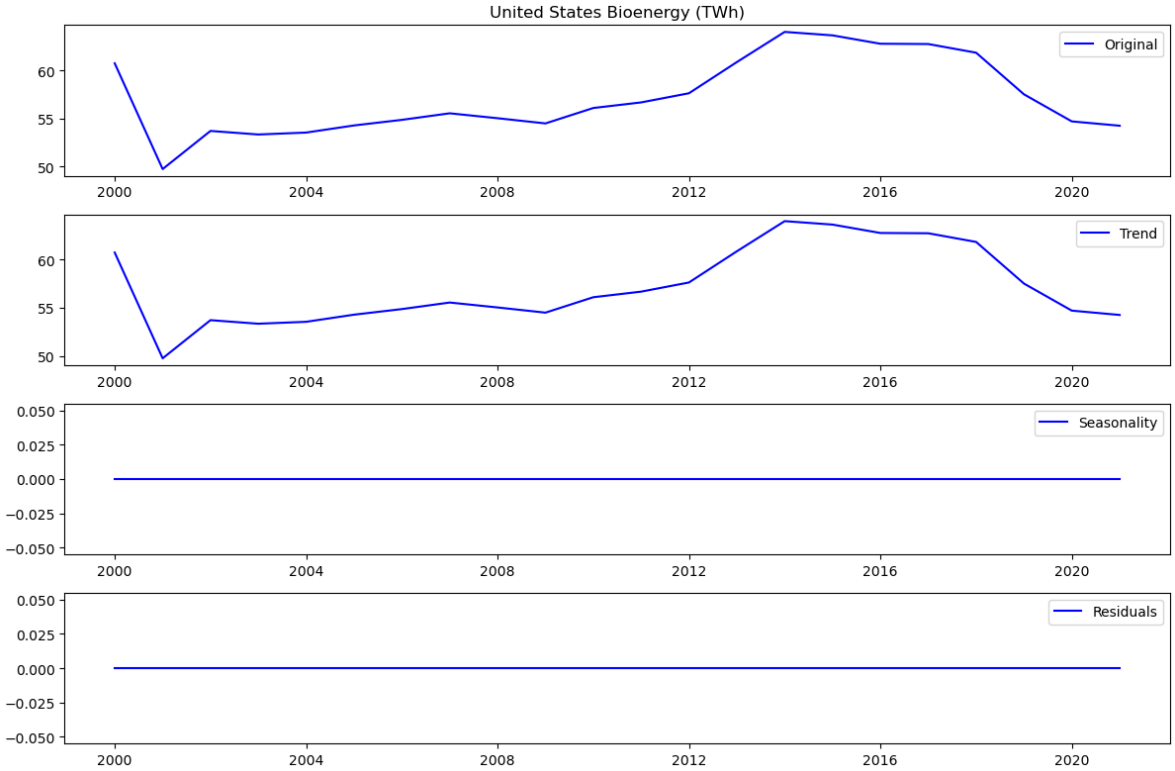
Out[139]:

	Dataset	Mean	Median	Standard Deviation	Skew	Fisher Kurtosis
0	United States Coal (TWh)	1,607.17	1,744.66	399.15	-0.68	-0.90
1	United States Gas (TWh)	1,060.39	1,000.70	325.82	0.29	-1.21
2	United States Oil (TWh)	67.20	45.97	36.59	0.89	-0.92
3	United States Nuclear (TWh)	790.04	790.04	15.55	-0.64	-0.53
4	United States Hydro (TWh)	264.36	263.82	21.08	-0.22	1.03
5	United States Bioenergy (TWh)	57.18	55.81	3.92	0.35	-0.98

This is very close to normal distribution, with a slightly positive skew. It has wide, flat, tails.

7.6.2 Decomposing The Data

In [140]:
```
decomp_graph(model_data.iloc[:,5:6])
```

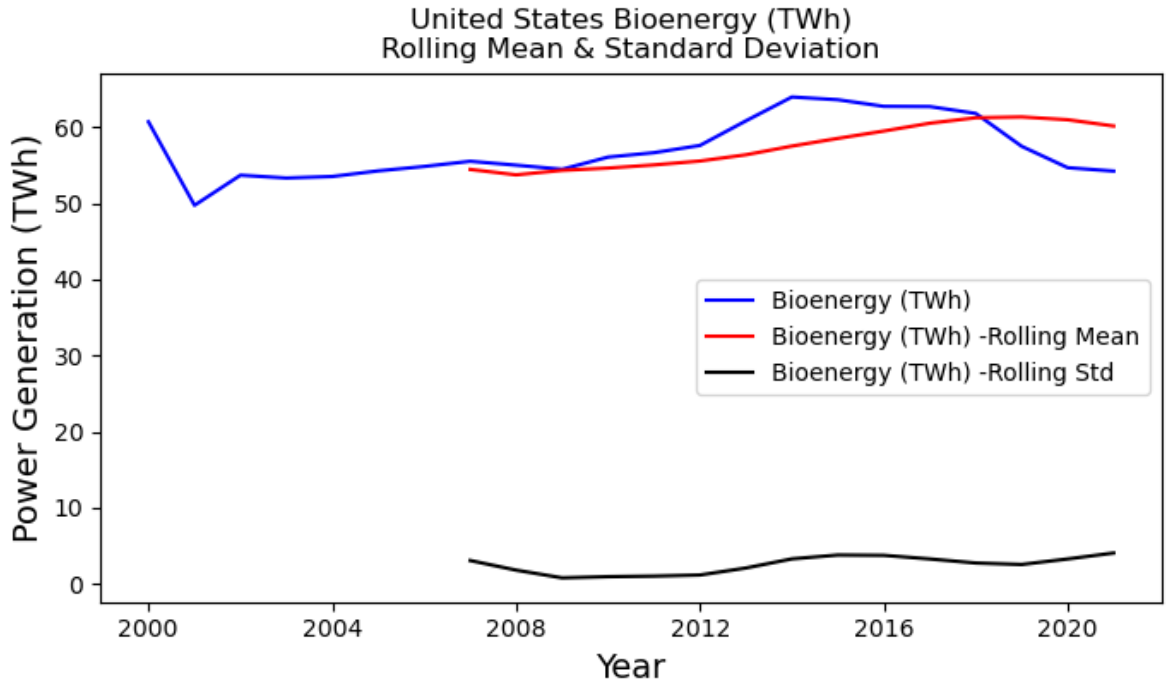


No seasonality or residuals. I'll check for stationarity.

7.6.3 Checking for Stationarity and Flattening

In [141]:
```
pd.set_option('display.max_rows', 30)
stationarity_check(model_data.iloc[:,5:6], s=14)
```

```
United States Bioenergy (TWh)
Results of Dickey-Fuller test:

Test Statistic                -1.06
p-value                        0.73
#Lags Used                     9.00
Number of Observations Used   12.00
Critical Value (1%)           -4.14
Critical Value (5%)           -3.15
Critical Value (10%)          -2.71
dtype: float64
```



This is not close to stationary. After playing around with transformations, I found that if I took the log and then subtracted the rolling average of nine values, I can get it stationary.

In [142]:
```
stored = model_data_t.copy()
```

In [143]:
```
model_data_t = stored.copy()
t = subtract_roll_mean(log_transform(model_data.iloc[:,5:6]), n=9)
model_data_t.insert(11,t.columns[0],t)

model_data_t.iloc[:,10:12].head(10)
```

Out[143]:

Year	United States Bioenergy (TWh)	Subtract Rolling Mean of Natural Logged United States Bioenergy (TWh)
2000-01-01	60.73	NaN
2001-01-01	49.75	NaN
2002-01-01	53.71	NaN
2003-01-01	53.34	NaN
2004-01-01	53.54	NaN
2005-01-01	54.28	NaN
2006-01-01	54.86	NaN
2007-01-01	55.54	NaN
2008-01-01	55.03	0.01
2009-01-01	54.49	0.01

Before I move forward, I want to make sure I can transform this data back after modeling.

In [144]:
```
test = model_data_t.iloc[:,10:12].copy()

test['Results'] = inv_log_inv_sub(test, n=9)
test
```

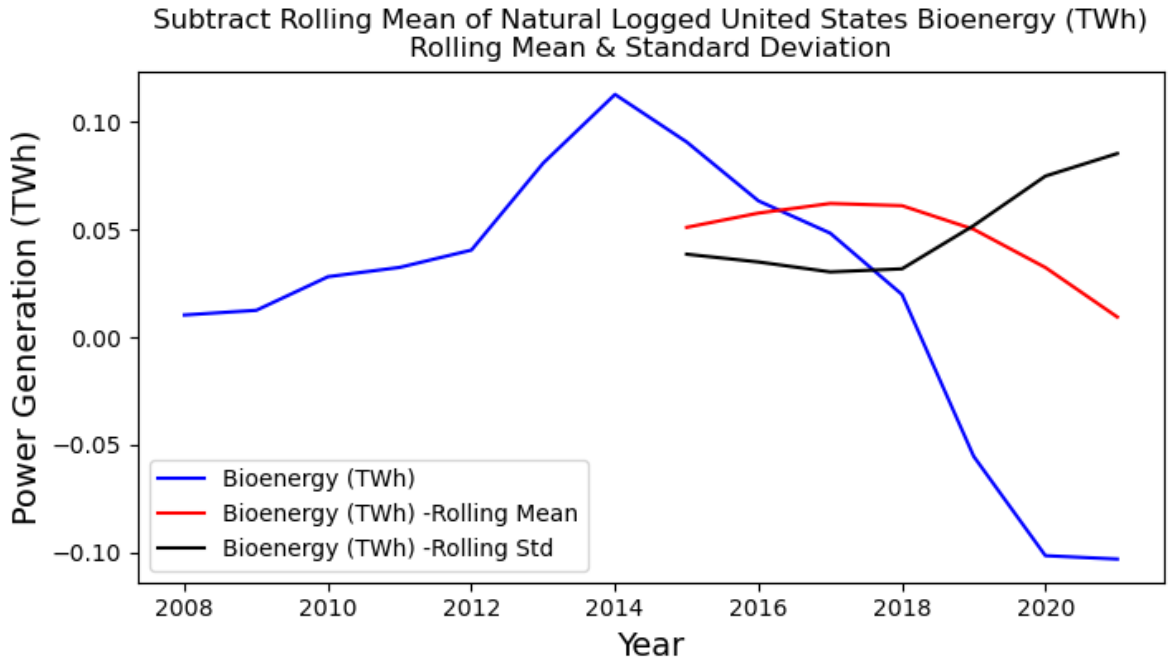
Out[144]:

Year	United States Bioenergy (TWh)	Subtract Rolling Mean of Natural Logged United States Bioenergy (TWh)	Results
2000-01-01	60.73	NaN	60.73
2001-01-01	49.75	NaN	49.75
2002-01-01	53.71	NaN	53.71
2003-01-01	53.34	NaN	53.34
2004-01-01	53.54	NaN	53.54
2005-01-01	54.28	NaN	54.28
2006-01-01	54.86	NaN	54.86
2007-01-01	55.54	NaN	55.54
2008-01-01	55.03	0.01	55.03
2009-01-01	54.49	0.01	54.49
2010-01-01	56.09	0.03	56.09
2011-01-01	56.67	0.03	56.67
2012-01-01	57.62	0.04	57.62
2013-01-01	60.86	0.08	60.86
2014-01-01	63.99	0.11	63.99
2015-01-01	63.63	0.09	63.63
2016-01-01	62.76	0.06	62.76
2017-01-01	62.73	0.05	62.73
2018-01-01	61.83	0.02	61.83
2019-01-01	57.51	-0.06	57.51
2020-01-01	54.70	-0.10	54.70
2021-01-01	54.25	-0.10	54.25

Transferring back is a possibility. Therefore, I can move forward. I'll again check for stationarity.

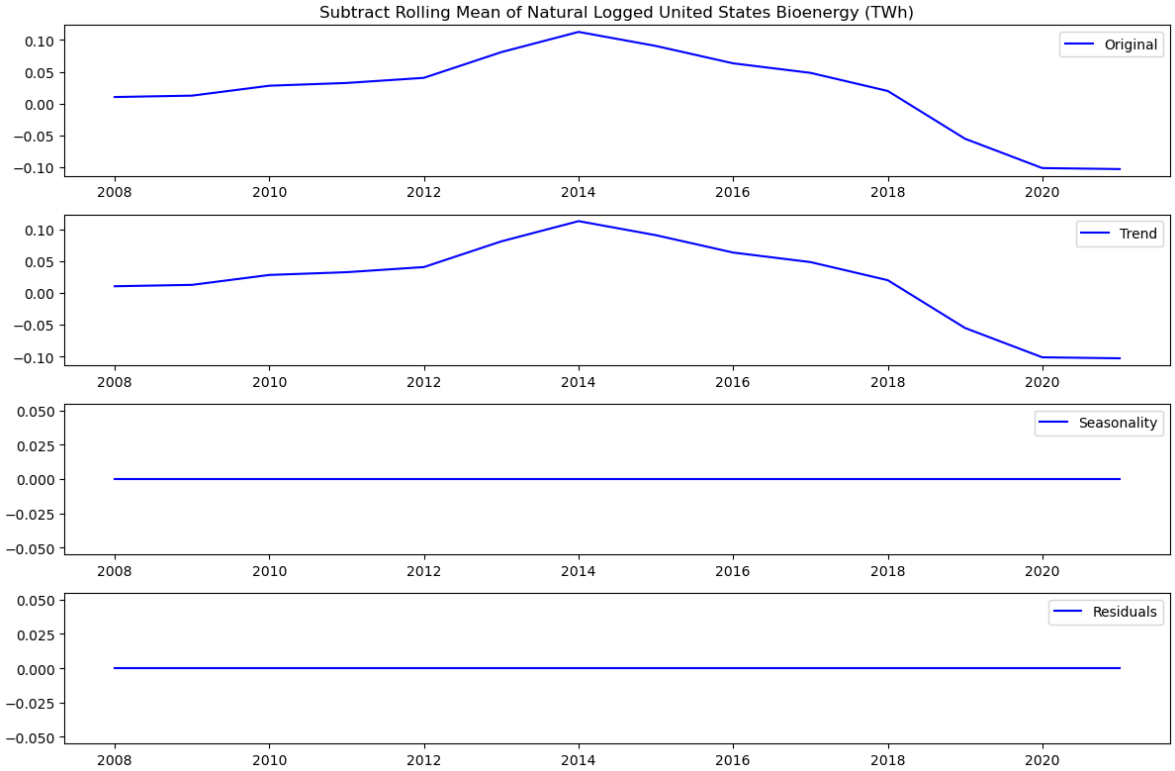
In [145]:
```
stationarity_check(model_data_t.iloc[:,11:12], s=54)
```

```
Subtract Rolling Mean of Natural Logged United States Bioenergy (TWh)
Results of Dickey-Fuller test:

Test Statistic                -2.91
p-value                        0.04
#Lags Used                     4.00
Number of Observations Used    9.00
Critical Value (1%)           -4.47
Critical Value (5%)           -3.29
Critical Value (10%)          -2.77
dtype: float64
```



My p-value is less than .05. Stationarity achieved. I'll look at the decomposition graph again.

In [146]:
```
decomp_graph(model_data_t.iloc[:,11:12].dropna())
```

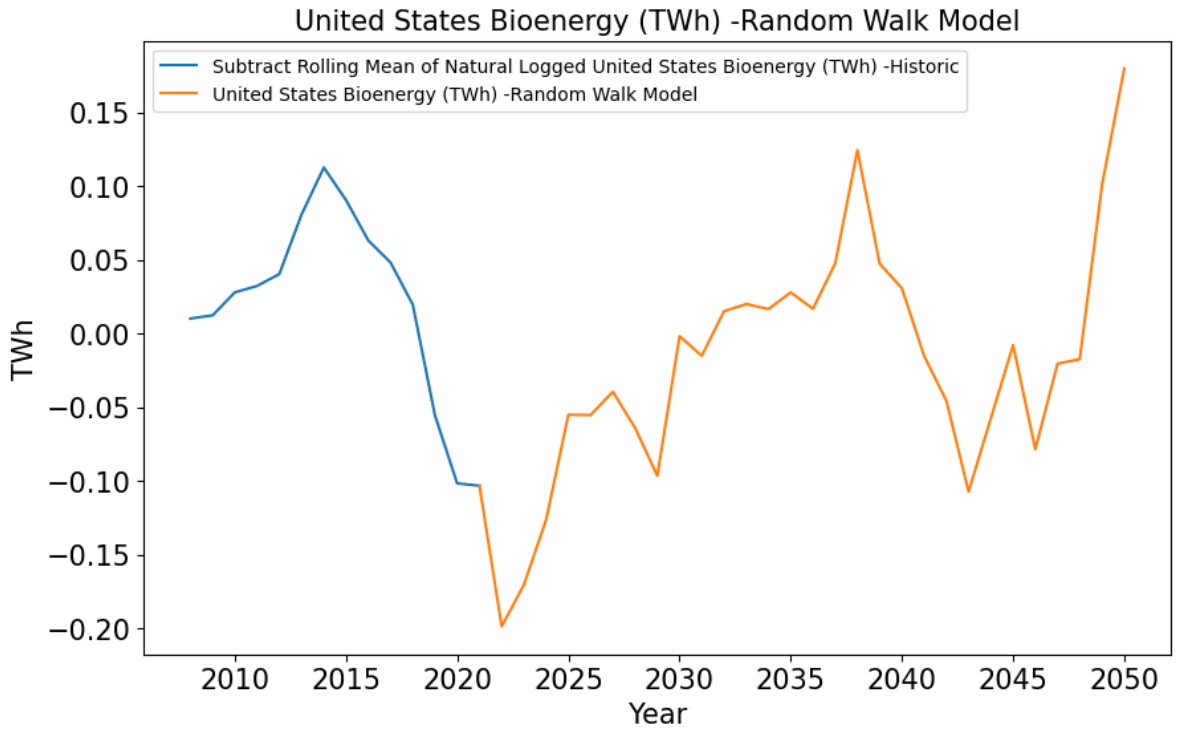


Looks like there's still a trend line, but it does pass stationarity, so I'm moving forward to the baseline random walk model.

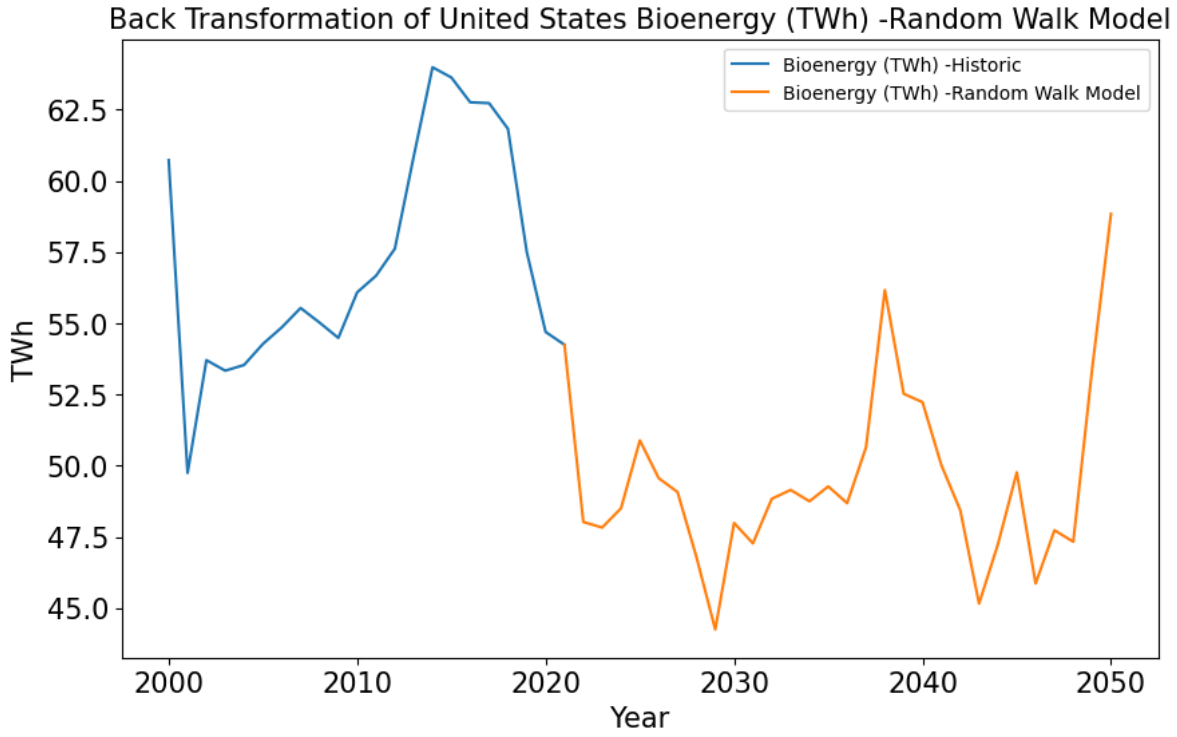
7.6.4 Random Walk Model

In [147]:
```
pd.set_option('display.max_rows',None)
y_hat_proj = random_walk(model_data_t.iloc[:,11:12])
y_hat_proj.columns = [model_data.columns[5]+' -Random Walk Model']

# Plot the transformed Random Walk
pred_graph(model_data_t.iloc[:,11:12], y_hat_proj.iloc[:,0:1])
```

```
Random Walk with Standard Deviation of Transformed Data set: 0.06
```



In [148]:
```
# Prepare the DataFrame for Back Transformation
ranplot = model_data_t.iloc[:,10:12].copy()
ranplot.rename(columns={str(ranplot.columns[1]): str(y_hat_proj.columns[0])},
               inplace=True)

ranplot = pd.concat([ranplot,y_hat_proj],axis=0)

# Perform Back Transformation
y_hat_proj = inv_log_inv_sub(ranplot, n=9)

#Plot the new graph
pred_graph(model_data_t.iloc[:,10:11], y_hat_proj.iloc[22:,0:1], 14)
```

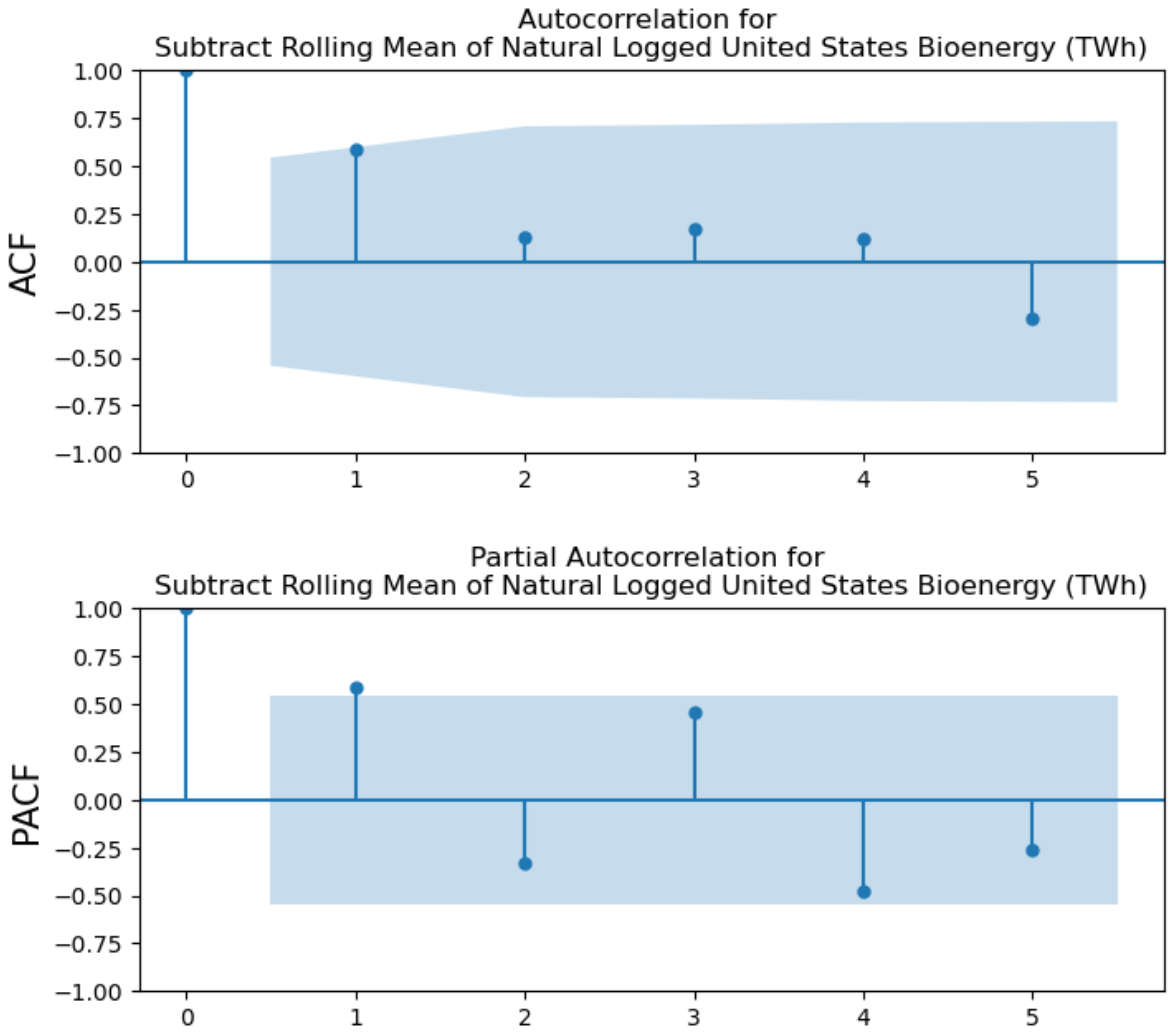


This model is as unreliable as it's name would imply, random walk. This one works ok because the standard deviation is incredibly small.

Now I'll move forward with an ARMA model. First, I need to evaluate the ACF and PACF plots to find the best Autoregressive and Moving Average terms.

7.6.5 Evaluating Initial AR and MA Values with ACF and PACF Plots

In [149]:
```
plotacf(model_data_t.iloc[:,11:12], lags = 5)
plotpacf(model_data_t.iloc[:,11:12], lags = 5)
```

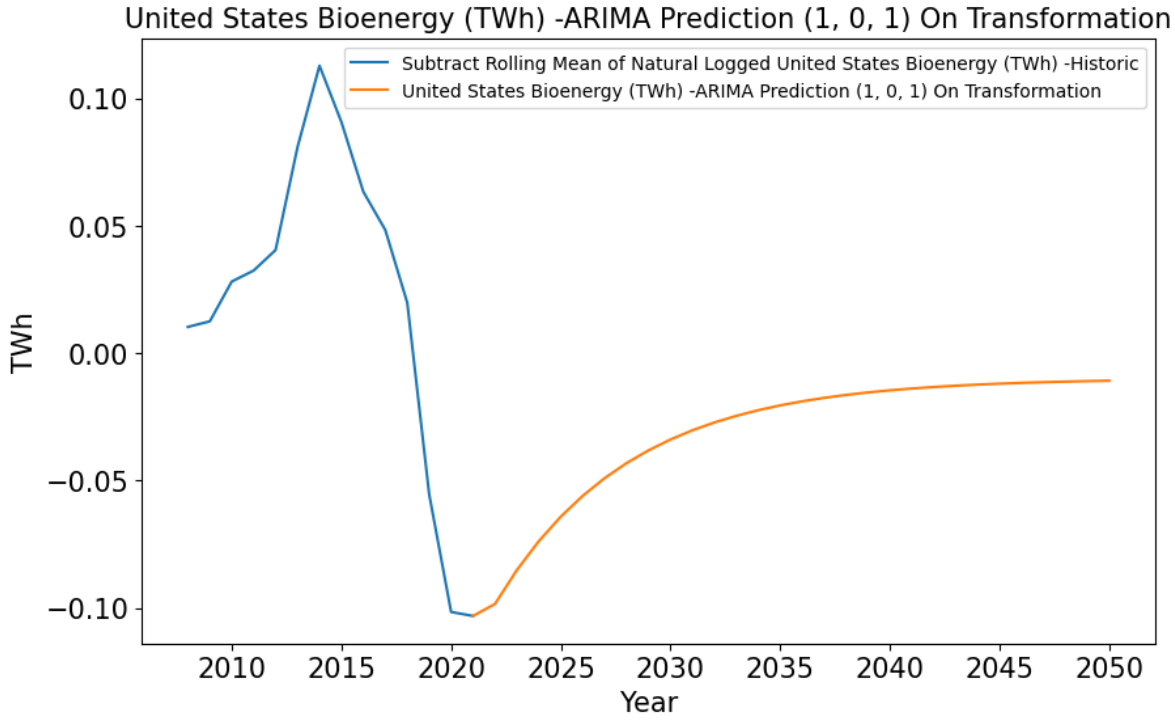


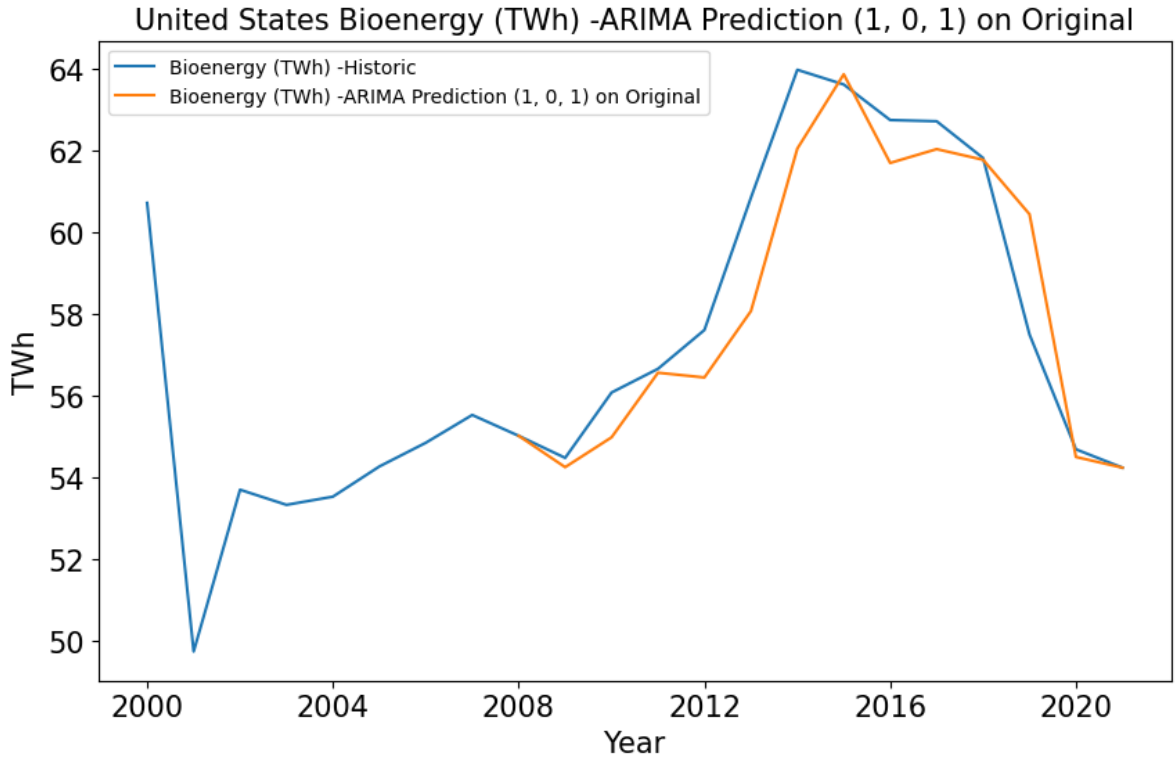
The ACF and PACF both breach at 1. Also, PACF nearly breaches at 3. I'll try (1,0,1) and (3,0,1) for my initial ARMA models.

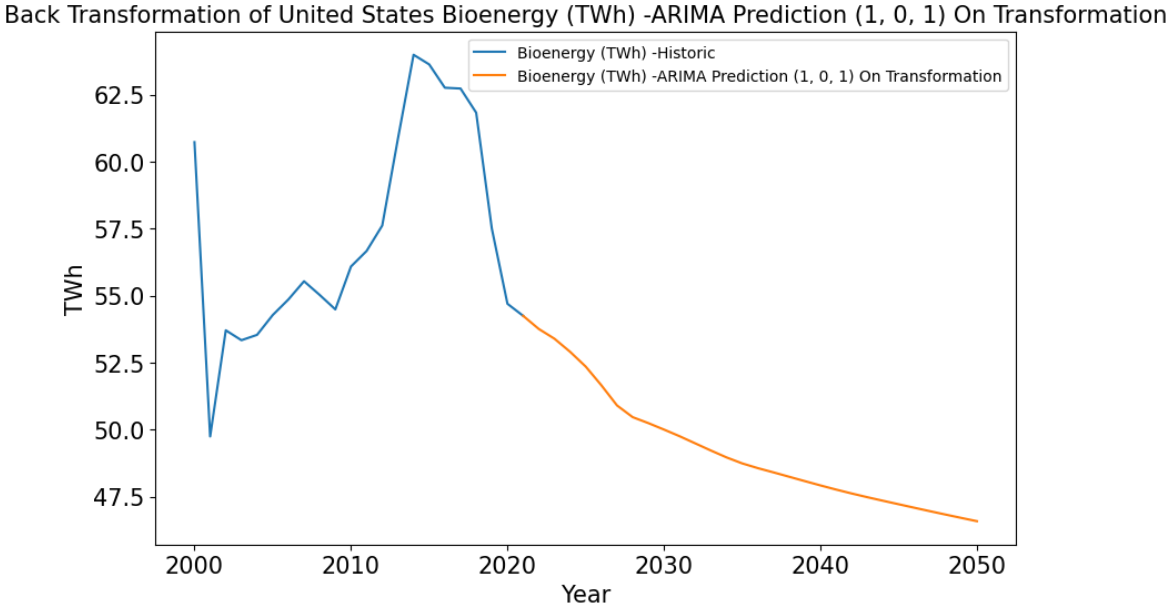
7.6.6 ARMA Model

In [150]:
```
order = (1,0,1)
model_summary(model_data_t.iloc[:,10:12], order, 22, 50, 14,
          tran = 'inv_log_inv_sub', n=9)
```

```
ARIMA: (1, 0, 1), RMSE=1.37, AIC=-57.12
**********************************************************************
**
United States Bioenergy (TWh) model results.

                                                           SARIMAX Results
==========================================================================
==================================================
Dep. Variable:     Subtract Rolling Mean of Natural Logged United States Bioe
nergy (TWh)   No. Observations:                   22
Model:                                                                 ARI
MA(1, 0, 1)   Log Likelihood                32.562
Date:                                                                Sat,
29 Apr 2023   AIC                           -57.124
Time:
06:21:36   BIC                           -52.760
Sample:
01-01-2000   HQIC                          -56.096
                                                                       -
01-01-2021
Covariance Type:
opg
==========================================================================
=
                 coef    std err          z      P>|z|      [0.025      0.97
5]
--------------------------------------------------------------------------
-
const         -0.0099      0.094     -0.105      0.916      -0.195       0.17
5
ar.L1          0.8494      0.199      4.265      0.000       0.459       1.24
0
ma.L1          0.8304      0.625      1.328      0.184      -0.395       2.05
6
sigma2         0.0004      0.000      2.428      0.015    8.39e-05       0.00
1
==========================================================================
======
Ljung-Box (L1) (Q):                   3.05   Jarque-Bera (JB):
25.57
Prob(Q):                              0.08   Prob(JB):
0.00
Heteroskedasticity (H):                inf   Skew:
-1.31
Prob(H) (two-sided):                  0.00   Kurtosis:
7.58
==========================================================================
======

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (comple
x-step).
```


United States Bioenergy (TWh) -ARIMA Prediction (1, 0, 1) On Transformation
Subtract Rolling Mean of Natural Logged United States Bioenergy (TWh) -Historic
United States Bioenergy (TWh) -ARIMA Prediction (1, 0, 1) On Transformation
0.10
0.05
0.00
−0.05
−0.10
TWh
2010
2015
2020
2025
2030
2035
2040
2045
2050
Year

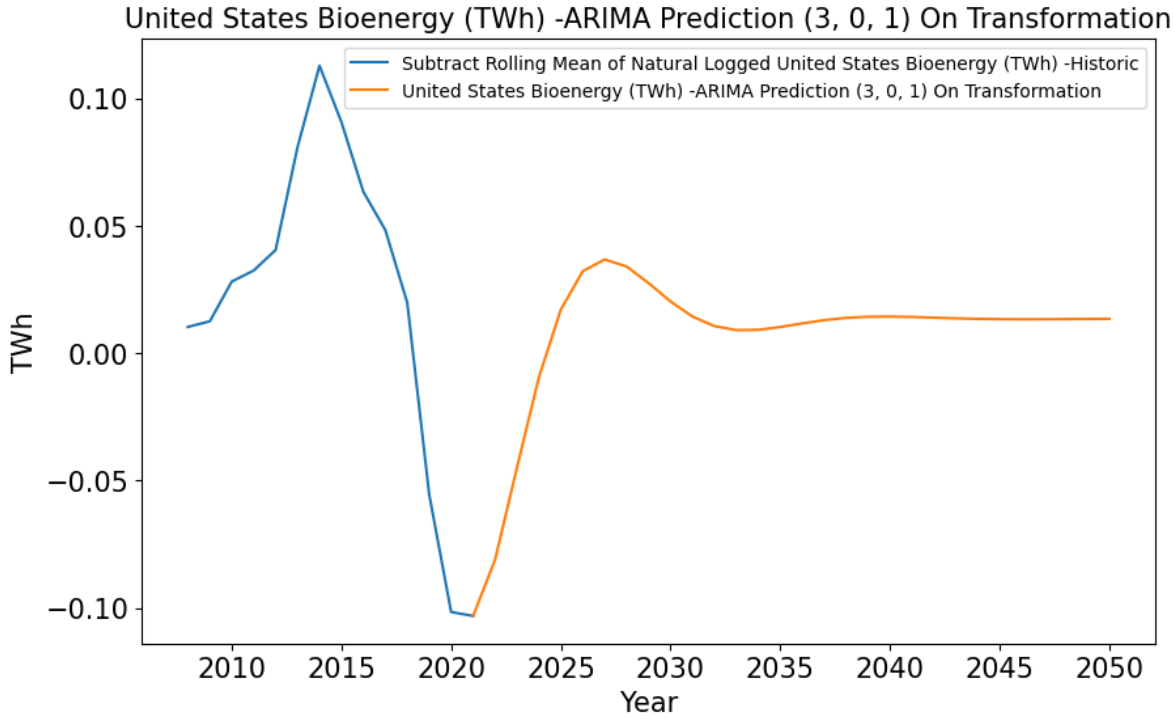

United States Bioenergy (TWh) -ARIMA Prediction (1, 0, 1) on Original
Bioenergy (TWh) -Historic
Bioenergy (TWh) -ARIMA Prediction (1, 0, 1) on Original
64
62
60
58
56
54
52
50
TWh
2000
2004
2008
2012
2016
2020
Year

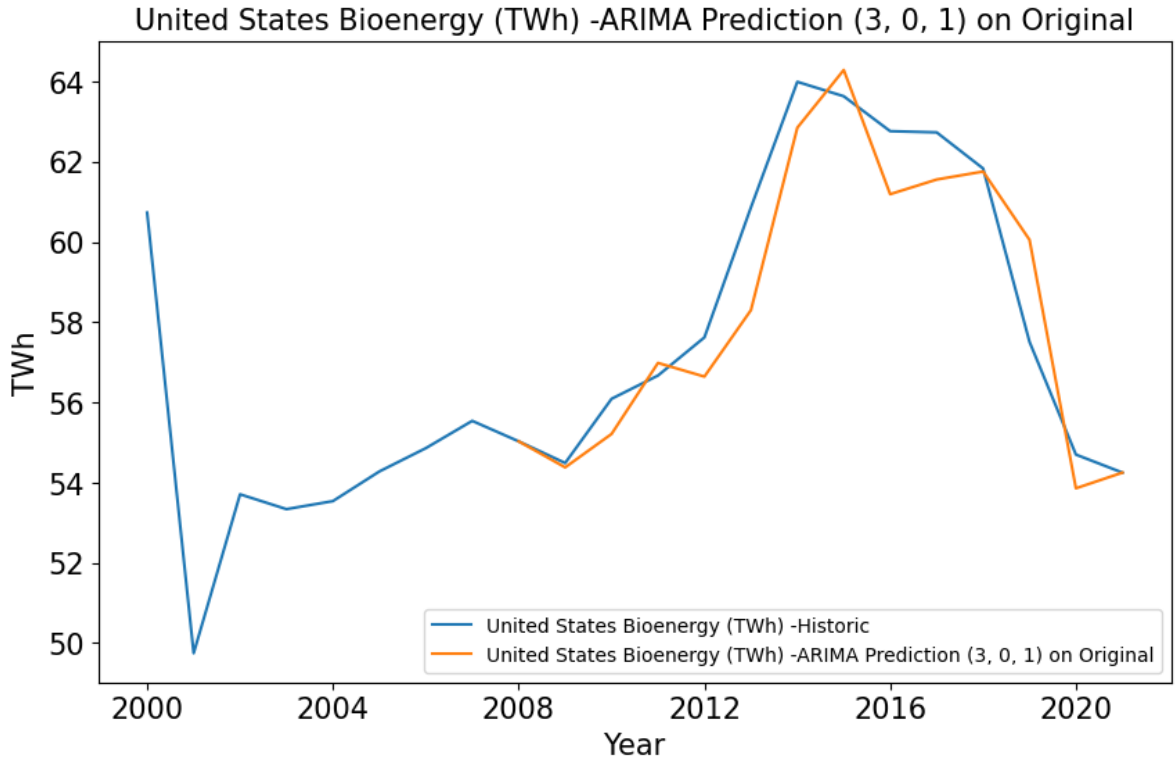

Back Transformation of United States Bioenergy (TWh) -ARIMA Prediction (1, 0, 1) On Transformation
Bioenergy (TWh) -Historic
Bioenergy (TWh) -ARIMA Prediction (1, 0, 1) On Transformation
TWh
62.5
60.0
57.5
55.0
52.5
50.0
47.5
2000
2010
2020
2030
2040
2050
Year

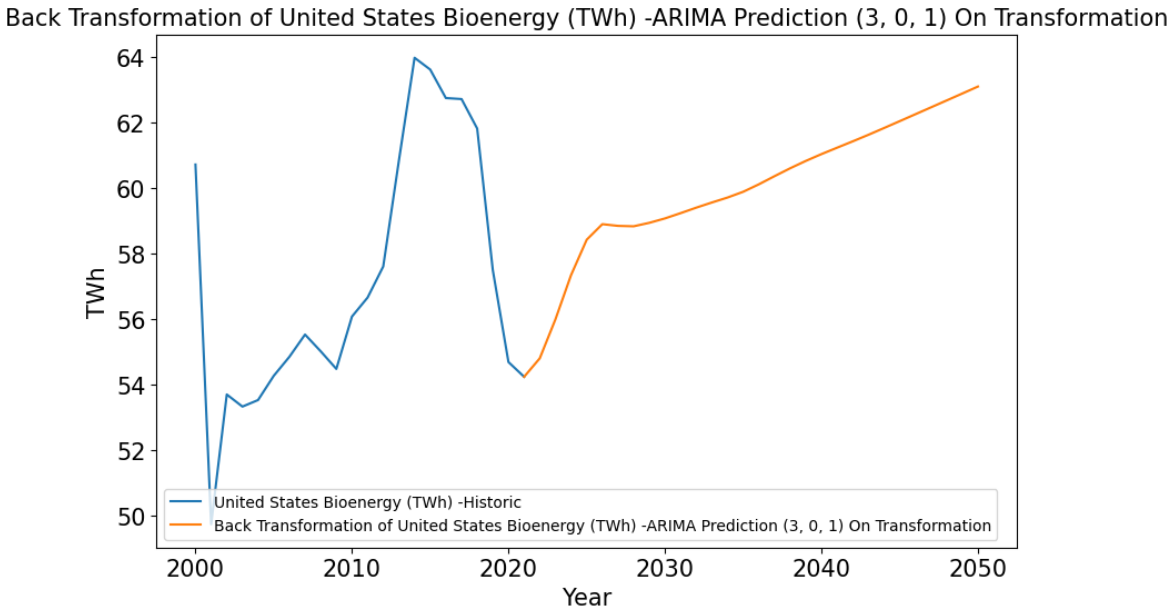
In [151]:
```
order = (3,0,1)
model_summary(model_data_t.iloc[:,10:12], order, 22, 50, 0,
          tran = 'inv_log_inv_sub', n=9)
```

```
ARIMA: (3, 0, 1), RMSE=1.28, AIC=-55.67
**********************************************************************
**
United States Bioenergy (TWh) model results.

                                                           SARIMAX Results
==========================================================================
==================================================
Dep. Variable:     Subtract Rolling Mean of Natural Logged United States Bioe
nergy (TWh)   No. Observations:                   22
Model:                                                                 ARI
MA(3, 0, 1)   Log Likelihood                33.836
Date:                                                              Sat,
29 Apr 2023   AIC                           -55.671
Time:
06:21:37   BIC                           -49.125
Sample:
01-01-2000   HQIC                          -54.129
                                                                       -
01-01-2021
Covariance Type:
opg
==========================================================================
=
                 coef    std err          z      P>|z|      [0.025      0.97
5]
--------------------------------------------------------------------------
-
const          0.0135      0.066      0.204      0.839      -0.116       0.14
3
ar.L1          1.2091      1.439      0.840      0.401      -1.611       4.02
9
ar.L2         -0.3907      2.361     -0.165      0.869      -5.019       4.23
7
ar.L3         -0.0911      1.385     -0.066      0.948      -2.805       2.62
3
ma.L1          0.6365      1.347      0.472      0.637      -2.004       3.27
7
sigma2         0.0004      0.000      1.666      0.096   -6.34e-05       0.00
1
==========================================================================
======
Ljung-Box (L1) (Q):                   0.22   Jarque-Bera (JB):
19.84
Prob(Q):                              0.64   Prob(JB):
0.00
Heteroskedasticity (H):                inf   Skew:
-1.13
Prob(H) (two-sided):                  0.00   Kurtosis:
7.07
==========================================================================
======

Warnings:
```

```
[1] Covariance matrix calculated using the outer product of gradients (comple
x-step).
```





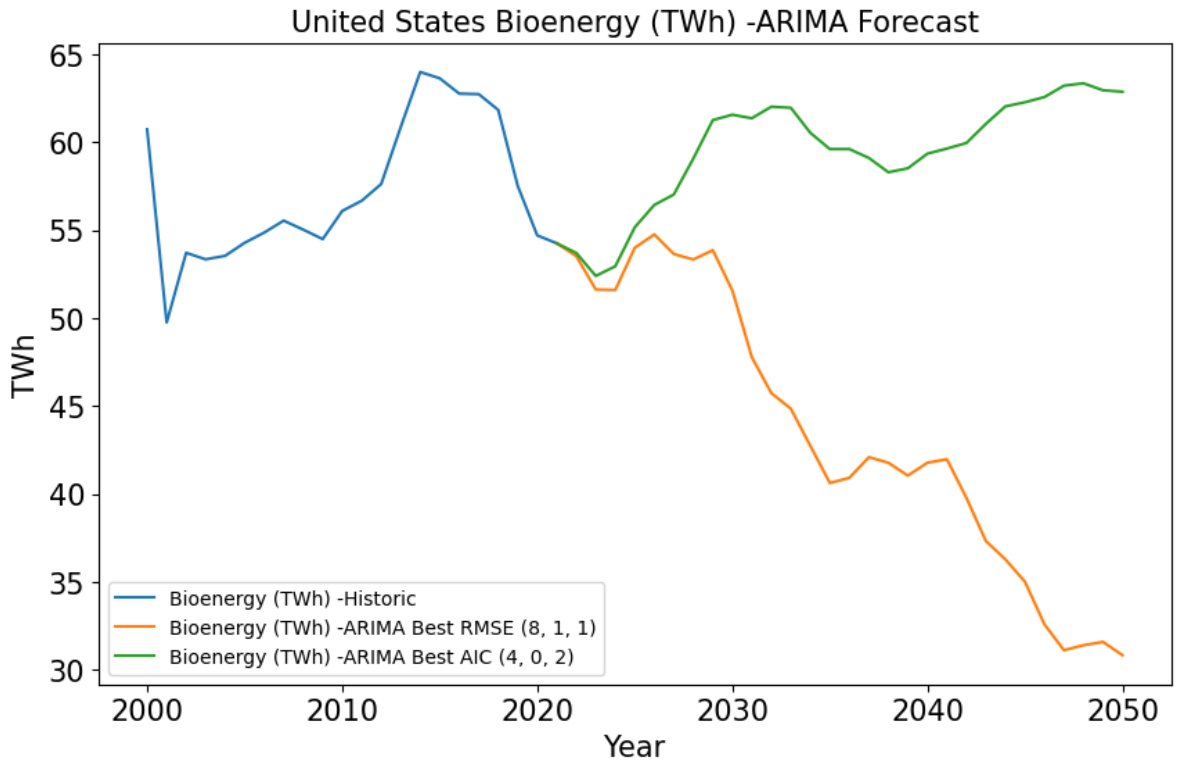


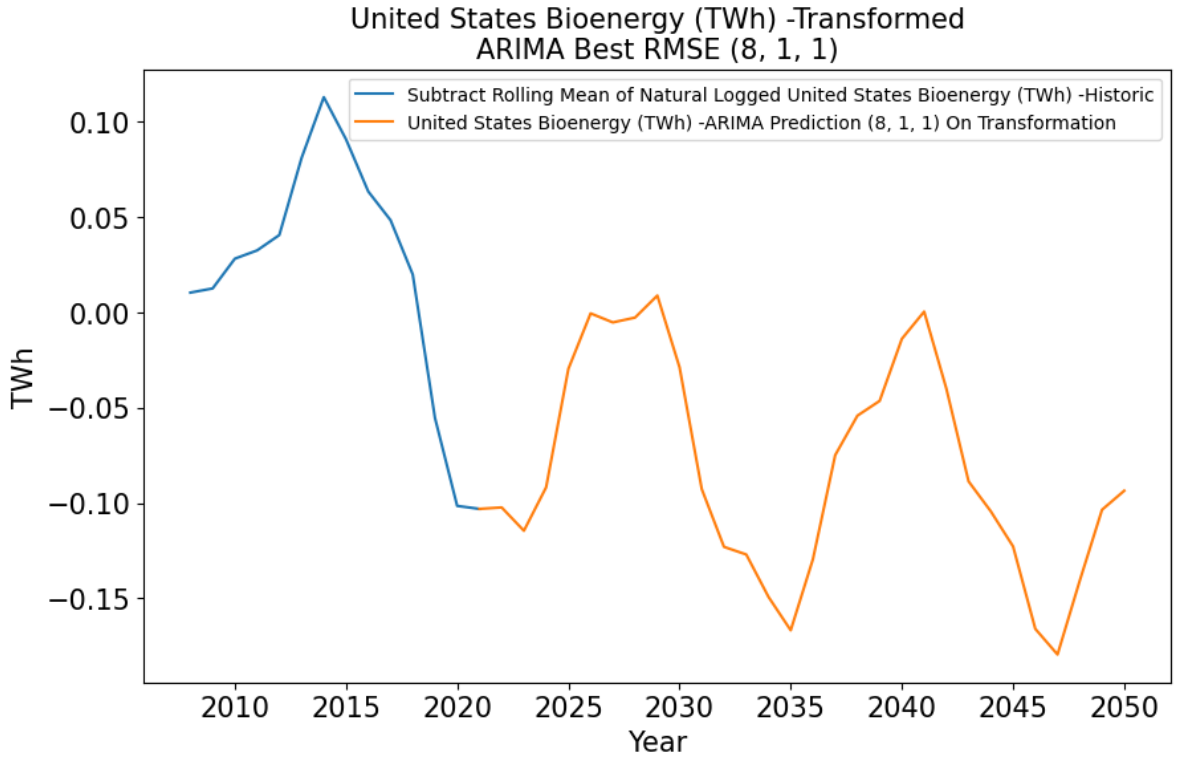
The first shows a slight decline, the second a slight incline. I'll have to see what Grid Search says.

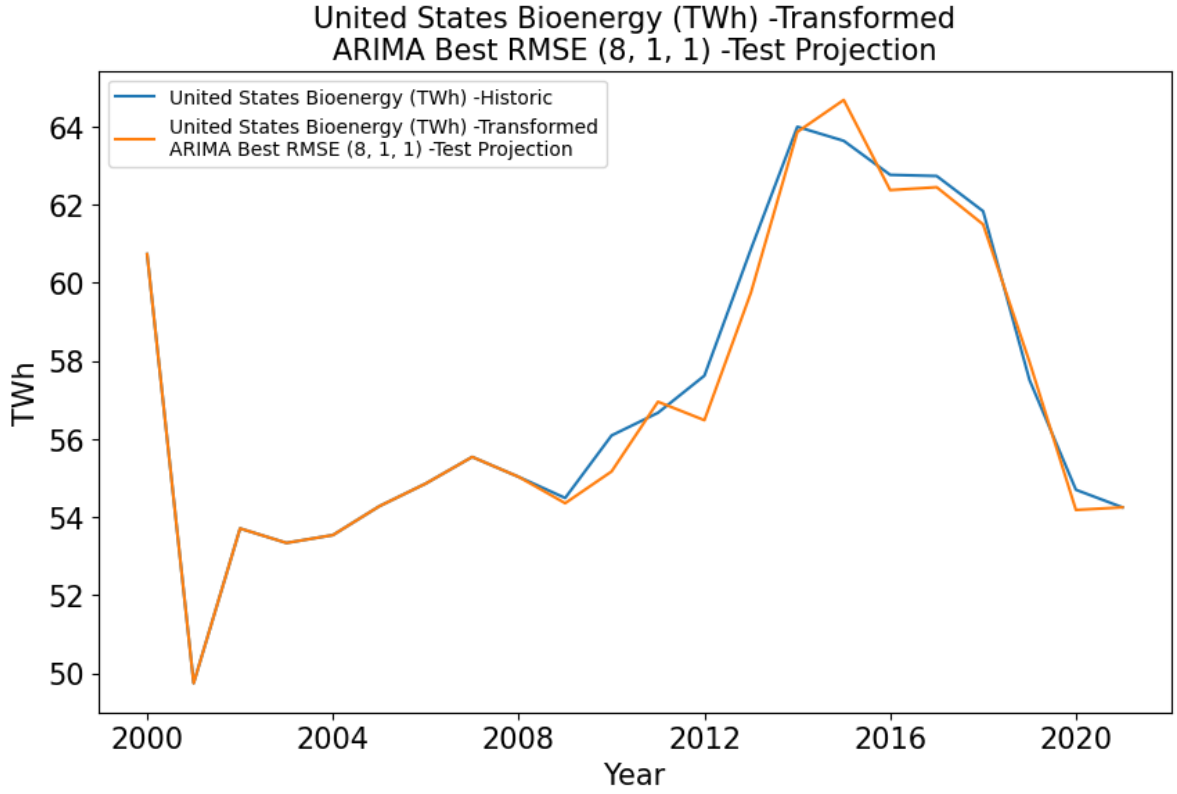
7.6.7 ARIMA Model and Grid Search

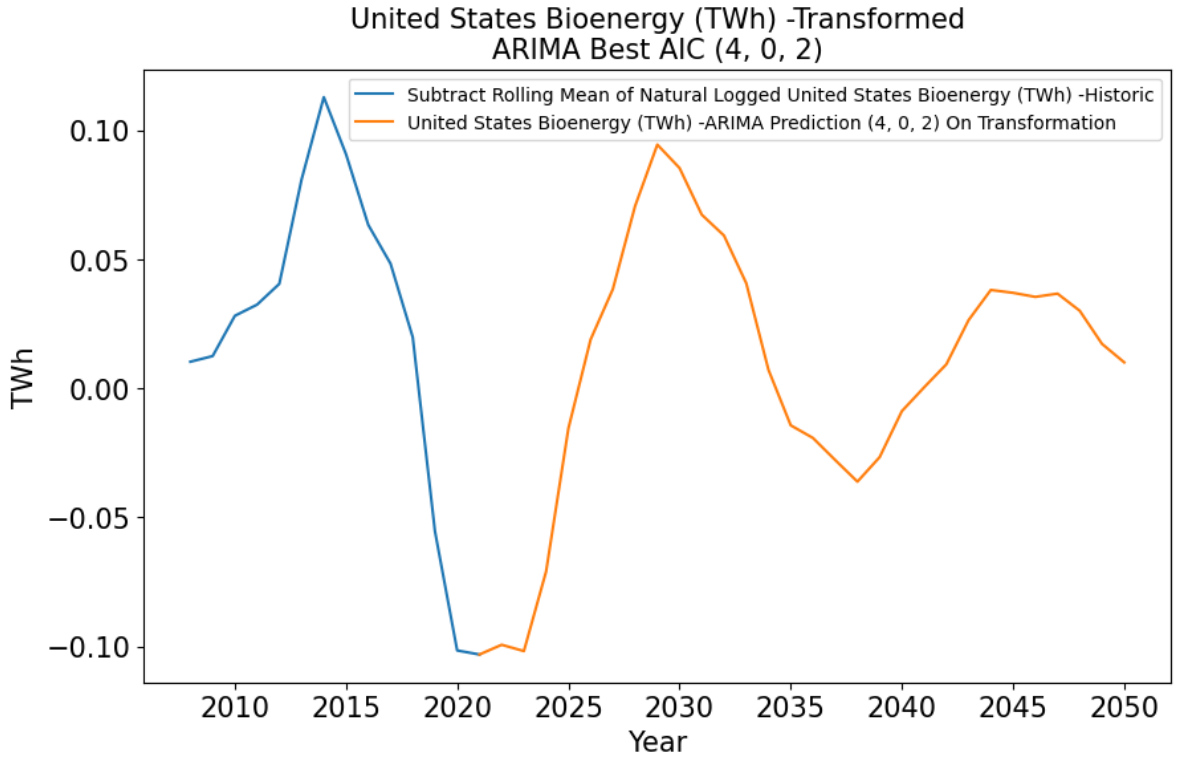
In [152]:
```
bio_rmse_cfg, oil_aic_cfg = arima_pdq(
            model_data_t.iloc[:,10:12], s=14, tran='inv_log_inv_sub', n=9)
```

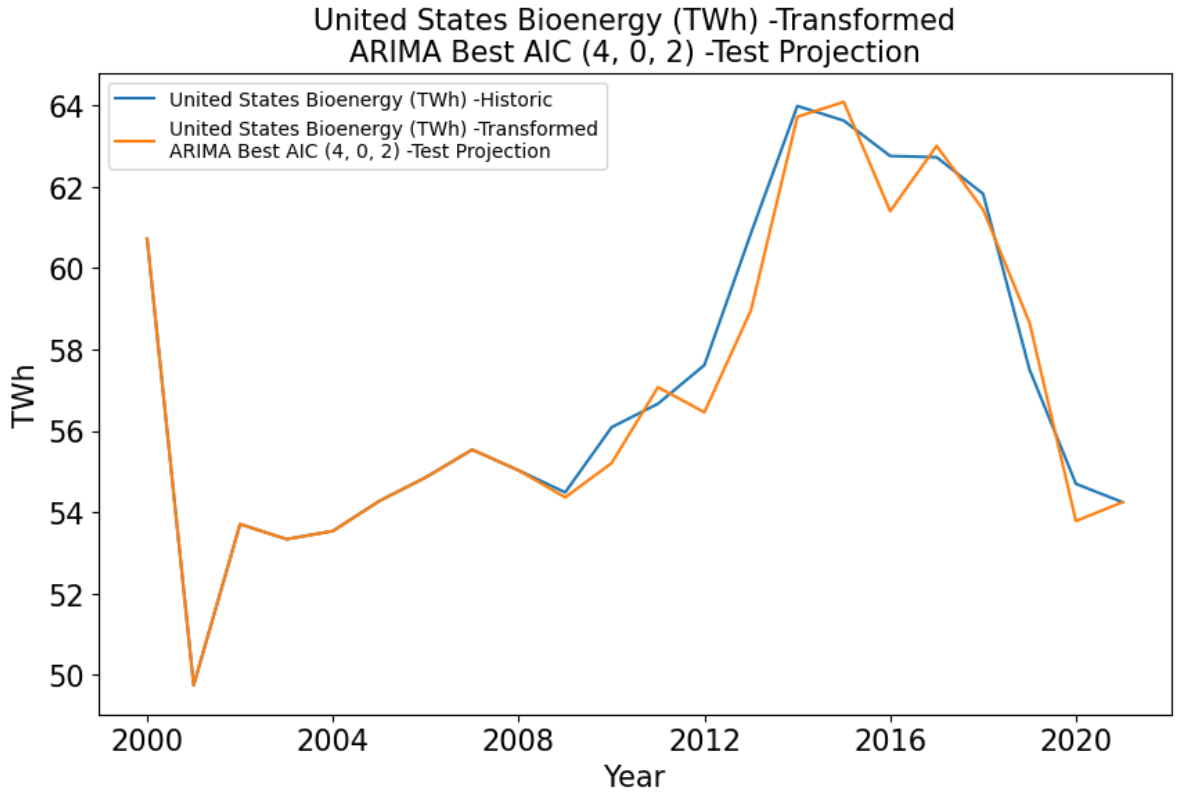
```
RMSE ARIMA: (0, 0, 0), RMSE= 4.14, AIC= -33.47, TWh-2050= 68.12
RMSE ARIMA: (0, 0, 1), RMSE= 2.52, AIC= -45.13, TWh-2050= 64.99
RMSE ARIMA: (0, 0, 2), RMSE= 1.66, AIC= -54.27, TWh-2050= 63.23
RMSE ARIMA: (0, 0, 3), RMSE= 1.28, AIC= -57.91, TWh-2050= 61.54
RMSE ARIMA: (0, 1, 2), RMSE= 1.20, AIC= -42.74, TWh-2050= 31.06
RMSE ARIMA: (0, 1, 3), RMSE= 1.19, AIC= -41.05, TWh-2050= 29.67
RMSE ARIMA: (0, 1, 4), RMSE= 1.17, AIC= -40.00, TWh-2050= 21.69
RMSE ARIMA: (0, 1, 5), RMSE= 1.05, AIC= -39.90, TWh-2050= 28.91
RMSE ARIMA: (0, 1, 7), RMSE= 1.00, AIC= -37.25, TWh-2050= 55.11
AIC  ARIMA: (1, 0, 2), RMSE= 1.17, AIC= -59.55, TWh-2050= 57.57
RMSE ARIMA: (1, 1, 7), RMSE= 0.98, AIC= -36.25, TWh-2050= 56.25
RMSE ARIMA: (2, 1, 7), RMSE= 0.97, AIC= -34.42, TWh-2050= 57.77
RMSE ARIMA: (3, 1, 2), RMSE= 0.95, AIC= -41.86, TWh-2050= 11.54
RMSE ARIMA: (3, 1, 4), RMSE= 0.83, AIC= -40.87, TWh-2050= 18.98
AIC  ARIMA: (4, 0, 2), RMSE= 0.89, AIC= -59.60, TWh-2050= 62.87
RMSE ARIMA: (4, 1, 4), RMSE= 0.78, AIC= -42.06, TWh-2050= 40.22
RMSE ARIMA: (6, 1, 2), RMSE= 0.72, AIC= -43.31, TWh-2050= 48.61
RMSE ARIMA: (6, 1, 5), RMSE= 0.71, AIC= -36.95, TWh-2050= 46.84
RMSE ARIMA: (7, 1, 4), RMSE= 0.69, AIC= -42.26, TWh-2050= 43.52
RMSE ARIMA: (8, 1, 0), RMSE= 0.69, AIC= -45.88, TWh-2050= 22.11
RMSE ARIMA: (8, 1, 1), RMSE= 0.65, AIC= -42.13, TWh-2050= 30.84
Best RMSE ARIMA: (8, 1, 1) RMSE= 0.65 AIC= -42.13, TWh-2050= 30.84
Best AIC  ARIMA: (4, 0, 2) RMSE= 0.89 AIC= -59.60, TWh-2050= 62.87
Graph_formatting produced error at (8, 1, 1) or (4, 0, 2)
```




United States Bioenergy (TWh) -Transformed
ARIMA Best RMSE (8, 1, 1)
Subtract Rolling Mean of Natural Logged United States Bioenergy (TWh) -Historic
United States Bioenergy (TWh) -ARIMA Prediction (8, 1, 1) On Transformation
TWh
0.10
0.05
0.00
−0.05
−0.10
−0.15
2010
2015
2020
2025
2030
2035
2040
2045
2050
Year


United States Bioenergy (TWh) -Transformed
ARIMA Best RMSE (8, 1, 1) -Test Projection
United States Bioenergy (TWh) -Historic
United States Bioenergy (TWh) -Transformed
ARIMA Best RMSE (8, 1, 1) -Test Projection
TWh
64
62
60
58
56
54
52
50
2000
2004
2008
2012
2016
2020
Year



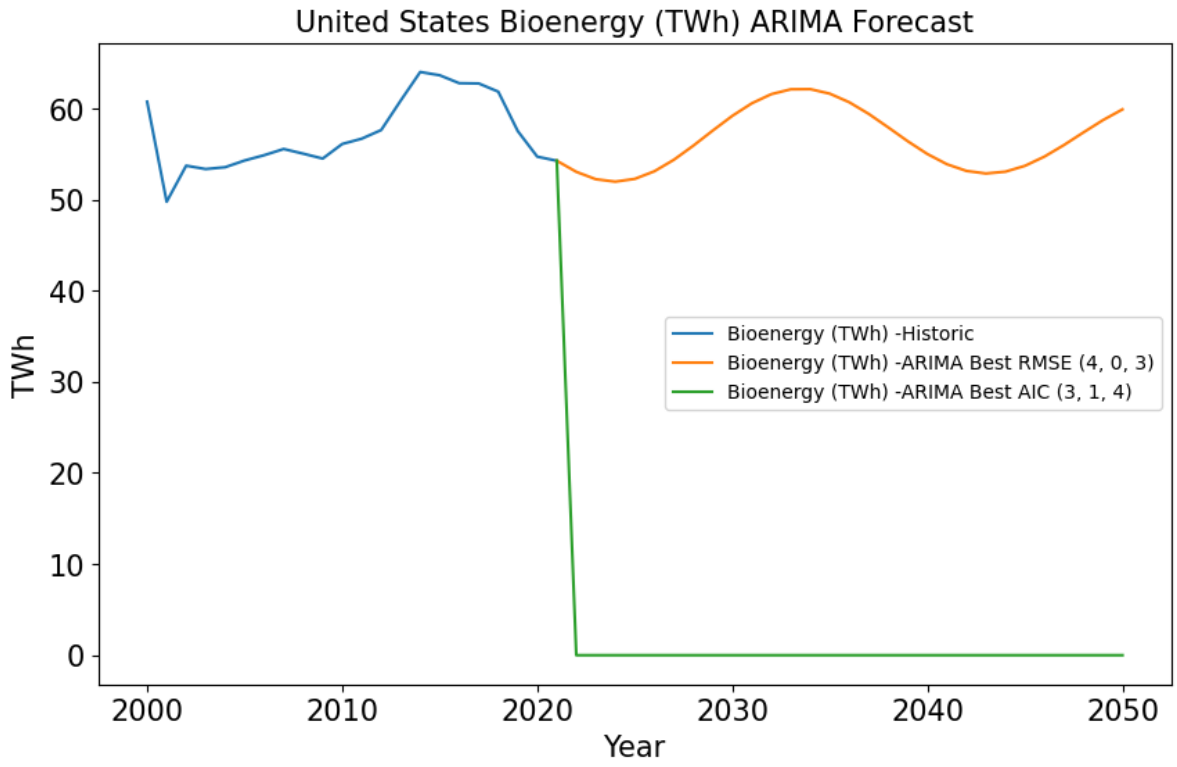


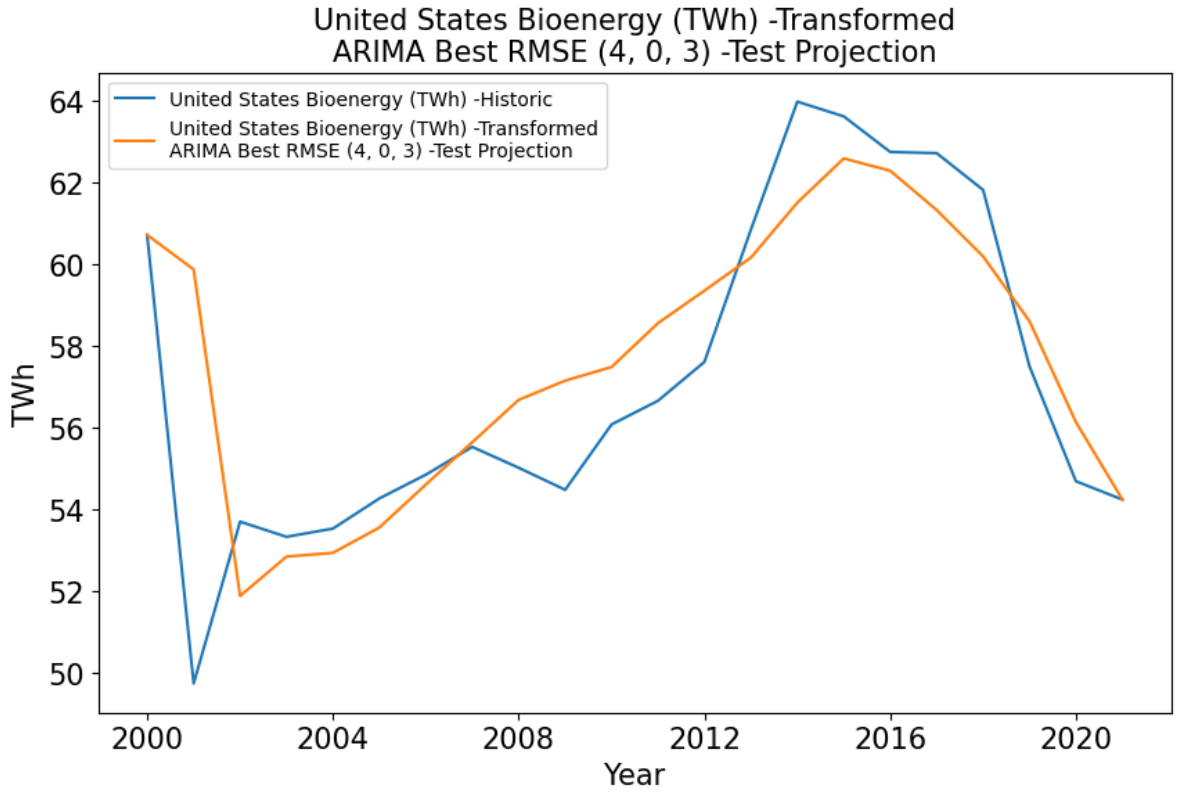
Bio energy is a dubious concept to begin with. It burns trees and replants with regularity, claiming that the new trees make it carbon neutral. Not only dubious, but egregious too. I will use the best RMSE graph for this, hoping for more of an overall decline rather than a rebound in the industry.

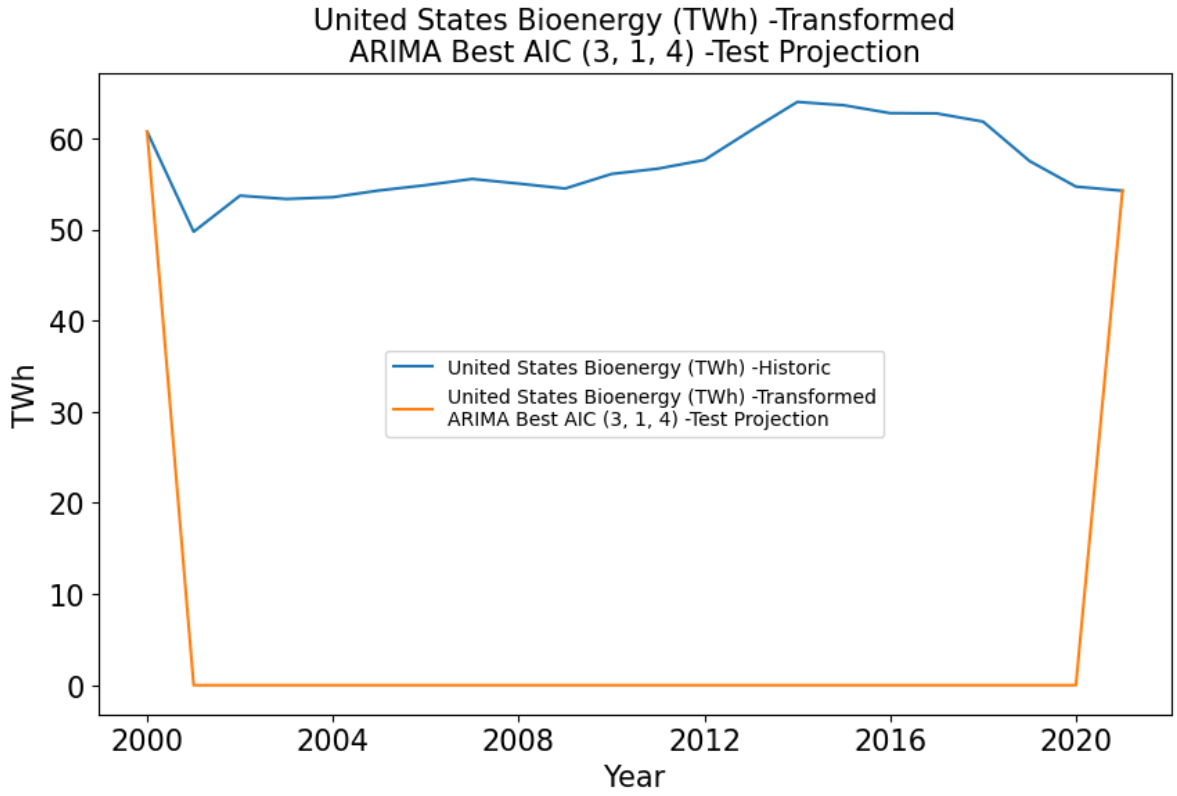
7.6.8 ARIMA Without Transformation

In [153]:
```
bio_rmse_cfg, oil_aic_cfg = arima_pdq_no_tran(
                                        model_data.iloc[:,5:6], 22, 50, s=14)
```

```
United States Bioenergy (TWh) Grid Search:
RMSE ARIMA: (0, 0, 0), RMSE= 3.79, AIC= 126.52, TWh-2050= 57.18
RMSE ARIMA: (0, 0, 1), RMSE= 3.04, AIC= 117.79, TWh-2050= 57.37
RMSE ARIMA: (0, 0, 2), RMSE= 2.88, AIC= 115.23, TWh-2050= 57.53
RMSE ARIMA: (0, 0, 3), RMSE= 2.77, AIC= 116.74, TWh-2050= 57.14
RMSE ARIMA: (0, 0, 4), RMSE= 2.74, AIC= 114.98, TWh-2050= 56.98
RMSE ARIMA: (0, 0, 7), RMSE= 2.72, AIC= 118.76, TWh-2050= 56.84
AIC  ARIMA: (0, 1, 0), RMSE= 2.94, AIC= 107.94, TWh-2050= 54.25
AIC  ARIMA: (0, 2, 2), RMSE= 9.59, AIC= 107.27, TWh-2050= 11.84
RMSE ARIMA: (1, 0, 0), RMSE= 2.70, AIC= 113.89, TWh-2050= 57.23
RMSE ARIMA: (1, 0, 3), RMSE= 2.63, AIC= 115.52, TWh-2050= 56.85
RMSE ARIMA: (1, 0, 7), RMSE= 2.63, AIC= 120.33, TWh-2050= 57.31
RMSE ARIMA: (1, 0, 8), RMSE= 2.55, AIC= 120.77, TWh-2050= 57.02
AIC  ARIMA: (3, 1, 4), RMSE= 54.62, AIC= 16.00, TWh-2050= 0.00
RMSE ARIMA: (4, 0, 3), RMSE= 2.54, AIC= 118.26, TWh-2050= 59.88
Best RMSE ARIMA: (4, 0, 3) RMSE= 2.54 AIC= 118.26, TWh-2050= 59.88
Best AIC  ARIMA: (3, 1, 4) RMSE= 54.62 AIC= 16.00, TWh-2050= 0.00
```




United States Bioenergy (TWh) -Transformed
ARIMA Best RMSE (4, 0, 3) -Test Projection
United States Bioenergy (TWh) -Historic
United States Bioenergy (TWh) -Transformed
ARIMA Best RMSE (4, 0, 3) -Test Projection
TWh
Year
64
62
60
58
56
54
52
50
2000
2004
2008
2012
2016
2020


United States Bioenergy (TWh) -Transformed
ARIMA Best AIC (3, 1, 4) -Test Projection
United States Bioenergy (TWh) -Historic
United States Bioenergy (TWh) -Transformed
ARIMA Best AIC (3, 1, 4) -Test Projection
TWh
Year
60
50
40
30
20
10
0
2000
2004
2008
2012
2016
2020

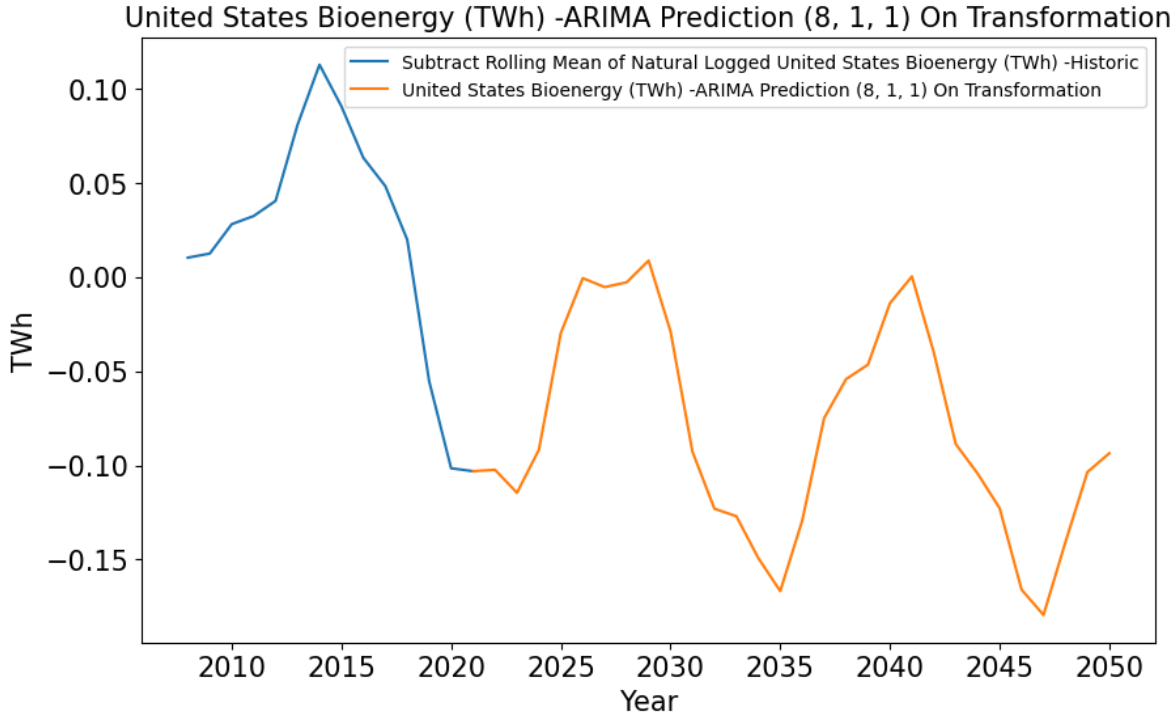
7.6.9 Model Selection

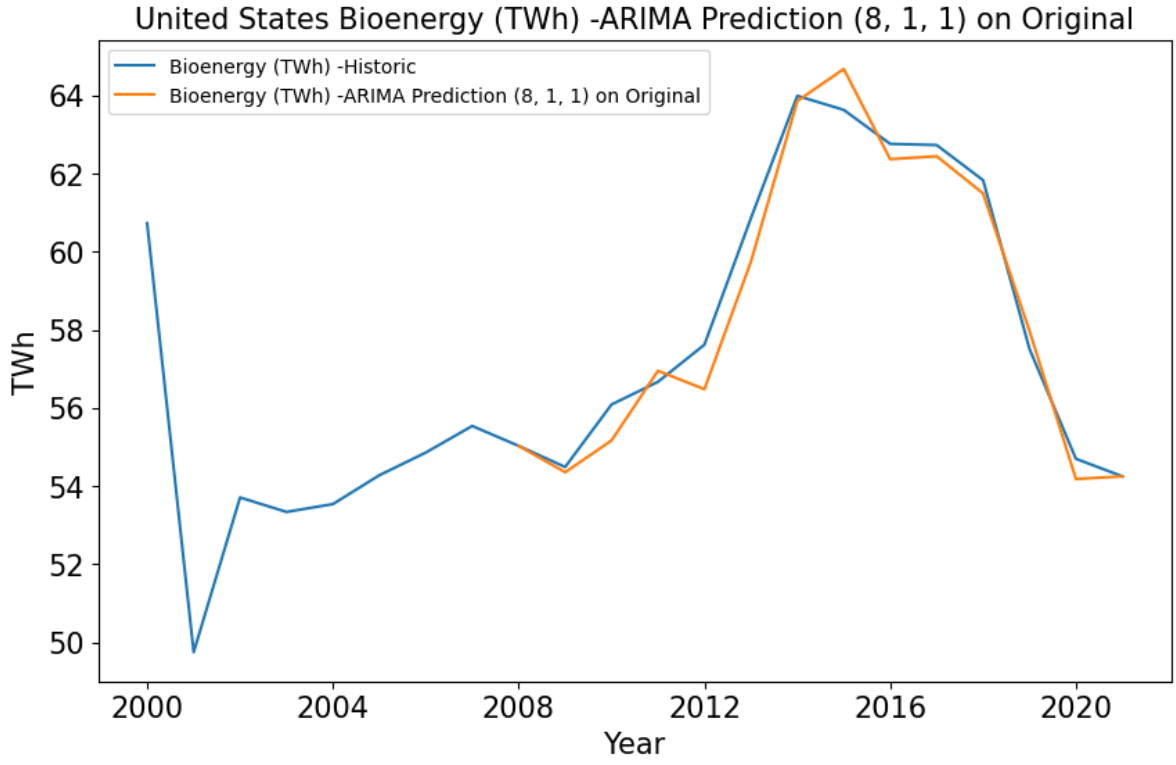
In [154]:
```
#Deleteme
stored_bio = selected_models.copy()
```

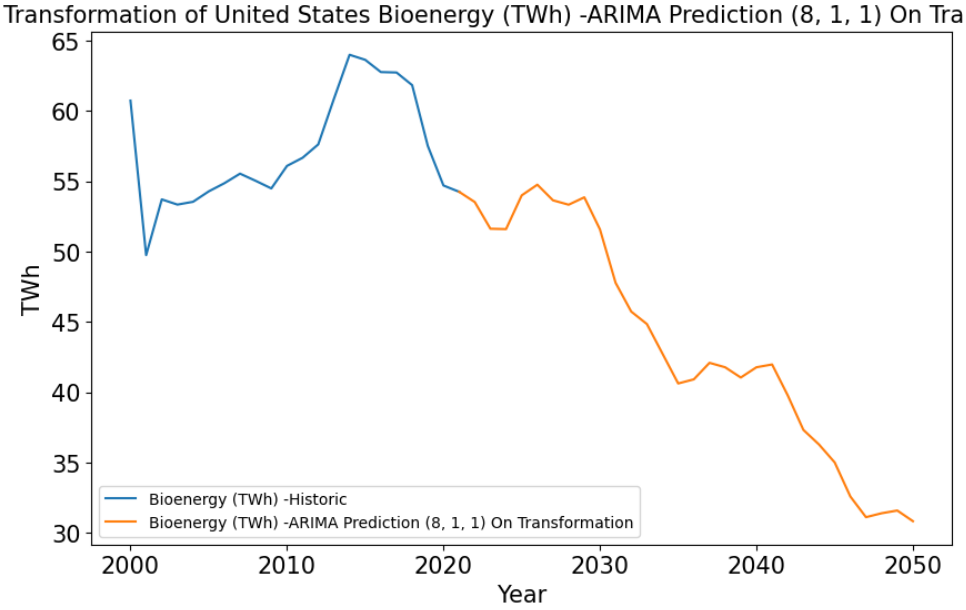
In [155]:
```
selected_models = stored_bio.copy() #deleteme
df_bio = model_data_t.iloc[:,10:12]

#                        Model,       df_o,  pdq,    tran,             n,PDQs
selected_models.append(['ARIMA_tran', df_bio,(8,1,1),'inv_log_inv_sub',9,None]

m = 5
model_summary(selected_models[m][1], selected_models[m][2], 22, 50, 14,
              selected_models[m][3], selected_models[m][4])
```

```
ARIMA: (8, 1, 1), RMSE=0.65, AIC=-42.13
**********************************************************************
**
United States Bioenergy (TWh) model results.

                                                           SARIMAX Results
==========================================================================
==================================================
Dep. Variable:     Subtract Rolling Mean of Natural Logged United States Bioe
nergy (TWh)   No. Observations:                   22
Model:                                                                 ARI
MA(8, 1, 1)   Log Likelihood                31.064
Date:                                                              Sat,
29 Apr 2023   AIC                           -42.129
Time:
06:22:52   BIC                           -31.683
Sample:
01-01-2000   HQIC                          -39.862
                                                                       -
01-01-2021
Covariance Type:
opg
==========================================================================
=
                 coef    std err          z      P>|z|      [0.025      0.97
5]
--------------------------------------------------------------------------
-
ar.L1          0.1624     40.313      0.004      0.997     -78.850      79.17
4
ar.L2          0.8122     37.550      0.022      0.983     -72.785      74.40
9
ar.L3         -0.1176     12.547     -0.009      0.993     -24.710      24.47
5
ar.L4          0.2580      8.294      0.031      0.975     -15.998      16.51
4
ar.L5         -0.3797     12.387     -0.031      0.976     -24.658      23.89
9
ar.L6         -0.9363     20.889     -0.045      0.964     -41.877      40.00
5
ar.L7          0.4163     31.715      0.013      0.990     -61.743      62.57
6
ar.L8          0.4795     41.277      0.012      0.991     -80.422      81.38
1
ma.L1          0.8826     28.332      0.031      0.975     -54.647      56.41
2
sigma2      3.296e-05      0.001      0.058      0.953      -0.001       0.00
1
==========================================================================
======
Ljung-Box (L1) (Q):                   0.28   Jarque-Bera (JB):
25.84
Prob(Q):                              0.59   Prob(JB):
0.00
Heteroskedasticity (H):                inf   Skew:
-1.91
```

```
Prob(H) (two-sided):                  0.00   Kurtosis:
6.87
================================================================================
======

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (comple
x-step).
```







This is the best RMSE, and I think it's reasonable to say a slight decline will happen in the next few years.

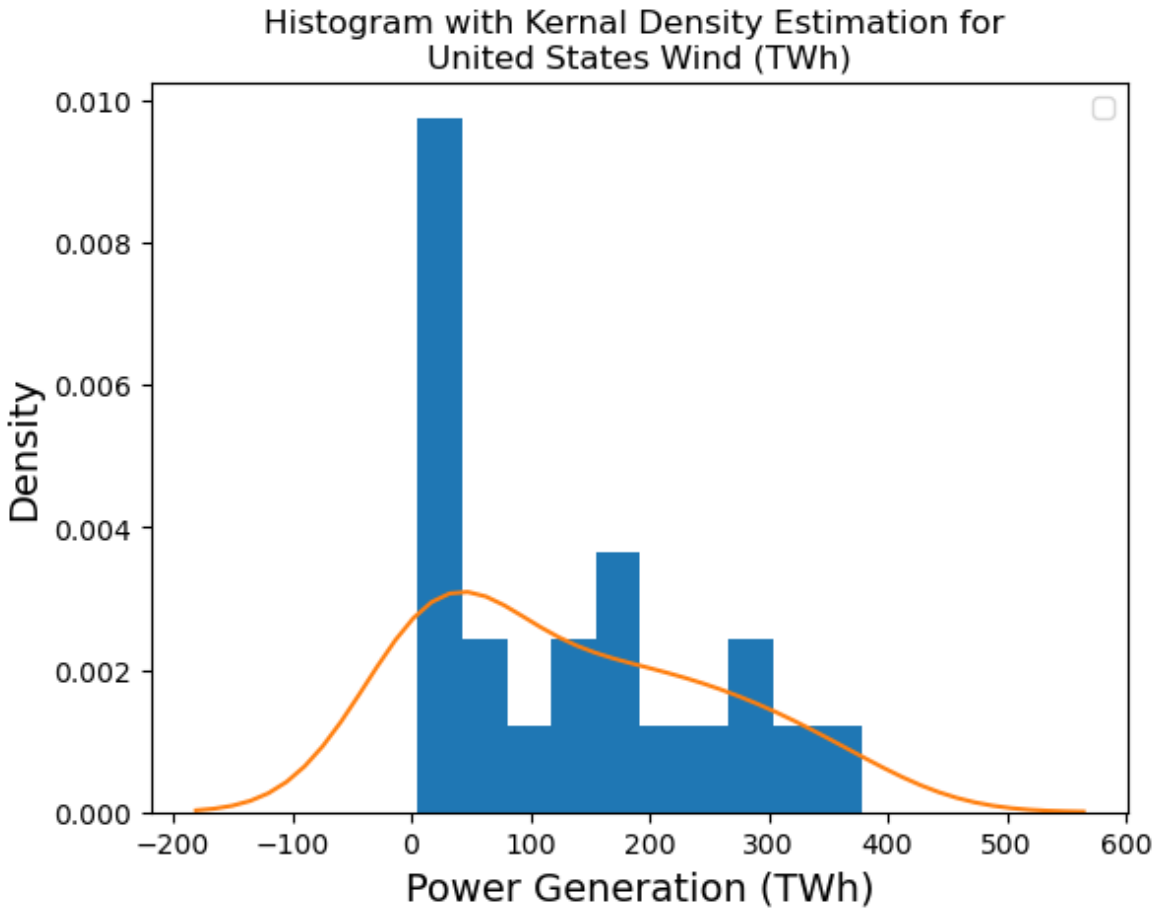
7.7 Wind

7.7.1 Distribution Investigation

In [156]:
```
stored_model_data = model_data.copy()
stored_model_data_t = model_data_t.copy()
```

In [157]:
```
model_data = stored_model_data.copy()
model_data_t = stored_model_data_t.copy()
```

In [158]:
```
hist(model_data.iloc[:,6:7])
```



In [159]:
```
stats_block = s_block(model_data.iloc[:,6:7], stats_block)
stats_block
```

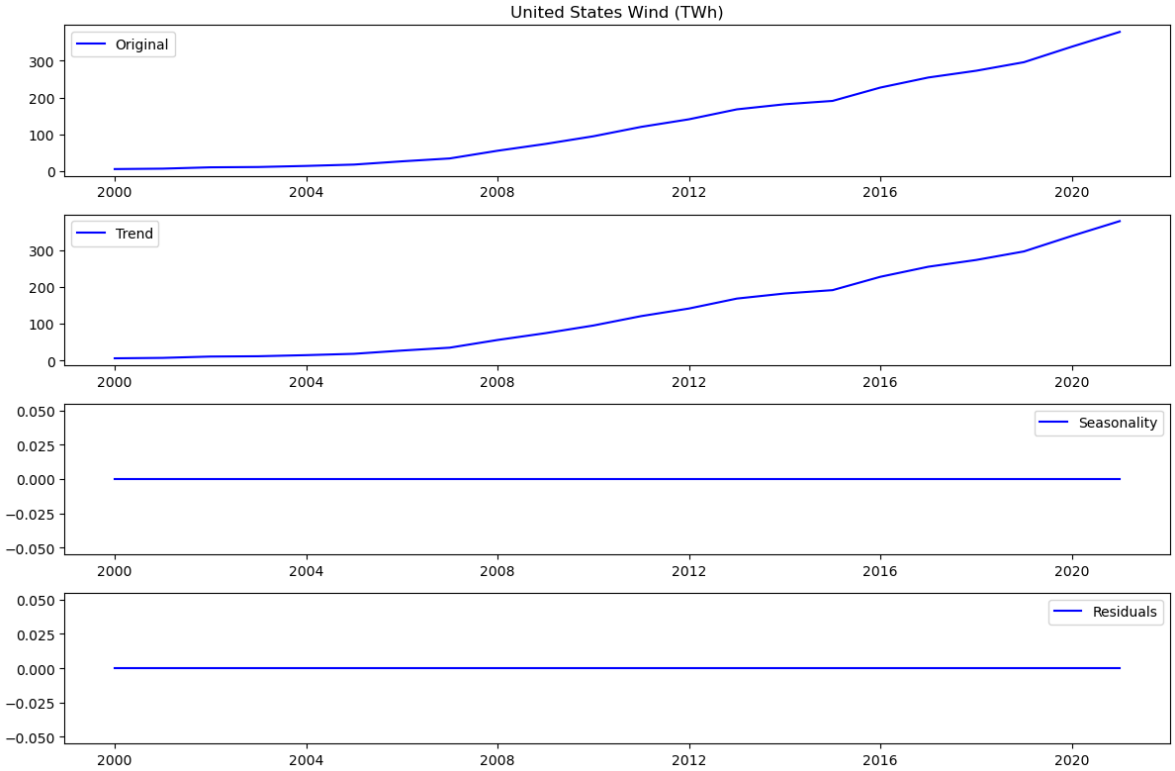
Out[159]:

	Dataset	Mean	Median	Standard Deviation	Skew	Fisher Kurtosis
0	United States Coal (TWh)	1,607.17	1,744.66	399.15	-0.68	-0.90
1	United States Gas (TWh)	1,060.39	1,000.70	325.82	0.29	-1.21
2	United States Oil (TWh)	67.20	45.97	36.59	0.89	-0.92
3	United States Nuclear (TWh)	790.04	790.04	15.55	-0.64	-0.53
4	United States Hydro (TWh)	264.36	263.82	21.08	-0.22	1.03
5	United States Bioenergy (TWh)	57.18	55.81	3.92	0.35	-0.98
6	United States Wind (TWh)	132.63	107.42	116.84	0.57	-0.95

This has a positive skew with wide and flat tails.

7.7.2 Decomposing The Data

In [160]:
```
decomp_graph(model_data.iloc[:,6:7])
```

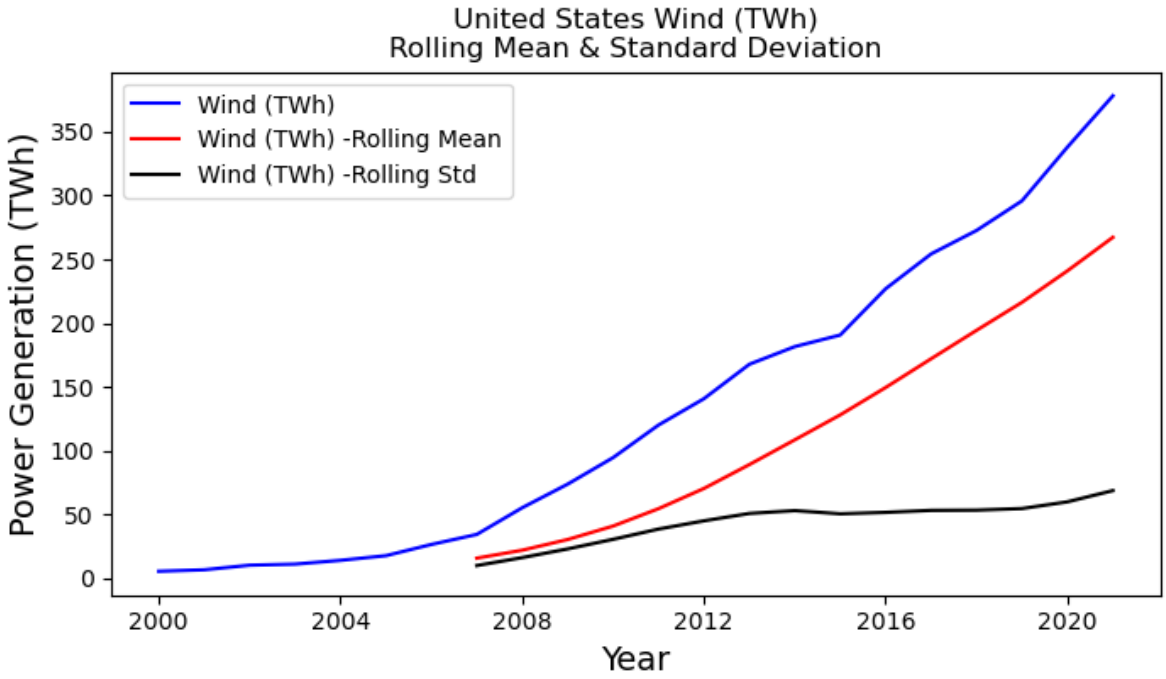


No seasonality or residuals. I'll check for stationarity.

7.7.3 Checking for Stationarity and Flattening

In [161]:
```
pd.set_option('display.max_rows', 30)
stationarity_check(model_data.iloc[:,6:7], s=14)
```

```
United States Wind (TWh)
Results of Dickey-Fuller test:

Test Statistic                -3.35
p-value                        0.01
#Lags Used                     9.00
Number of Observations Used   12.00
Critical Value (1%)           -4.14
Critical Value (5%)           -3.15
Critical Value (10%)          -2.71
dtype: float64
```



For the first time, we have stationary data. There is no need for transforming at all. However, for the sake of keeping the transformation column consistent, I'll still create a copy of this data into the transformation DataFrame.

In [162]:
```
stored = model_data_t.copy()
```

In [163]:
```
model_data_t = stored.copy()
t = model_data.iloc[:,6:7]
t.columns = ['Copy of ' + model_data.columns[6]]
model_data_t.insert(13,t.columns[0],t)

model_data_t.iloc[:,12:14].head(10)
```

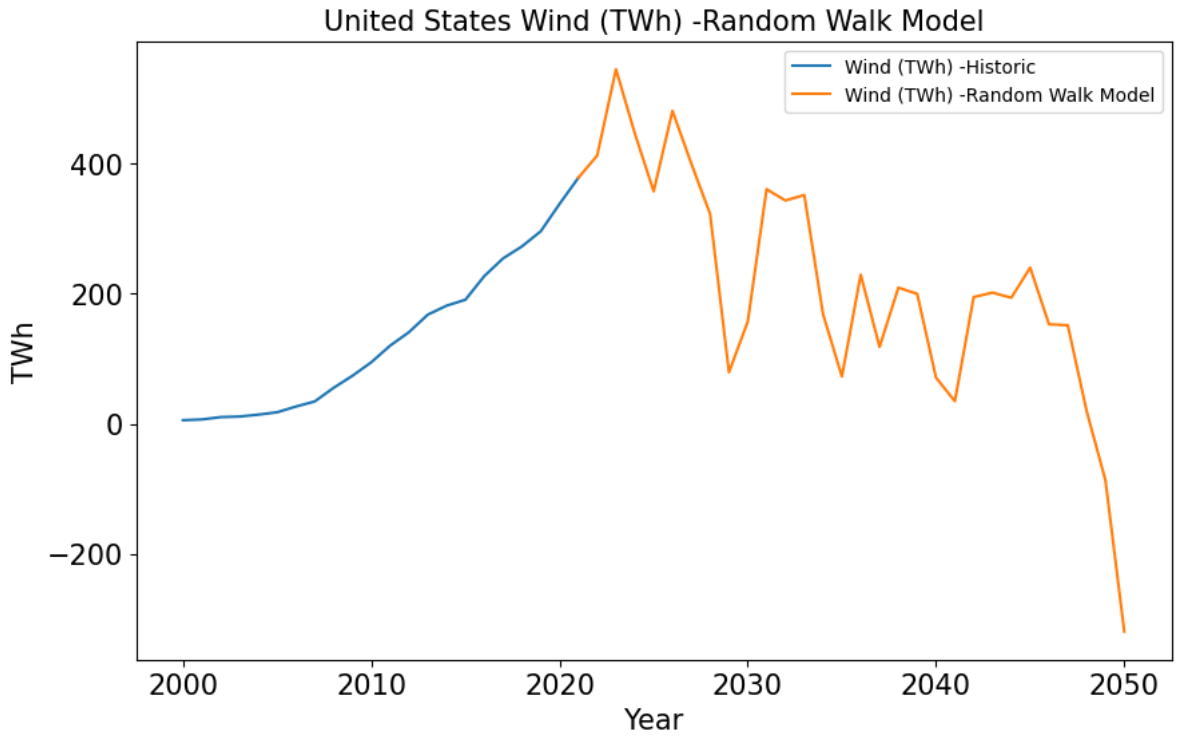
Out[163]:

Year	United States Wind (TWh)	Copy of United States Wind (TWh)
2000-01-01	5.59	5.59
2001-01-01	6.74	6.74
2002-01-01	10.35	10.35
2003-01-01	11.19	11.19
2004-01-01	14.14	14.14
2005-01-01	17.81	17.81
2006-01-01	26.59	26.59
2007-01-01	34.45	34.45
2008-01-01	55.36	55.36
2009-01-01	73.89	73.89

7.7.4 Random Walk Model

In [164]:
```
pd.set_option('display.max_rows',None)
y_hat_proj = random_walk(model_data.iloc[:,6:7])
y_hat_proj.columns = [model_data.columns[6]+' -Random Walk Model']

# Plot the transformed Random Walk
pred_graph(model_data.iloc[:,6:7], y_hat_proj.iloc[:,0:1], 14)
```

```
Random Walk with Standard Deviation of Transformed Data set: 116.84
```

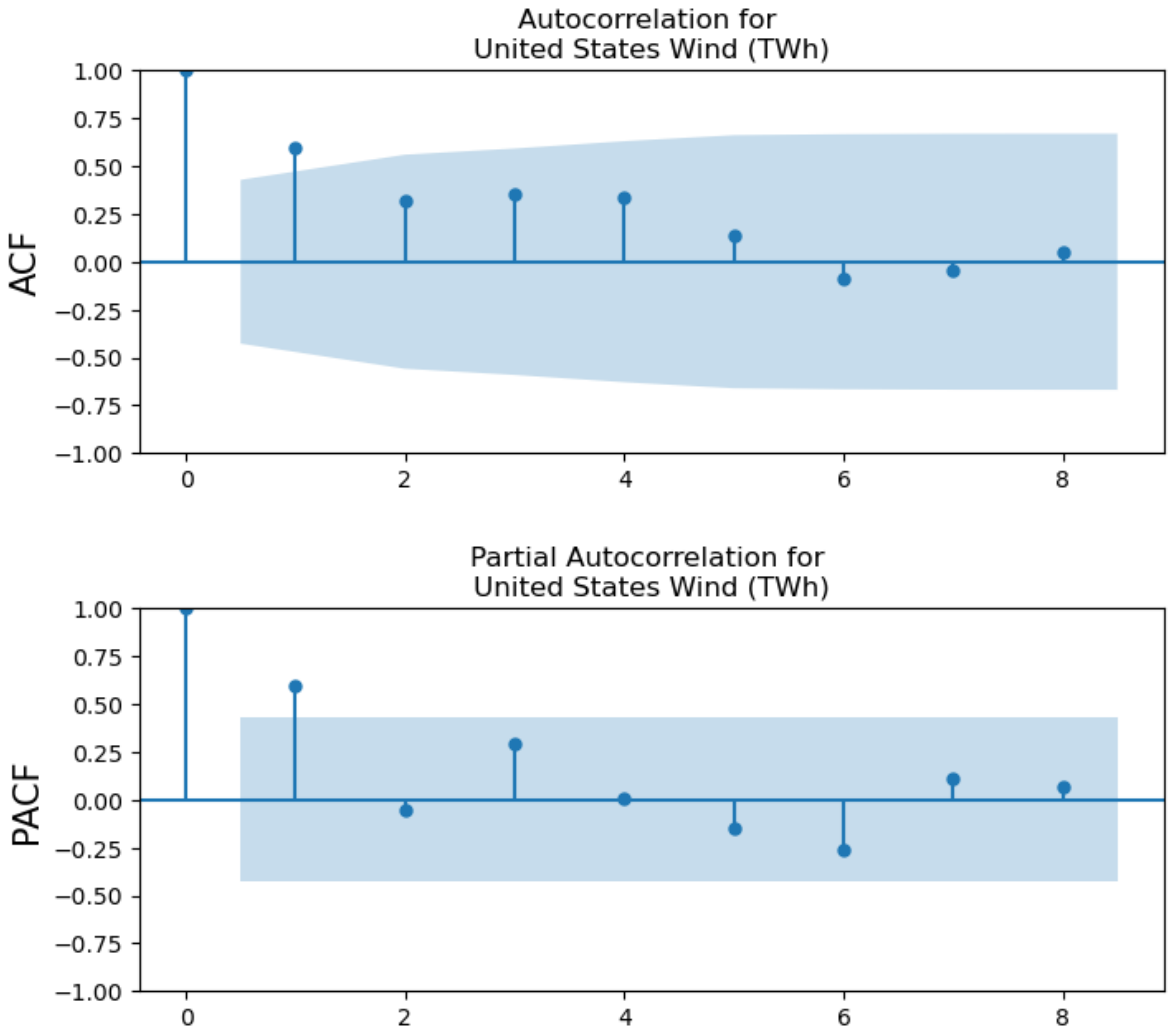


This model is as unreliable as it's name would imply, random walk. This one works better because there is no transformation.

Now I'll move forward with an ARMA model. First, I need to evaluate the ACF and PACF plots to find the best Autoregressive and Moving Average terms.

7.7.5 Evaluating Initial AR and MA Values with ACF and PACF Plots

In [165]:
```
plotacf(model_data.iloc[:,6:7], lags = 8)
plotpacf(model_data.iloc[:,6:7], lags = 8)
```

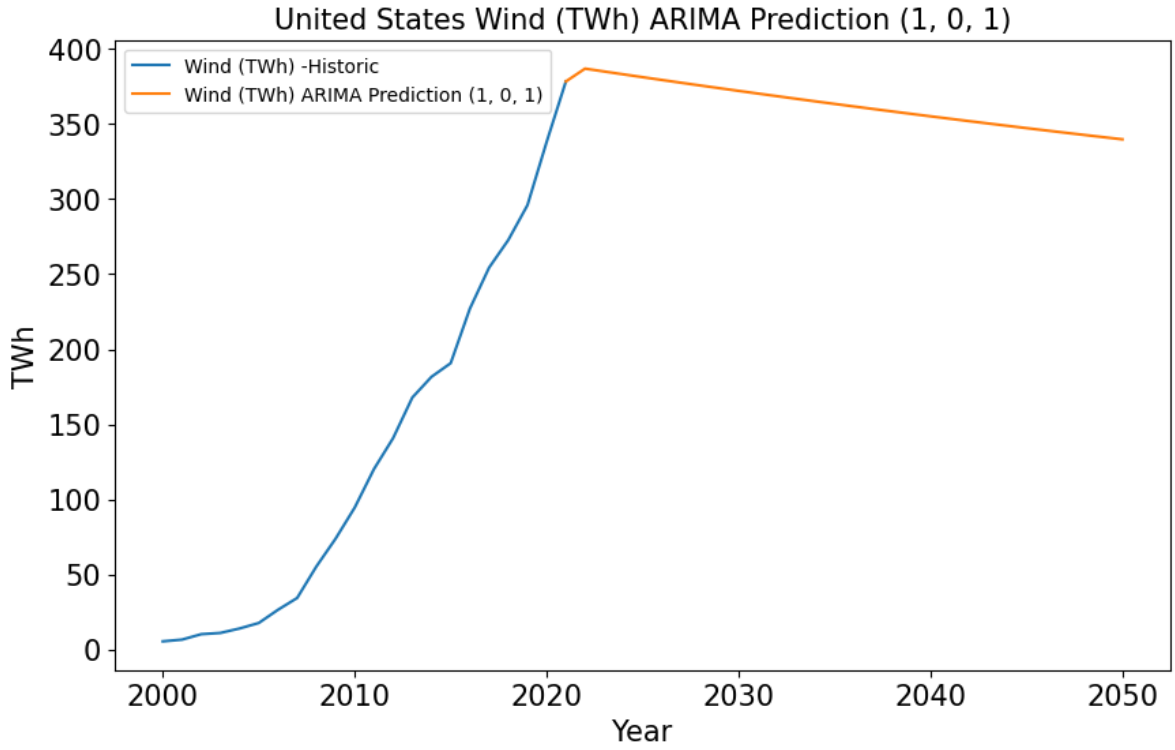


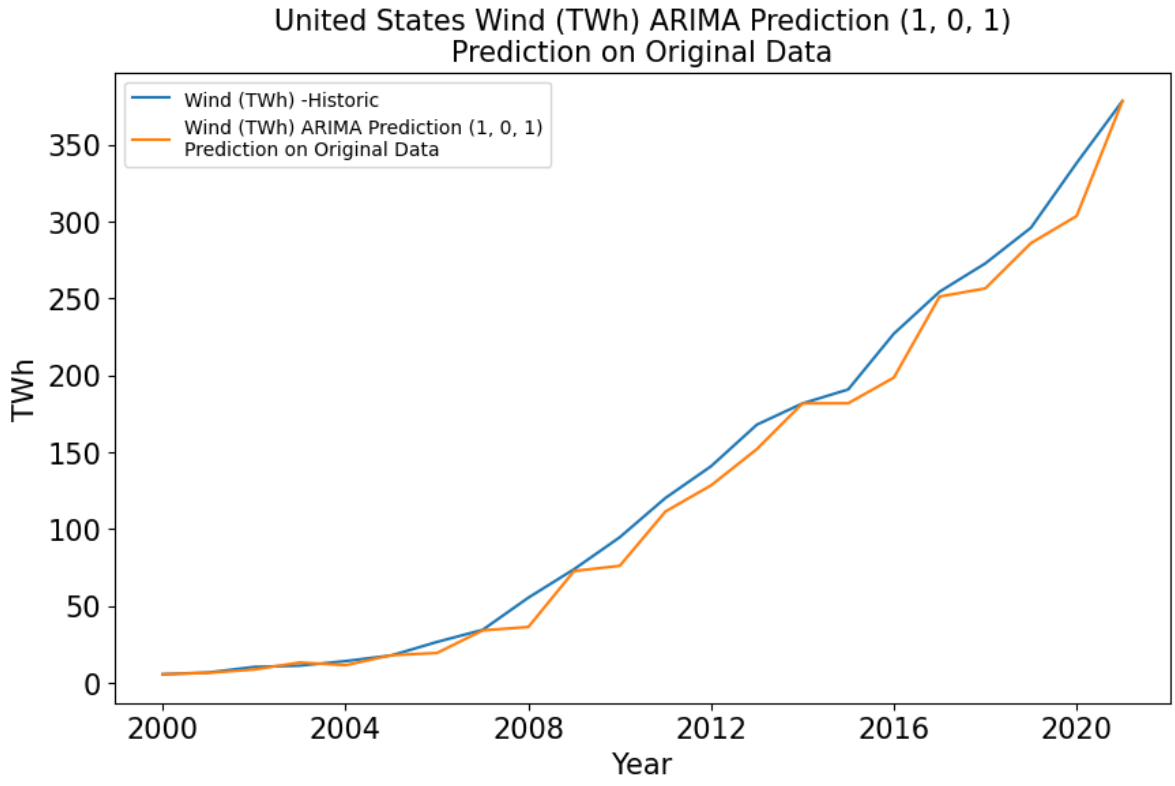
ACF and PACF both breach at one. I'll try (1, 0, 1) for my first ARMA model.

7.7.6 ARMA Model

In [166]:
```
order = (1,0,1)
model_summary_no_tran(model_data.iloc[:,6:7], order, 22, 50, 14);
```

```
ARIMA: (1, 0, 1), RMSE=13.25, AIC=191.62
**********************************************************************
**
United States Wind (TWh) -ARIMA Prediction (1, 0, 1) model results.

                                SARIMAX Results
==========================================================================
=======
Dep. Variable:     United States Wind (TWh)   No. Observations:
22
Model:                        ARIMA(1, 0, 1)   Log Likelihood
-91.811
Date:                      Sat, 29 Apr 2023   AIC
191.622
Time:                              06:22:55   BIC
195.986
Sample:                         01-01-2000   HQIC
192.650
                              - 01-01-2021
Covariance Type:                        opg
==========================================================================
=
                 coef    std err          z      P>|z|      [0.025      0.97
5]
--------------------------------------------------------------------------
-
const        192.6872    174.737      1.103      0.270    -149.790     535.16
5
ar.L1          0.9901      0.060     16.517      0.000       0.873       1.10
8
ma.L1          0.8598      0.187      4.603      0.000       0.494       1.22
6
sigma2       183.6270     60.533      3.033      0.002      64.984     302.27
0
==========================================================================
======
Ljung-Box (L1) (Q):                   0.16   Jarque-Bera (JB):
1.03
Prob(Q):                              0.69   Prob(JB):
0.60
Heteroskedasticity (H):               9.97   Skew:
0.50
Prob(H) (two-sided):                  0.01   Kurtosis:
3.35
==========================================================================
======

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (comple
x-step).
```



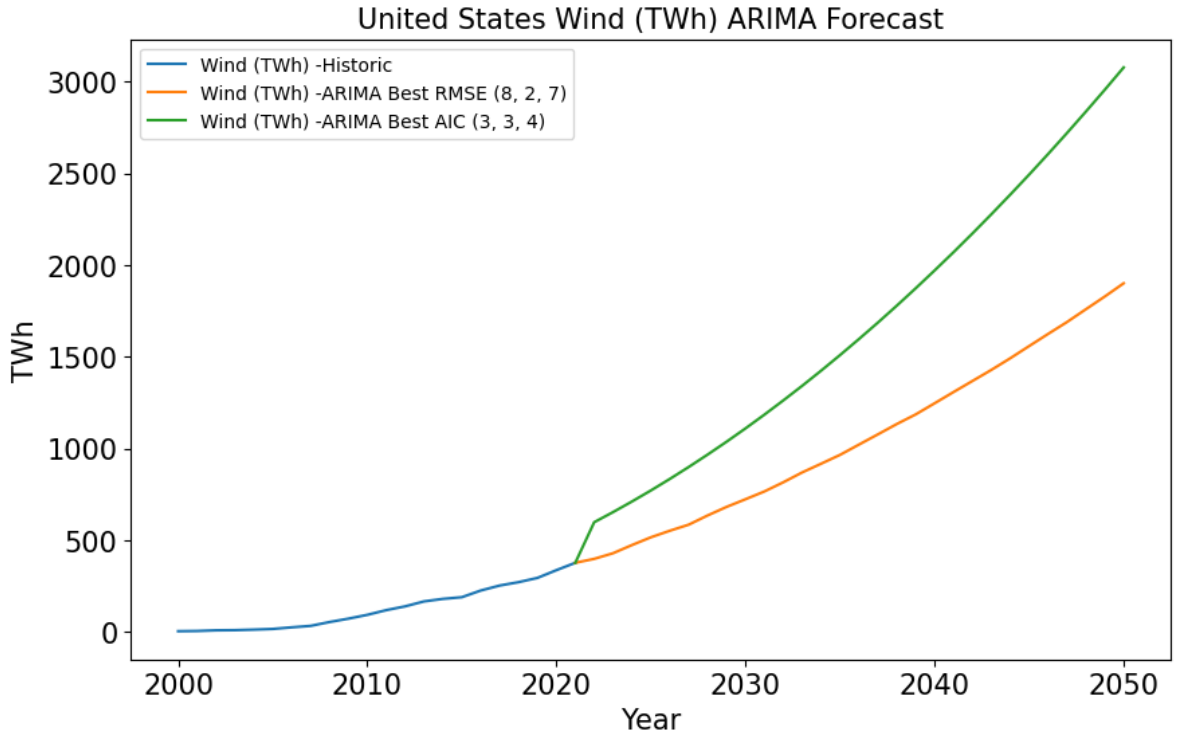


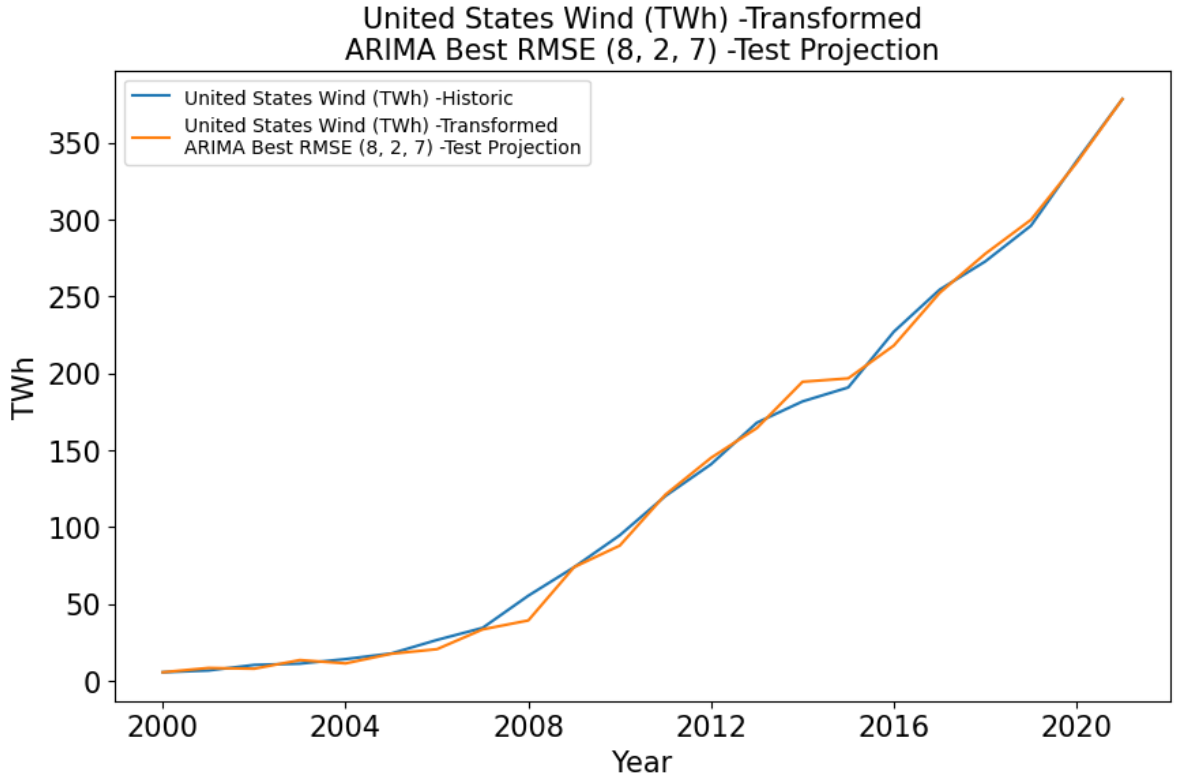
No, no and no. Wind is doing nothing but rising in the US. This graph should reflect that. I'll have to look at this through Grid Search.

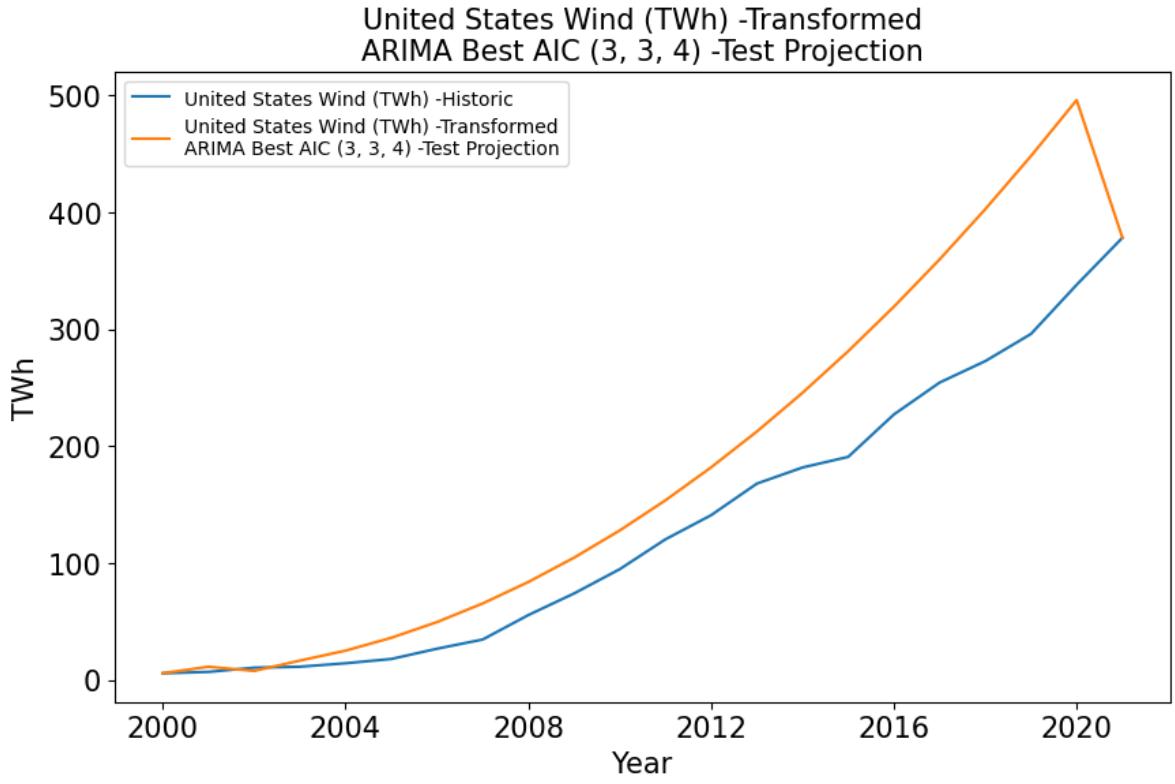
7.7.7 ARIMA Without Transformation

In [167]:
```
bio_rmse_cfg, oil_aic_cfg = arima_pdq_no_tran(
                                        model_data.iloc[:,6:7], 22, 50, s=14)
```

```
United States Wind (TWh) Grid Search:
RMSE ARIMA: (0, 0, 0), RMSE= 100.88, AIC= 275.91, TWh-2050= 132.63
RMSE ARIMA: (0, 0, 1), RMSE= 56.12, AIC= 253.41, TWh-2050= 135.42
RMSE ARIMA: (0, 0, 2), RMSE= 34.38, AIC= 233.04, TWh-2050= 138.21
RMSE ARIMA: (0, 0, 3), RMSE= 23.28, AIC= 220.15, TWh-2050= 142.14
RMSE ARIMA: (0, 0, 4), RMSE= 18.95, AIC= 210.59, TWh-2050= 140.30
RMSE ARIMA: (0, 0, 5), RMSE= 16.25, AIC= 206.27, TWh-2050= 136.61
RMSE ARIMA: (0, 0, 7), RMSE= 13.66, AIC= 205.14, TWh-2050= 133.58
RMSE ARIMA: (0, 0, 8), RMSE= 11.93, AIC= 200.75, TWh-2050= 147.38
AIC  ARIMA: (0, 1, 0), RMSE= 19.28, AIC= 190.64, TWh-2050= 378.20
AIC  ARIMA: (0, 1, 1), RMSE= 13.09, AIC= 174.46, TWh-2050= 387.89
RMSE ARIMA: (0, 1, 2), RMSE= 11.83, AIC= 172.12, TWh-2050= 402.06
RMSE ARIMA: (0, 1, 3), RMSE= 11.45, AIC= 172.69, TWh-2050= 409.18
RMSE ARIMA: (0, 1, 4), RMSE= 9.99, AIC= 171.39, TWh-2050= 440.86
RMSE ARIMA: (0, 1, 5), RMSE= 8.82, AIC= 166.86, TWh-2050= 451.59
RMSE ARIMA: (0, 1, 7), RMSE= 8.39, AIC= 168.88, TWh-2050= 441.80
RMSE ARIMA: (0, 1, 8), RMSE= 7.19, AIC= 165.51, TWh-2050= 481.42
AIC  ARIMA: (0, 2, 0), RMSE= 9.03, AIC= 148.68, TWh-2050= 1545.74
AIC  ARIMA: (0, 2, 1), RMSE= 8.34, AIC= 148.23, TWh-2050= 1464.81
AIC  ARIMA: (0, 2, 3), RMSE= 7.41, AIC= 147.90, TWh-2050= 1591.37
RMSE ARIMA: (0, 2, 7), RMSE= 6.94, AIC= 153.78, TWh-2050= 1393.97
RMSE ARIMA: (0, 2, 8), RMSE= 6.65, AIC= 153.93, TWh-2050= 1279.16
AIC  ARIMA: (0, 3, 1), RMSE= 9.24, AIC= 146.48, TWh-2050= 2396.36
AIC  ARIMA: (0, 3, 2), RMSE= 7.79, AIC= 144.44, TWh-2050= 2160.50
RMSE ARIMA: (0, 3, 8), RMSE= 6.58, AIC= 149.72, TWh-2050= 2029.50
RMSE ARIMA: (1, 2, 8), RMSE= 6.55, AIC= 155.15, TWh-2050= 1397.51
RMSE ARIMA: (1, 3, 7), RMSE= 6.52, AIC= 149.30, TWh-2050= 2024.67
RMSE ARIMA: (2, 2, 7), RMSE= 6.51, AIC= 155.95, TWh-2050= 1330.52
RMSE ARIMA: (2, 2, 8), RMSE= 6.51, AIC= 157.07, TWh-2050= 1396.23
AIC  ARIMA: (2, 3, 1), RMSE= 7.53, AIC= 143.94, TWh-2050= 2266.57
RMSE ARIMA: (2, 3, 7), RMSE= 6.30, AIC= 150.08, TWh-2050= 1990.30
RMSE ARIMA: (2, 3, 8), RMSE= 6.20, AIC= 151.09, TWh-2050= 2012.20
RMSE ARIMA: (3, 2, 8), RMSE= 6.09, AIC= 156.81, TWh-2050= 1827.13
AIC  ARIMA: (3, 3, 4), RMSE= 69.58, AIC= 16.00, TWh-2050= 3076.59
RMSE ARIMA: (3, 3, 8), RMSE= 6.09, AIC= 152.44, TWh-2050= 1976.86
RMSE ARIMA: (4, 2, 8), RMSE= 5.97, AIC= 158.23, TWh-2050= 1803.24
RMSE ARIMA: (5, 2, 8), RMSE= 5.93, AIC= 160.06, TWh-2050= 1948.21
RMSE ARIMA: (6, 2, 8), RMSE= 5.91, AIC= 162.04, TWh-2050= 1940.03
RMSE ARIMA: (7, 2, 7), RMSE= 5.88, AIC= 162.29, TWh-2050= 1876.05
RMSE ARIMA: (7, 3, 6), RMSE= 5.85, AIC= 155.45, TWh-2050= 1916.13
RMSE ARIMA: (7, 3, 7), RMSE= 5.79, AIC= 156.36, TWh-2050= 2017.14
RMSE ARIMA: (7, 3, 8), RMSE= 5.75, AIC= 157.28, TWh-2050= 2048.64
RMSE ARIMA: (8, 2, 7), RMSE= 5.71, AIC= 162.12, TWh-2050= 1900.81
Best RMSE ARIMA: (8, 2, 7) RMSE= 5.71 AIC= 162.12, TWh-2050= 1900.81
Best AIC  ARIMA: (3, 3, 4) RMSE= 69.58 AIC= 16.00, TWh-2050= 3076.59
```


United States Wind (TWh) ARIMA Forecast
Wind (TWh) -Historic
Wind (TWh) -ARIMA Best RMSE (8, 2, 7)
Wind (TWh) -ARIMA Best AIC (3, 3, 4)
TWh
Year
0
500
1000
1500
2000
2500
3000
2000
2010
2020
2030
2040
2050


United States Wind (TWh) -Transformed
ARIMA Best RMSE (8, 2, 7) -Test Projection
United States Wind (TWh) -Historic
United States Wind (TWh) -Transformed
ARIMA Best RMSE (8, 2, 7) -Test Projection
TWh
Year
0
50
100
150
200
250
300
350
2000
2004
2008
2012
2016
2020



The best RMSE looks decent, the best AIC is clearly wrong. Wind is on the rise, so here's to hoping it will continue to do so.

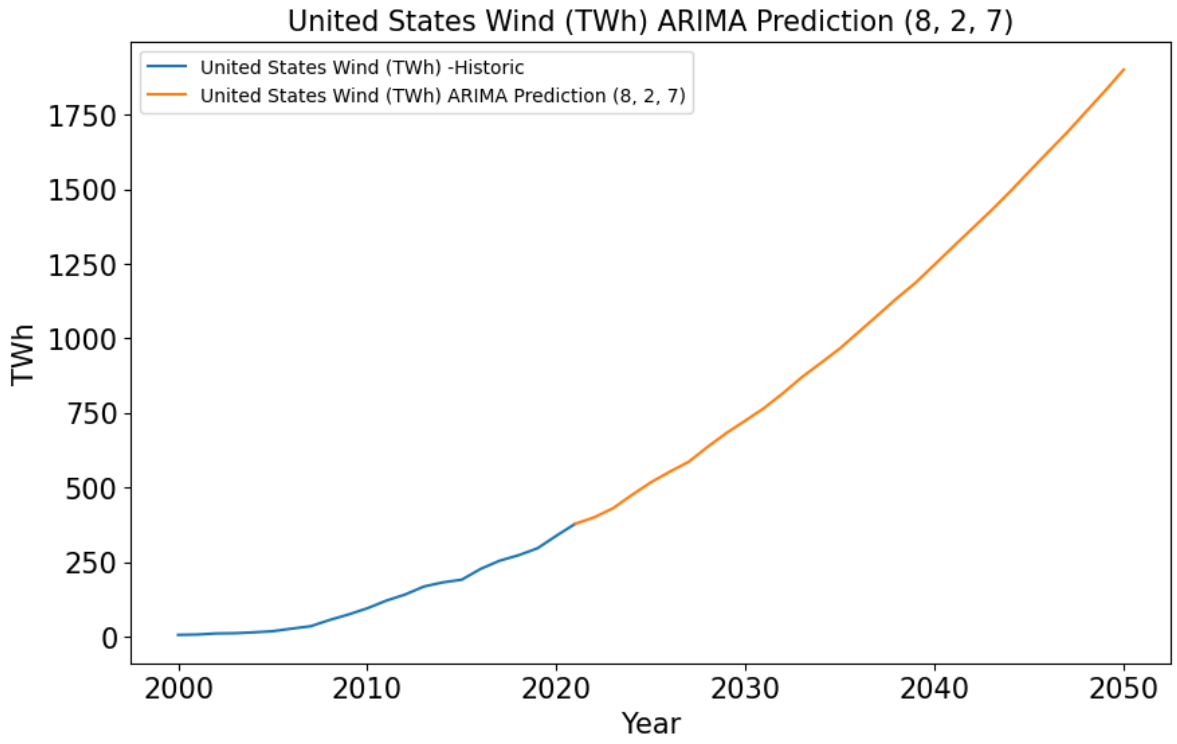
7.7.8 Model Selection

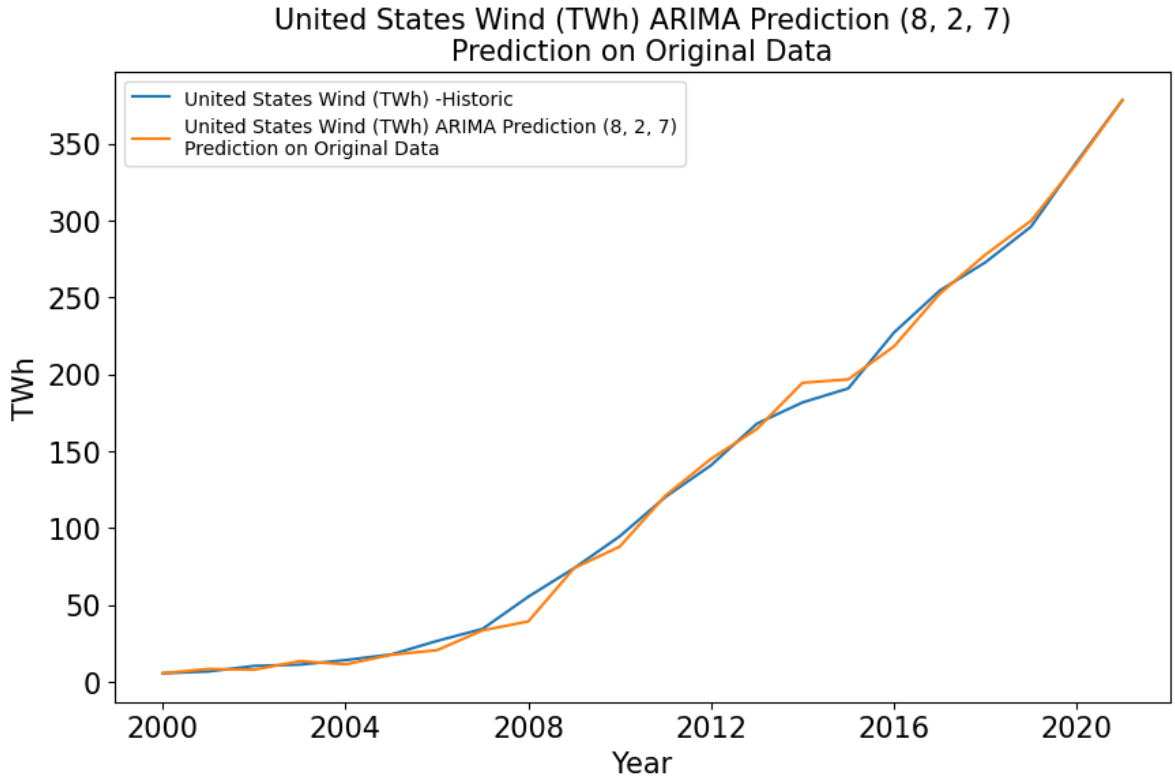
In [168]:
```
#Deleteme
stored_wind = selected_models.copy()
```

In [169]:
```
selected_models = stored_wind.copy() #deleteme
df_wind = model_data_t.iloc[:,12:13]

#                         Model,   df_o,   pdq,     tran, n, PDQs
selected_models.append(['ARIMA', df_wind, (8,2,7), None, 0, None])

m = 6
model_summary_no_tran(selected_models[m][1], selected_models[m][2], 22, 50);
```

```
ARIMA: (8, 2, 7), RMSE=5.71, AIC=162.12
**********************************************************************
**
United States Wind (TWh) -ARIMA Prediction (8, 2, 7) model results.

                                SARIMAX Results
==========================================================================
=======
Dep. Variable:     United States Wind (TWh)   No. Observations:
22
Model:                       ARIMA(8, 2, 7)   Log Likelihood
-65.060
Date:                      Sat, 29 Apr 2023   AIC
162.119
Time:                              06:23:52   BIC
178.051
Sample:                         01-01-2000   HQIC
165.229
                              - 01-01-2021
Covariance Type:                        opg
==========================================================================
=
                 coef    std err          z      P>|z|      [0.025      0.97
5]
--------------------------------------------------------------------------
-
ar.L1         -0.2625      5.326     -0.049      0.961     -10.701      10.17
6
ar.L2          0.6955      9.556      0.073      0.942     -18.033      19.42
4
ar.L3          0.1150      7.045      0.016      0.987     -13.694      13.92
4
ar.L4          0.1815      7.503      0.024      0.981     -14.523      14.88
6
ar.L5          0.3253      7.559      0.043      0.966     -14.490      15.14
0
ar.L6         -0.3578      9.525     -0.038      0.970     -19.027      18.31
1
ar.L7         -0.1876     12.050     -0.016      0.988     -23.805      23.43
0
ar.L8          0.4700      5.889      0.080      0.936     -11.072      12.01
2
ma.L1          0.0286     10.734      0.003      0.998     -21.010      21.06
7
ma.L2         -1.6245     15.472     -0.105      0.916     -31.950      28.70
1
ma.L3          0.8190     12.909      0.063      0.949     -24.481      26.12
0
ma.L4          0.8659     15.086      0.057      0.954     -28.702      30.43
4
ma.L5         -1.6362     11.976     -0.137      0.891     -25.108      21.83
6
ma.L6         -0.1504     13.535     -0.011      0.991     -26.679      26.37
8
ma.L7          0.8528     24.436      0.035      0.972     -47.041      48.74
7
```

```
sigma2        16.5651      3.034      5.460      0.000      10.618      22.51
2
===================================================================================
======
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):
0.60
Prob(Q):                              0.99   Prob(JB):
0.74
Heteroskedasticity (H):               0.85   Skew:
0.07
Prob(H) (two-sided):                  0.84   Kurtosis:
3.84
===================================================================================
======

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (comple
x-step).
[2] Covariance matrix is singular or near-singular, with condition number 3.7
1e+18. Standard errors may be unstable.
```





This model is great. My only hope is that wind is really on an exponential curve upward.

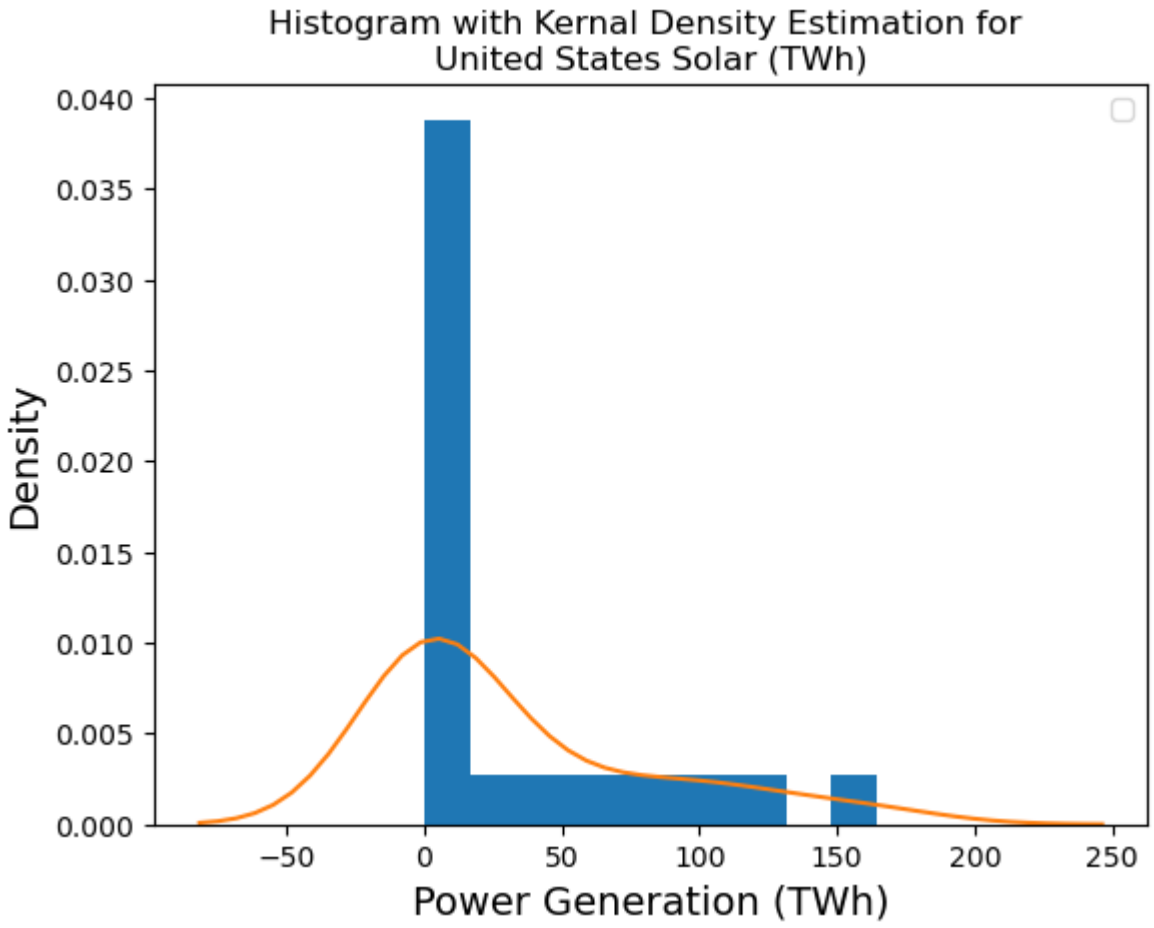
7.8 Solar

7.8.1 Distribution Investigation

In [170]:
```
stored_model_data = model_data.copy()
stored_model_data_t = model_data_t.copy()
```

In [171]:
```
model_data = stored_model_data.copy()
model_data_t = stored_model_data_t.copy()
```

In [172]:
```
hist(model_data.iloc[:,7:8])
```



In [173]:
```
stats_block = s_block(model_data.iloc[:,7:8], stats_block)
stats_block
```

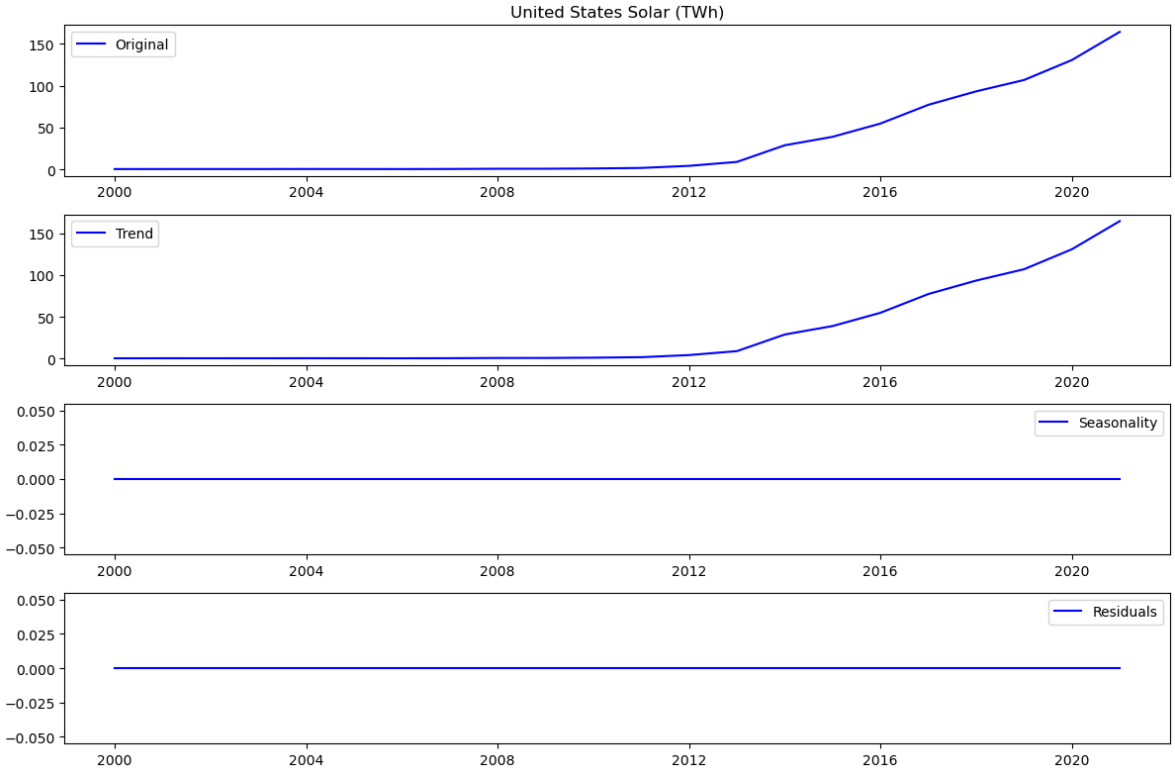
Out[173]:

	Dataset	Mean	Median	Standard Deviation	Skew	Fisher Kurtosis
0	United States Coal (TWh)	1,607.17	1,744.66	399.15	-0.68	-0.90
1	United States Gas (TWh)	1,060.39	1,000.70	325.82	0.29	-1.21
2	United States Oil (TWh)	67.20	45.97	36.59	0.89	-0.92
3	United States Nuclear (TWh)	790.04	790.04	15.55	-0.64	-0.53
4	United States Hydro (TWh)	264.36	263.82	21.08	-0.22	1.03
5	United States Bioenergy (TWh)	57.18	55.81	3.92	0.35	-0.98
6	United States Wind (TWh)	132.63	107.42	116.84	0.57	-0.95
7	United States Solar (TWh)	32.64	1.52	48.75	1.39	0.67

This is a very positively skewed distribution. It has thinner tails.

7.8.2 Decomposing The Data

In [174]:
```
decomp_graph(model_data.iloc[:,7:8])
```

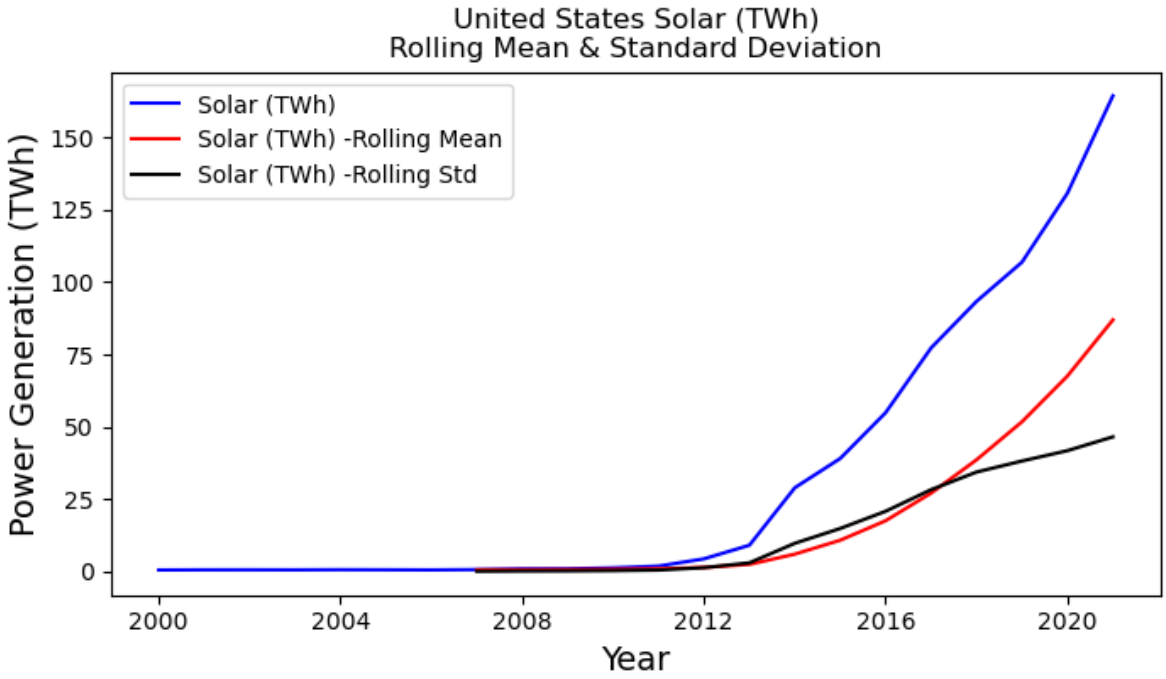


No seasonality or residuals. I'll check for stationarity.

7.8.3 Checking for Stationarity and Flattening

In [175]:
```
pd.set_option('display.max_rows', 30)
stationarity_check(model_data.iloc[:,7:8], s=14)
```

```
United States Solar (TWh)
Results of Dickey-Fuller test:

Test Statistic                  0.56
p-value                         0.99
#Lags Used                      9.00
Number of Observations Used    12.00
Critical Value (1%)            -4.14
Critical Value (5%)            -3.15
Critical Value (10%)           -2.71
dtype: float64
```



This is not close to stationary. After playing around with transformations, I found that if I took the log and then subtracted the rolling average of eight values, I can get it stationary.

In [176]:
```
stored = model_data_t.copy()
```

In [177]:
```
model_data_t = stored.copy()
t = subtract_roll_mean(log_transform(model_data.iloc[:,7:8]), n=8)
model_data_t.insert(15,t.columns[0],t)

model_data_t.iloc[:,14:16].head()
```

Out[177]:

Year	United States Solar (TWh)	Subtract Rolling Mean of Natural Logged United States Solar (TWh)
2000-01-01	0.49	NaN
2001-01-01	0.54	NaN
2002-01-01	0.55	NaN
2003-01-01	0.53	NaN
2004-01-01	0.58	NaN

Before I move forward, I want to make sure I can transform this data back after modeling.

In [178]:
```
test = model_data_t.iloc[:,14:16].copy()

test['Results'] = inv_log_inv_sub(test, n=8)
test
```

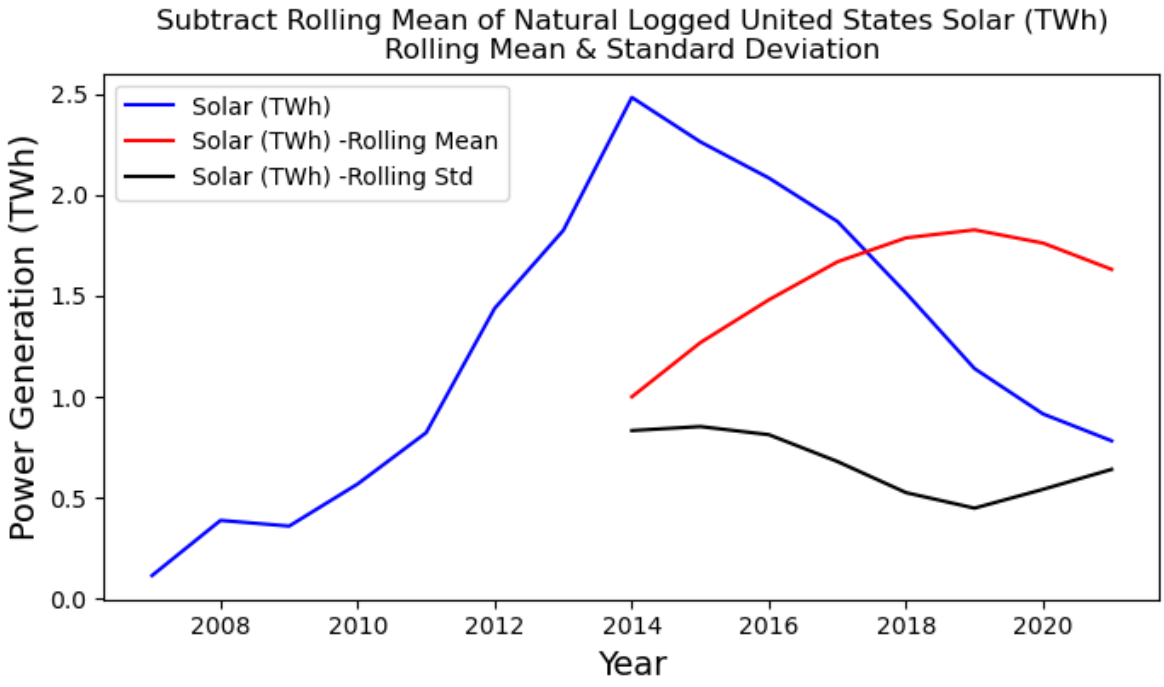
Out[178]:

Year	United States Solar (TWh)	Subtract Rolling Mean of Natural Logged United States Solar (TWh)	Results
2000-01-01	0.49	NaN	0.49
2001-01-01	0.54	NaN	0.54
2002-01-01	0.55	NaN	0.55
2003-01-01	0.53	NaN	0.53
2004-01-01	0.58	NaN	0.58
2005-01-01	0.55	NaN	0.55
2006-01-01	0.51	NaN	0.51
2007-01-01	0.61	0.11	0.61
2008-01-01	0.86	0.39	0.86
2009-01-01	0.89	0.36	0.89
2010-01-01	1.21	0.57	1.21
2011-01-01	1.82	0.82	1.82
2012-01-01	4.33	1.44	4.33
2013-01-01	9.04	1.82	9.04
2014-01-01	28.92	2.48	28.92
2015-01-01	39.03	2.26	39.03
2016-01-01	54.87	2.08	54.87
2017-01-01	77.28	1.87	77.28
2018-01-01	93.36	1.51	93.36
2019-01-01	106.89	1.14	106.89
2020-01-01	130.72	0.92	130.72
2021-01-01	164.42	0.78	164.42

Transferring back is a possibility. Therefore, I can move forward. I'll again check for stationarity.

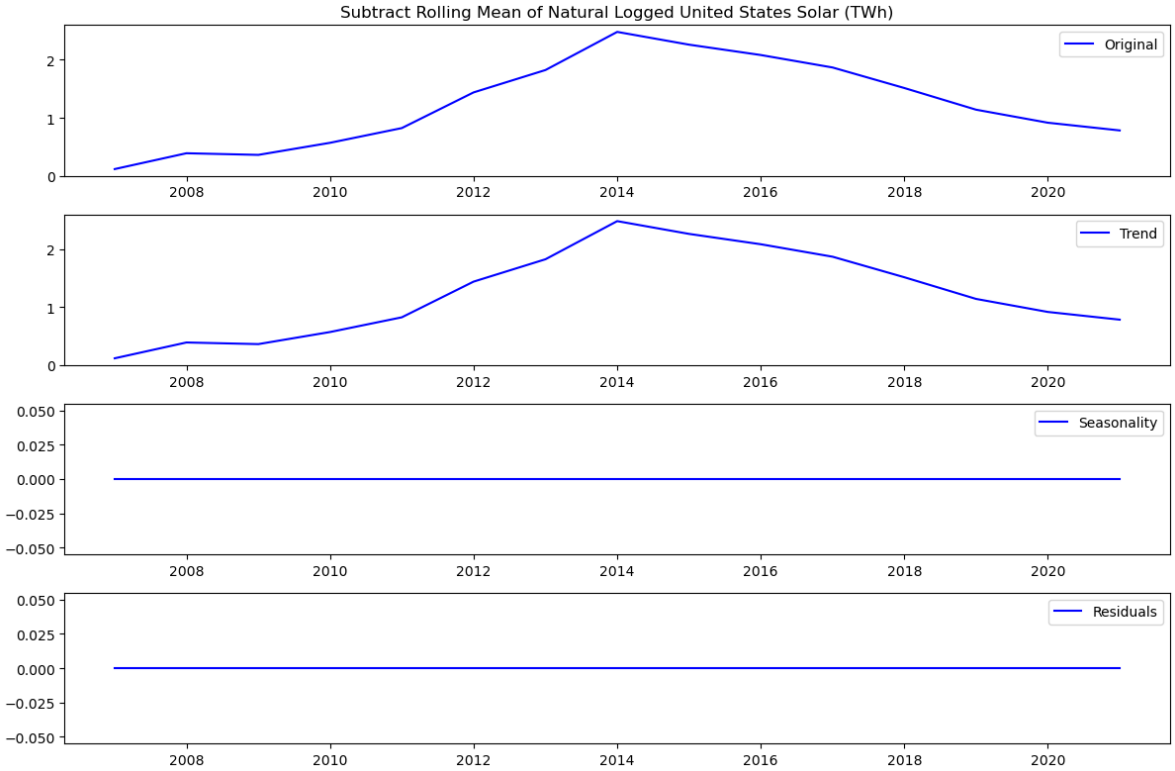
In [179]:
```
stationarity_check(model_data_t.iloc[:,15:16], s=54)
```

```
Subtract Rolling Mean of Natural Logged United States Solar (TWh)
Results of Dickey-Fuller test:

Test Statistic                -3.10
p-value                        0.03
#Lags Used                     2.00
Number of Observations Used   12.00
Critical Value (1%)           -4.14
Critical Value (5%)           -3.15
Critical Value (10%)          -2.71
dtype: float64
```



My p-value is less than .05. Stationarity achieved. I'll look at the decomposition graph again.

In [180]:
```
decomp_graph(model_data_t.iloc[:,15:16].dropna())
```

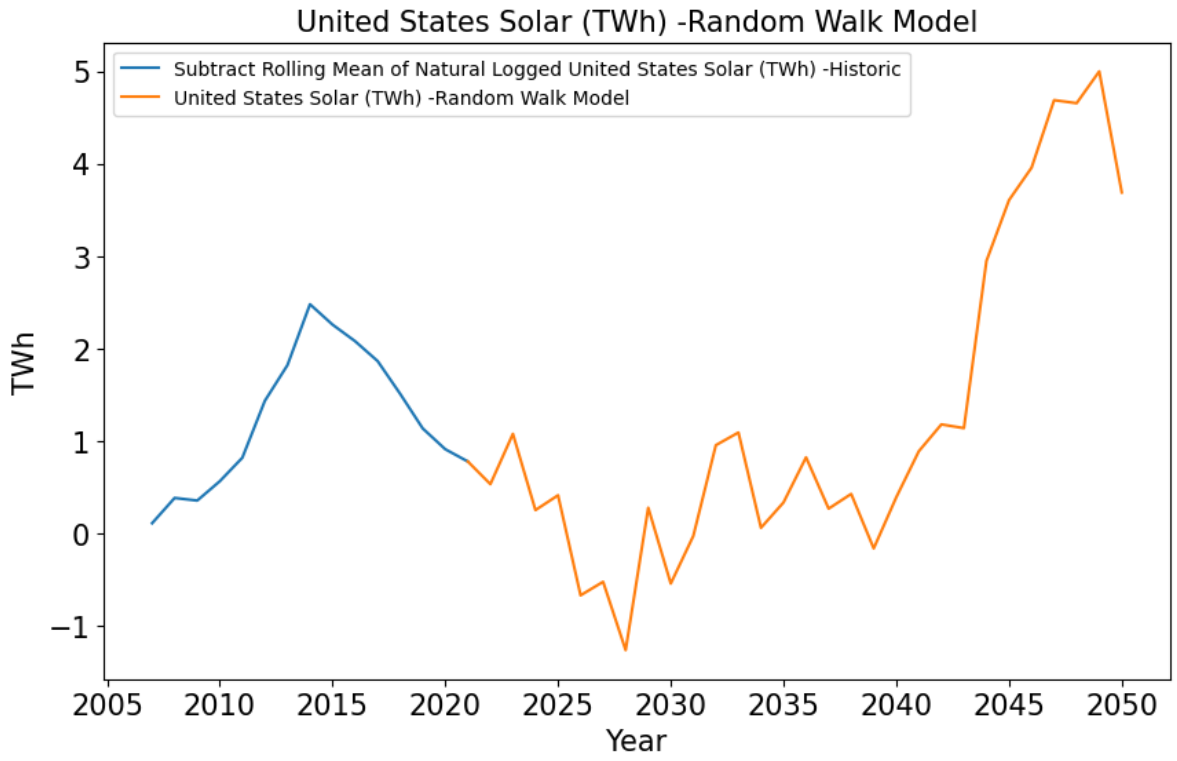


Looks like there's still a trend line, but it does pass stationarity, so I'm moving forward to the baseline random walk model.

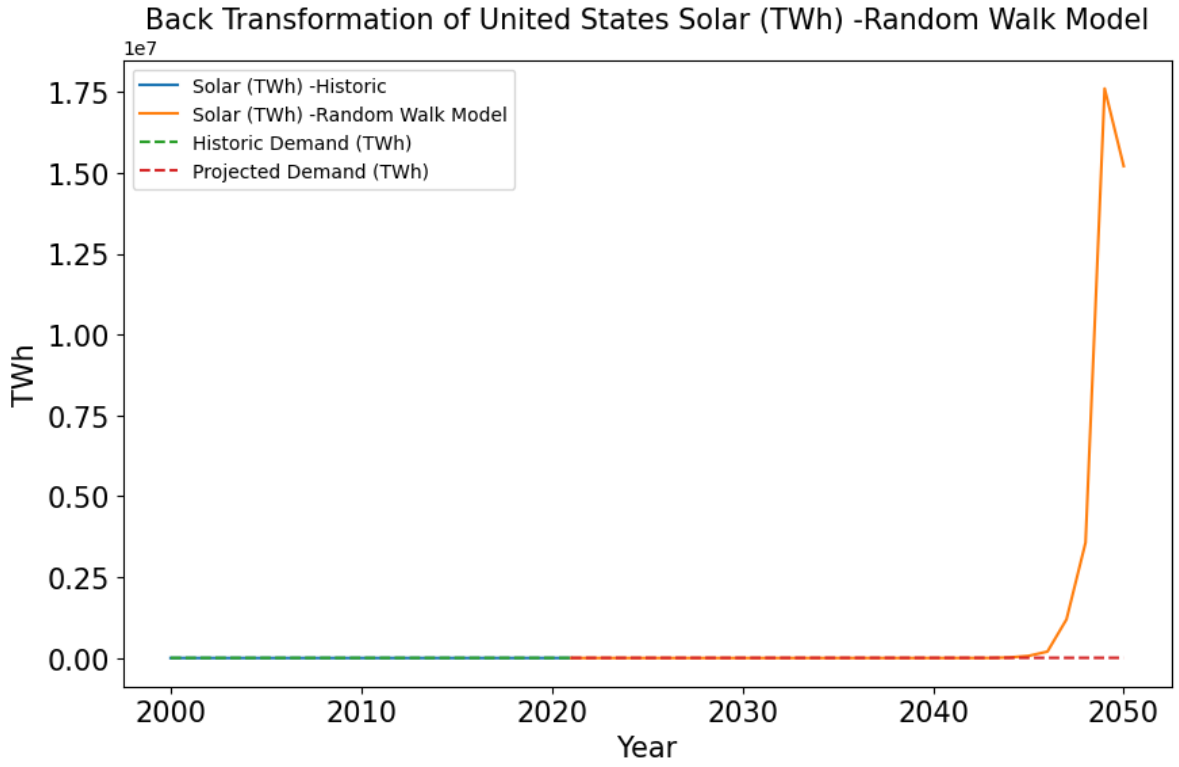
7.8.4 Random Walk Model

In [181]:
```
pd.set_option('display.max_rows',None)
y_hat_proj = random_walk(model_data_t.iloc[:,15:16])
y_hat_proj.columns = [model_data.columns[7]+' -Random Walk Model']

# Plot the transformed Random Walk
pred_graph(model_data_t.iloc[:,15:16], y_hat_proj.iloc[:,0:1])
```

```
Random Walk with Standard Deviation of Transformed Data set: 0.72
```



In [182]:
```
# Prepare the DataFrame for Back Transformation
ranplot = model_data_t.iloc[:,14:16].copy()
ranplot.rename(columns={str(ranplot.columns[1]): str(y_hat_proj.columns[0])},
               inplace=True)

ranplot = pd.concat([ranplot,y_hat_proj],axis=0)

# Perform Back Transformation
y_hat_proj = inv_log_inv_sub(ranplot, n=8)

#Plot the new graph
pred_graph(model_data_t.iloc[:,14:15], y_hat_proj.iloc[22:,0:1], 14)
```

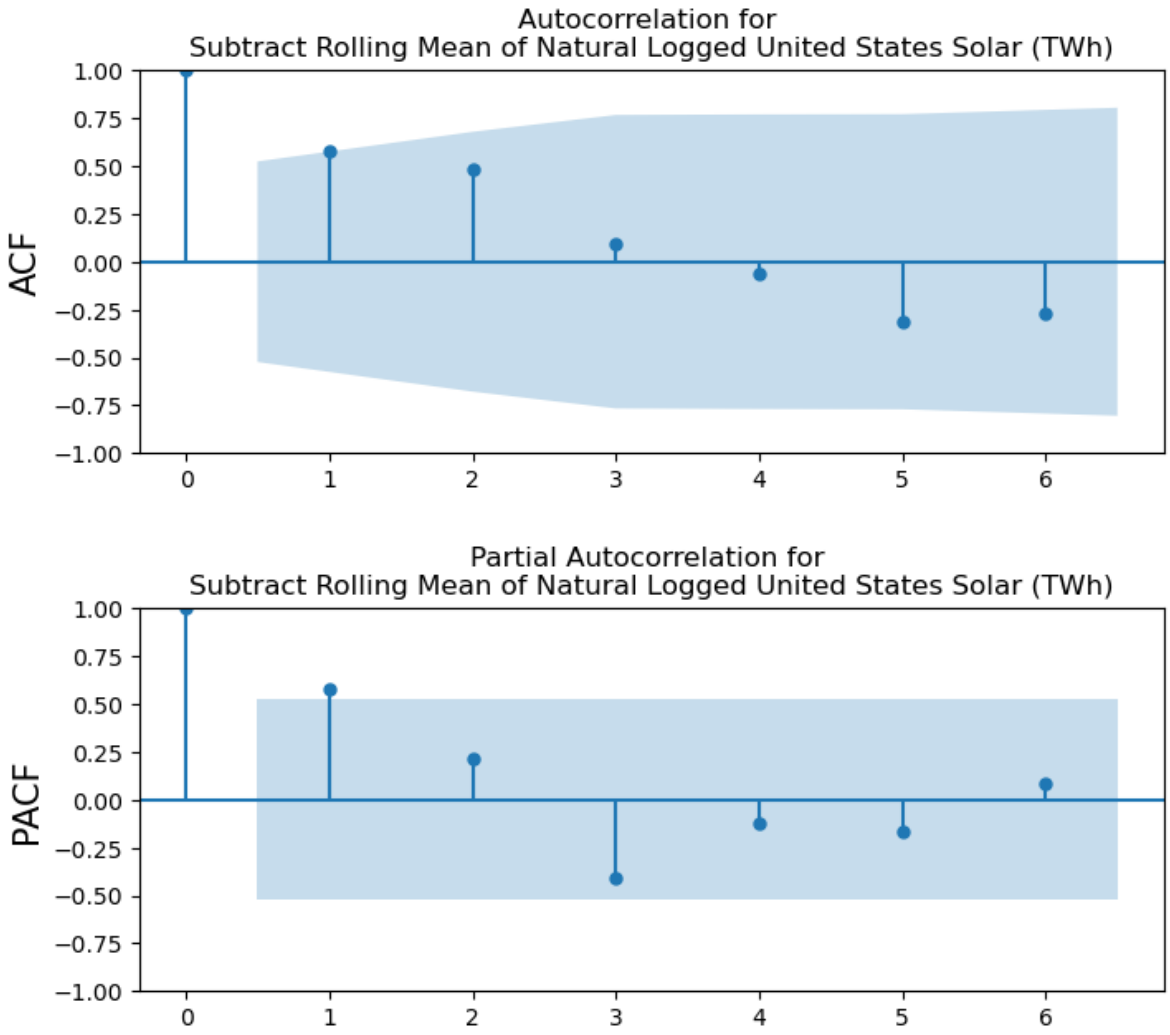


This model is as unreliable as it's name would imply, random walk. Oscillating about zero and a large standard deviation a bad random walk. This one doesn't disappoint.

Now I'll move forward with an ARMA model. First, I need to evaluate the ACF and PACF plots to find the best Autoregressive and Moving Average terms.

7.8.5 Evaluating Initial AR and MA Values with ACF and PACF Plots

In [183]:
```
plotacf(model_data_t.iloc[:,15:16], lags = 6)
plotpacf(model_data_t.iloc[:,15:16], lags = 6)
```

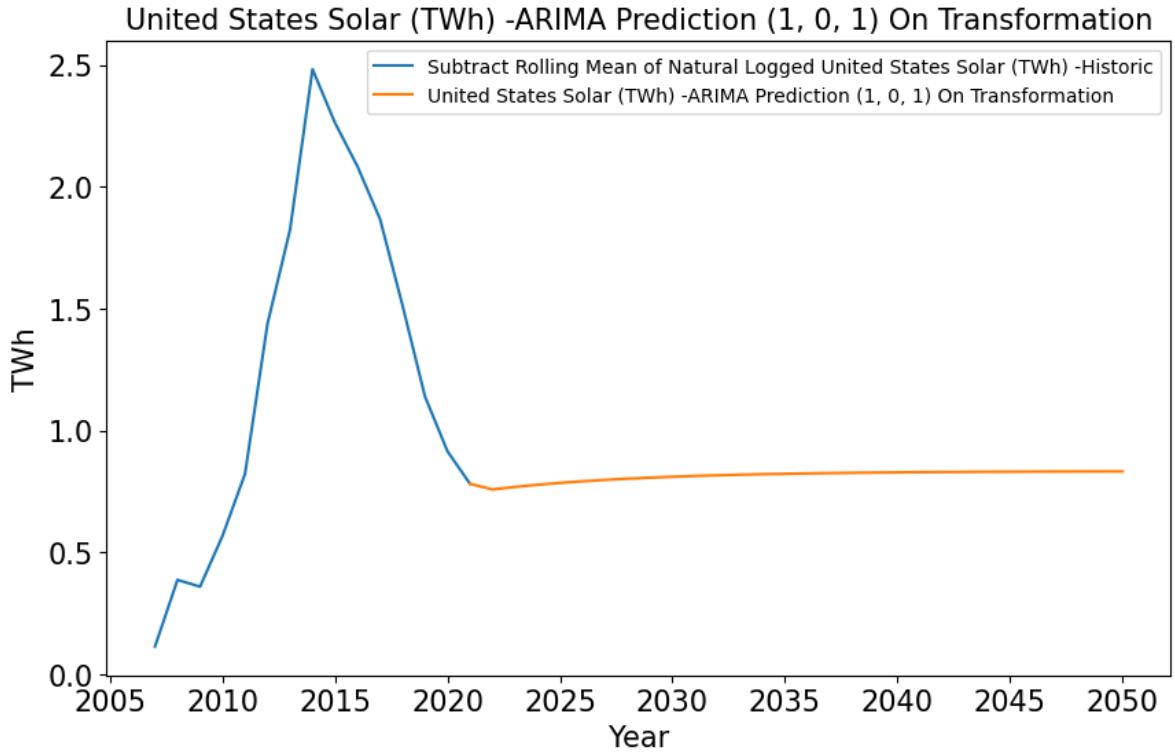


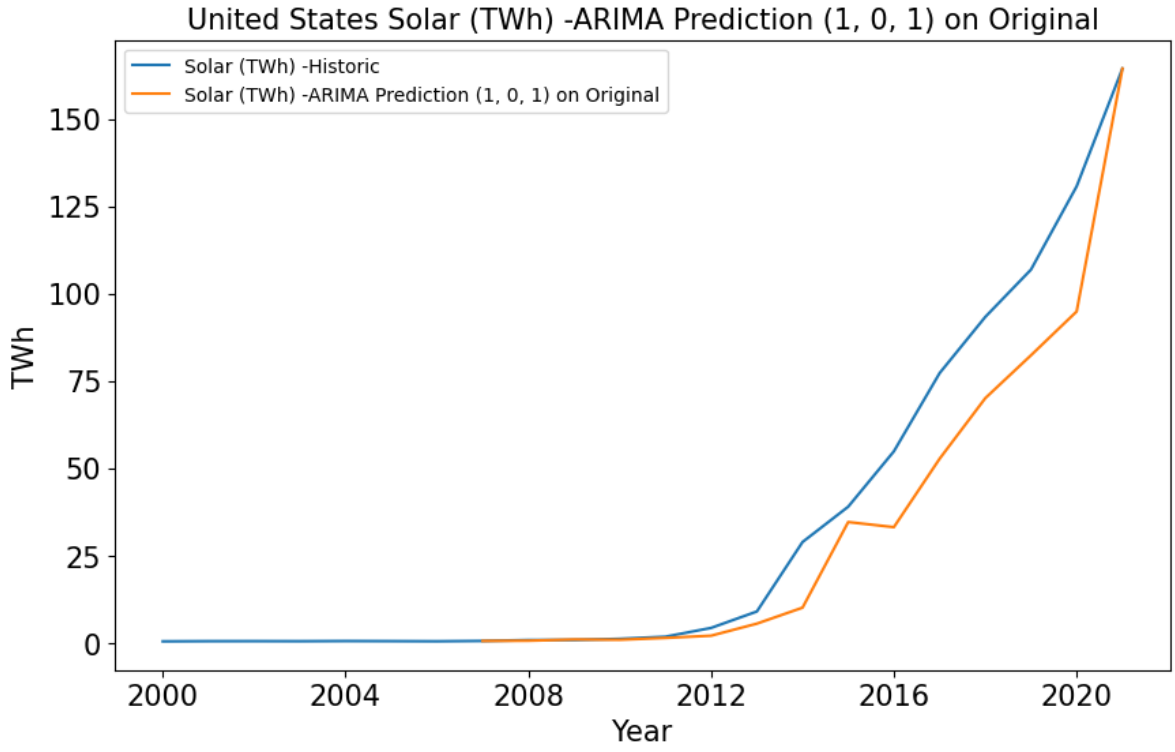
Both ACF and PACF breach at 1. I'll try (1, 0, 1) for the first ARMA model.

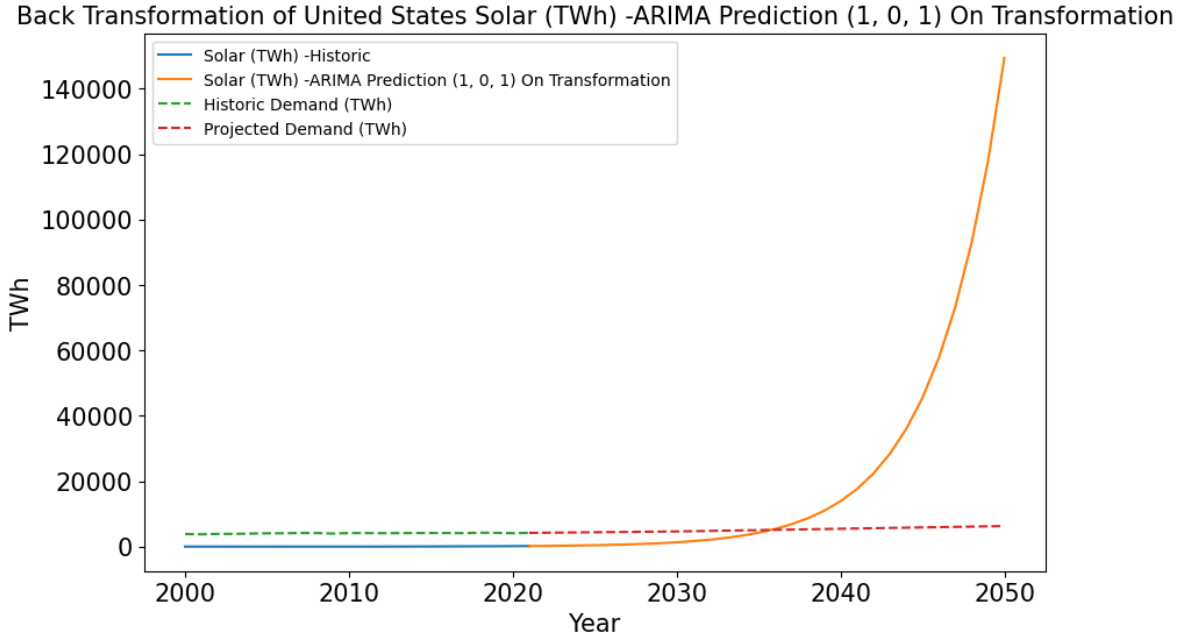
7.8.6 ARMA Model

In [184]:
```
pd.set_option('display.max_rows',60)
order = (1,0,1)
model_summary(model_data_t.iloc[:,14:16], order, 22, 50, 14,
          tran = 'inv_log_inv_sub', n=8)
```

```
ARIMA: (1, 0, 1), RMSE=20.57, AIC=15.35
**********************************************************************
**
United States Solar (TWh) model results.

                                                      SARIMAX Results
==========================================================================
==============================================
Dep. Variable:     Subtract Rolling Mean of Natural Logged United States Sola
r (TWh)   No. Observations:                   22
Model:                                                             ARIMA
(1, 0, 1)   Log Likelihood                -3.675
Date:                                                          Sat, 29 A
pr 2023   AIC                            15.350
Time:                                                                 0
6:23:56   BIC                            19.714
Sample:                                                            01-
01-2000   HQIC                           16.378
                                                                 - 01-
01-2021
Covariance Type:
opg
==========================================================================
=
                 coef    std err          z      P>|z|      [0.025      0.97
5]
--------------------------------------------------------------------------
-
const          0.8340      0.707      1.180      0.238      -0.551       2.21
9
ar.L1          0.8640      0.202      4.276      0.000       0.468       1.26
0
ma.L1          0.3602      0.422      0.854      0.393      -0.467       1.18
7
sigma2         0.0833      0.042      1.983      0.047       0.001       0.16
6
==========================================================================
======
Ljung-Box (L1) (Q):                   0.50   Jarque-Bera (JB):
7.36
Prob(Q):                              0.48   Prob(JB):
0.03
Heteroskedasticity (H):                inf   Skew:
1.25
Prob(H) (two-sided):                  0.00   Kurtosis:
4.34
==========================================================================
======

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (comple
x-step).
```


United States Solar (TWh) -ARIMA Prediction (1, 0, 1) On Transformation
Subtract Rolling Mean of Natural Logged United States Solar (TWh) -Historic
United States Solar (TWh) -ARIMA Prediction (1, 0, 1) On Transformation
TWh
Year
0.0
0.5
1.0
1.5
2.0
2.5
2005
2010
2015
2020
2025
2030
2035
2040
2045
2050


United States Solar (TWh) -ARIMA Prediction (1, 0, 1) on Original
Solar (TWh) -Historic
Solar (TWh) -ARIMA Prediction (1, 0, 1) on Original
TWh
Year
0
25
50
75
100
125
150
2000
2004
2008
2012
2016
2020

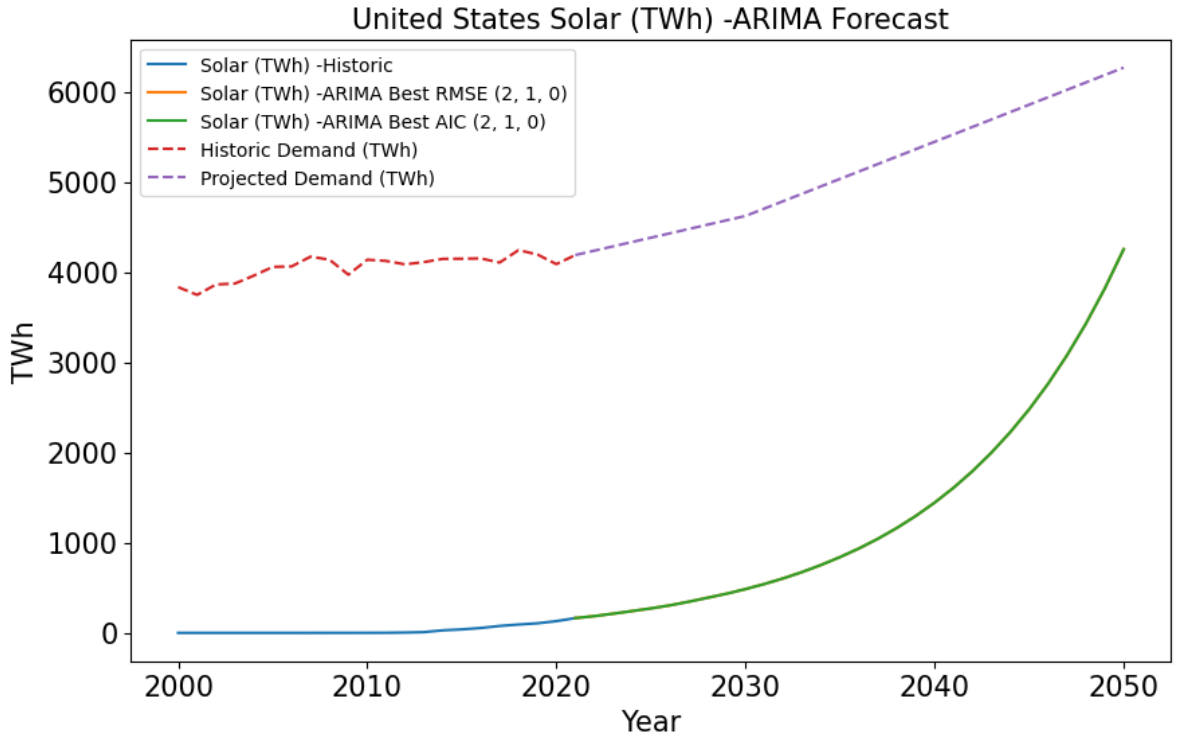


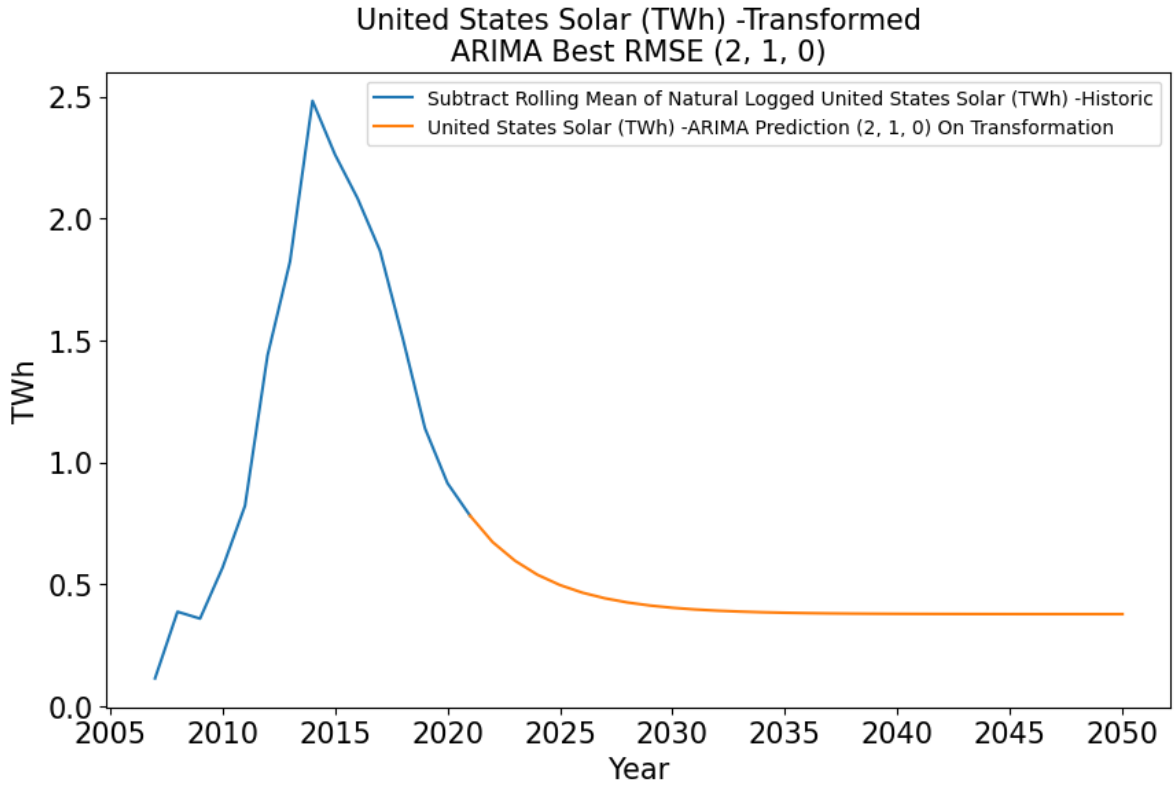
Solar is growing very but to product nearly thirty times the electricity needs of the united states would mean the entire surface of the planet would be one giant solar panel. No, not really, but this doesn't make sense regardless.

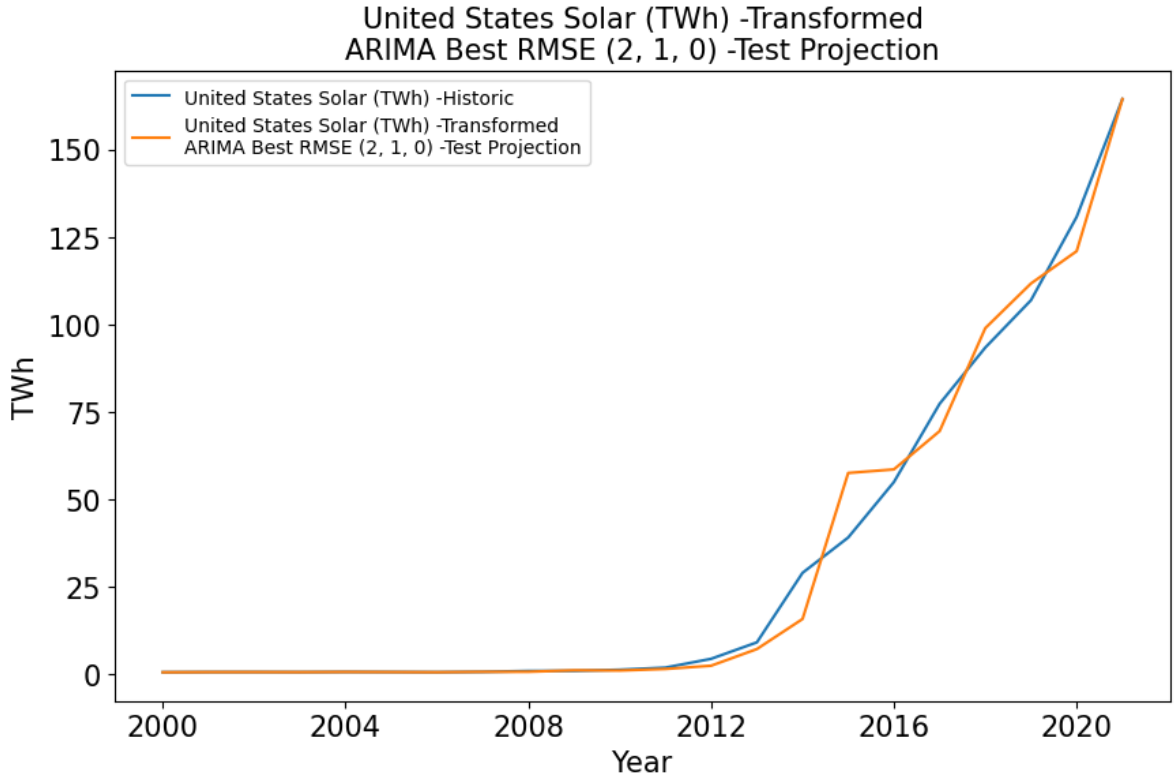
7.8.7 ARIMA Model and Grid Search

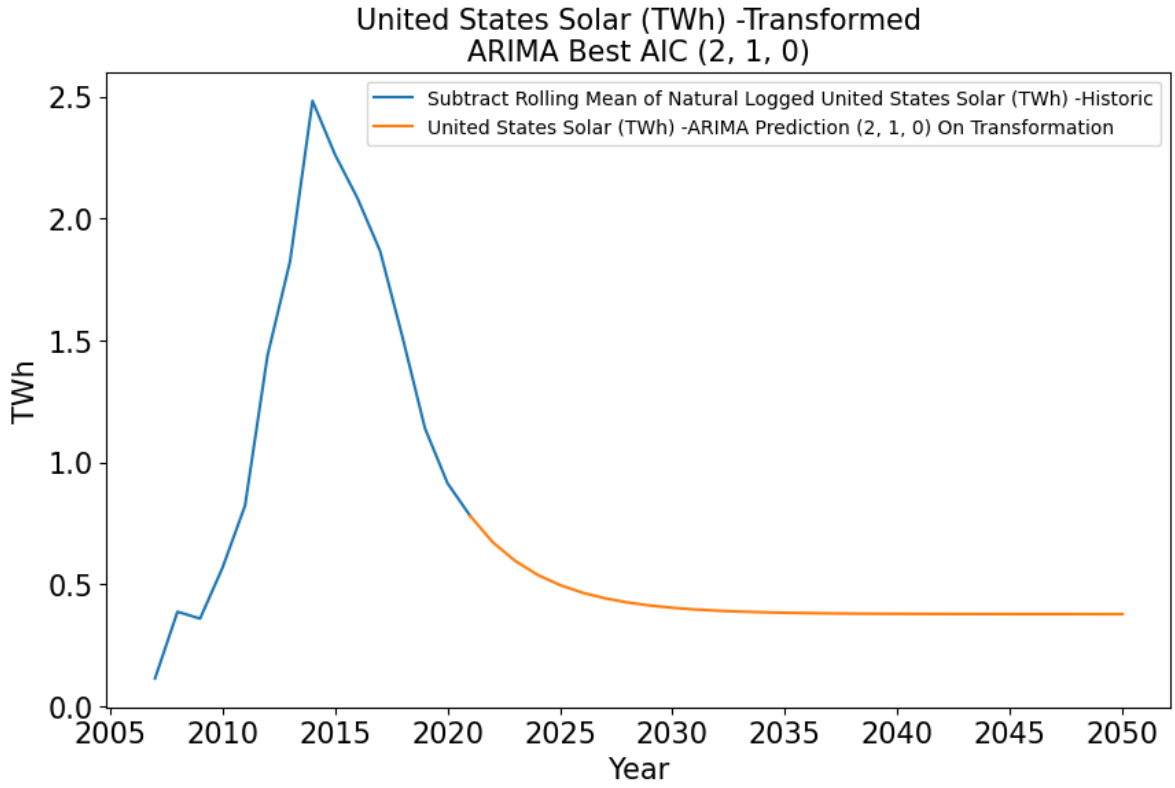
In [185]:
```
gas_rmse_cfg, gas_aic_cfg = arima_pdq(
                model_data_t.iloc[:,14:16], s=14, tran = 'inv_log_inv_sub', n=
```

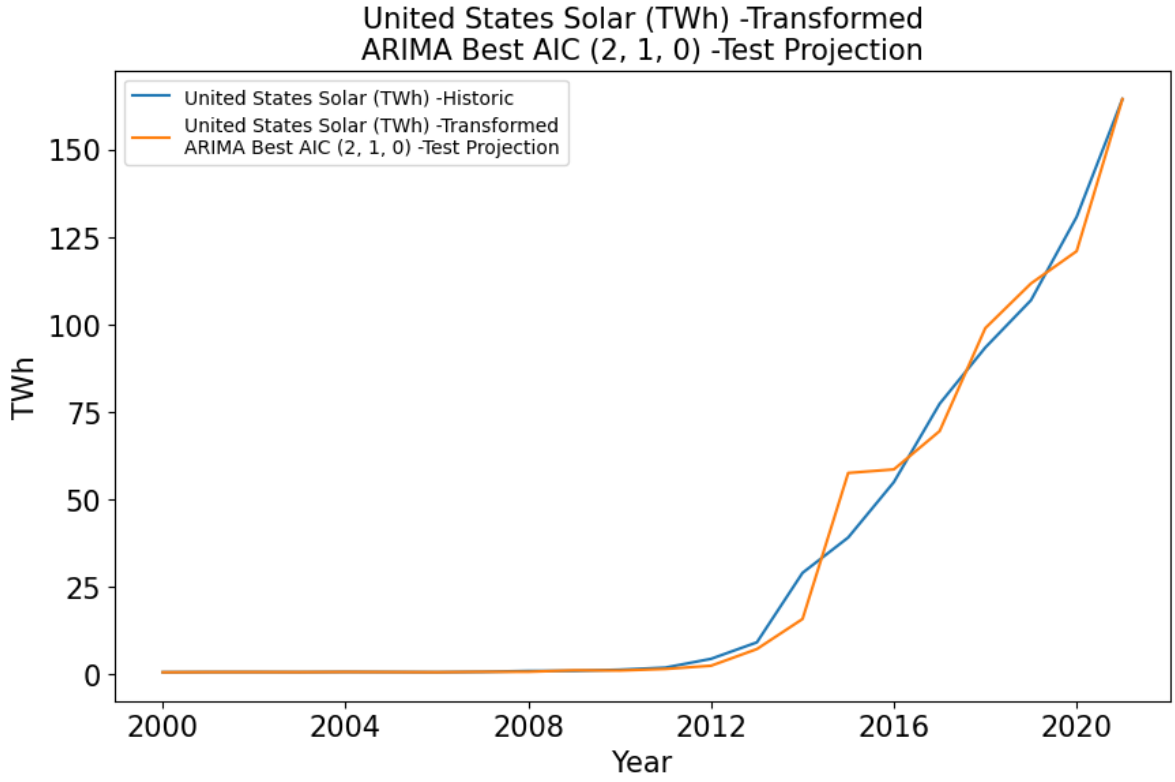
```
RMSE ARIMA: (1, 1, 1), RMSE= 8.33, AIC= 24.81, TWh-2050= 8694.89
RMSE ARIMA: (2, 1, 0), RMSE= 8.21, AIC= 24.42, TWh-2050= 4256.17
Best RMSE ARIMA: (2, 1, 0) RMSE= 8.21 AIC= 24.42, TWh-2050= 4256.17
Best AIC  ARIMA: (2, 1, 0) RMSE= 8.21 AIC= 24.42, TWh-2050= 4256.17
Graph_formatting produced error at (2, 1, 0) or (2, 1, 0)
```




United States Solar (TWh) -Transformed
ARIMA Best RMSE (2, 1, 0)
Subtract Rolling Mean of Natural Logged United States Solar (TWh) -Historic
United States Solar (TWh) -ARIMA Prediction (2, 1, 0) On Transformation
TWh
Year
0.0
0.5
1.0
1.5
2.0
2.5
2005
2010
2015
2020
2025
2030
2035
2040
2045
2050


United States Solar (TWh) -Transformed
ARIMA Best RMSE (2, 1, 0) -Test Projection
United States Solar (TWh) -Historic
United States Solar (TWh) -Transformed
ARIMA Best RMSE (2, 1, 0) -Test Projection
TWh
Year
0
25
50
75
100
125
150
2000
2004
2008
2012
2016
2020



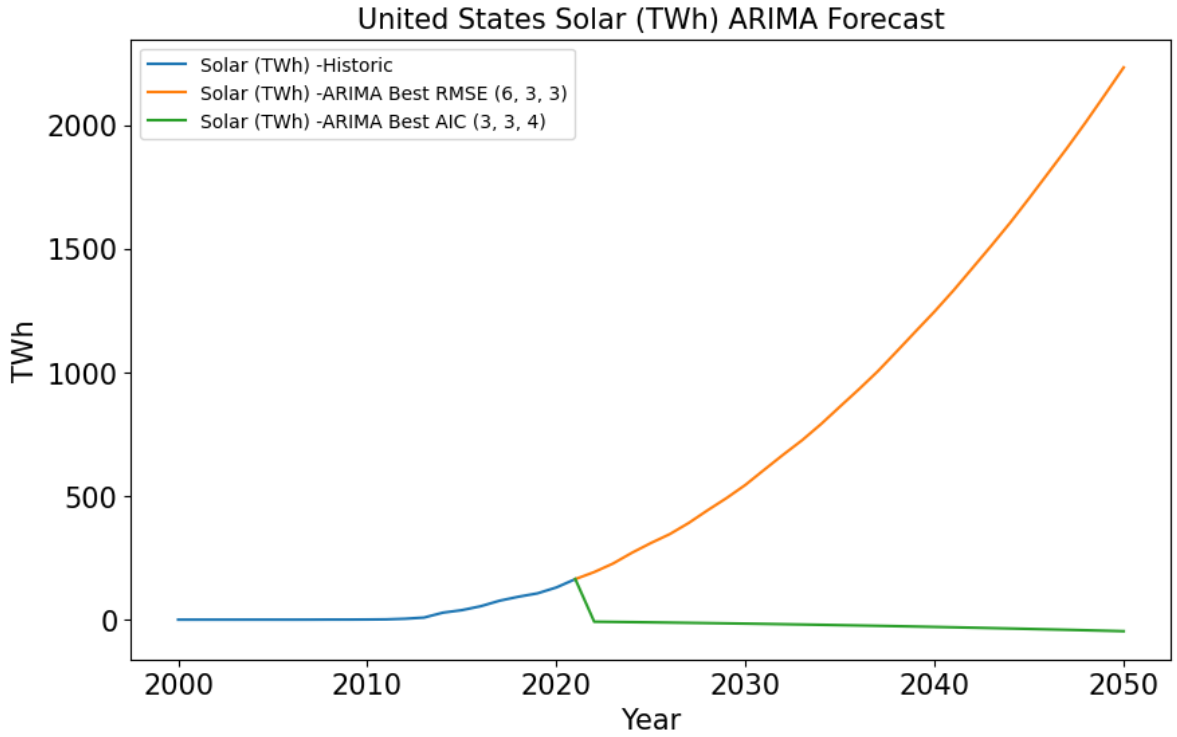


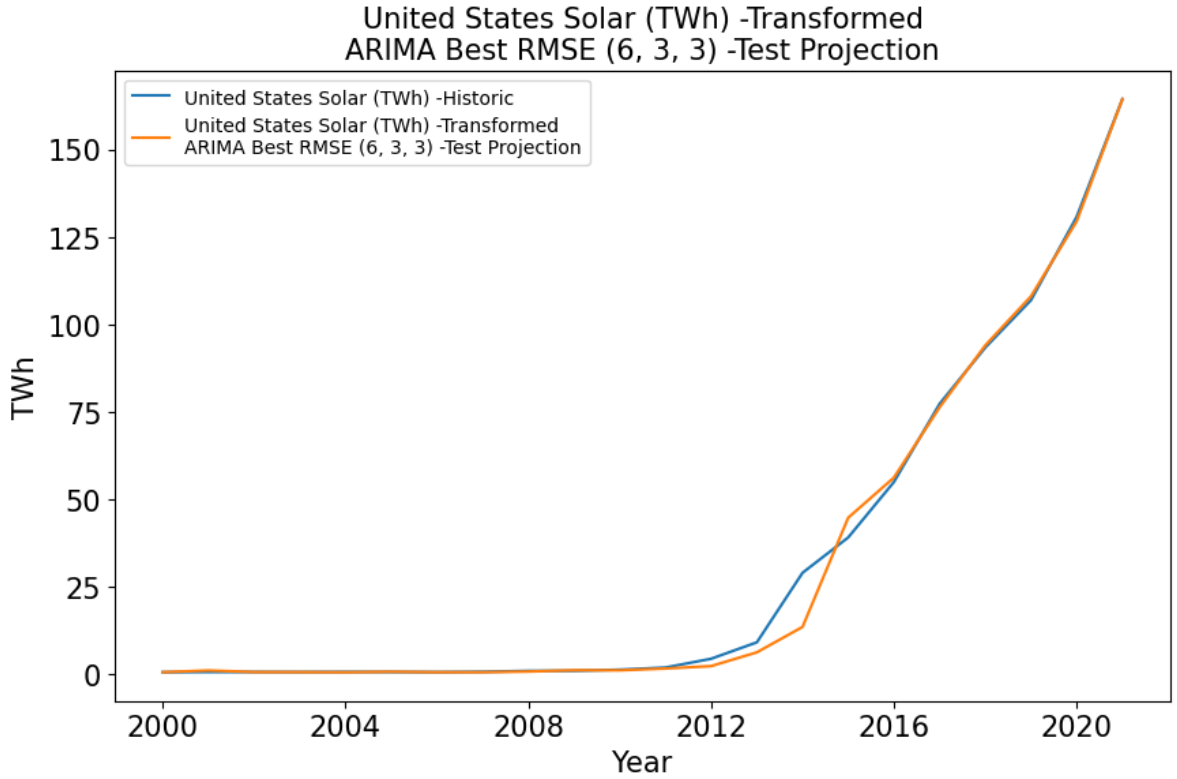
I'm glad that solar is on the rise, but it's doubtful that the United States will have enough solar energy to power the electricity needs of earth several times over by 2050. I'll see what the graph without transformations says.

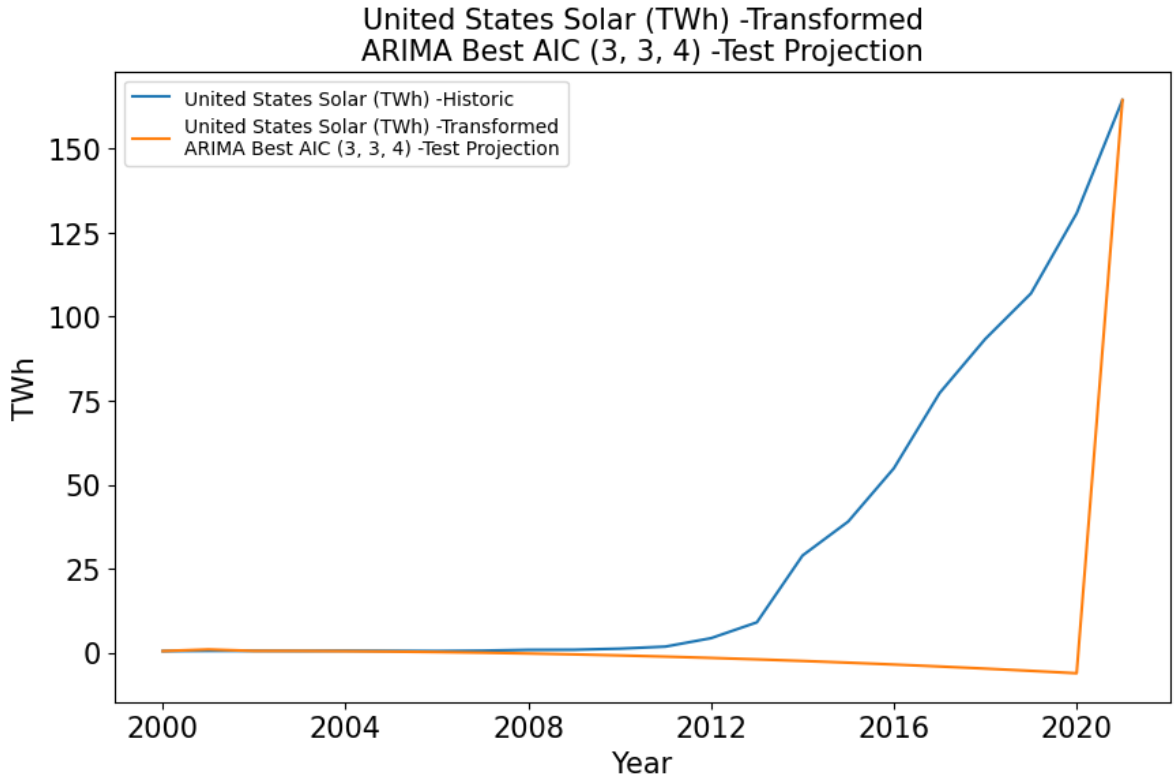
7.8.8 ARIMA Without Transformation

In [186]:
```
gas_rmse_cfg, gas_aic_cfg = arima_pdq_no_tran(
                                        model_data.iloc[:,7:8], 22, 50, s=14)
```

```
United States Solar (TWh) Grid Search:
RMSE ARIMA: (0, 0, 0), RMSE= 39.24, AIC= 237.45, TWh-2050= 32.64
RMSE ARIMA: (0, 0, 1), RMSE= 22.83, AIC= 216.59, TWh-2050= 35.02
RMSE ARIMA: (0, 0, 2), RMSE= 15.35, AIC= 201.82, TWh-2050= 37.30
RMSE ARIMA: (0, 0, 3), RMSE= 11.22, AIC= 187.04, TWh-2050= 39.18
RMSE ARIMA: (0, 0, 4), RMSE= 10.62, AIC= 185.46, TWh-2050= 40.56
RMSE ARIMA: (0, 0, 5), RMSE= 7.83, AIC= 178.97, TWh-2050= 45.24
RMSE ARIMA: (0, 0, 7), RMSE= 6.70, AIC= 176.10, TWh-2050= 49.32
RMSE ARIMA: (0, 0, 8), RMSE= 5.65, AIC= 173.79, TWh-2050= 36.79
AIC  ARIMA: (0, 1, 0), RMSE= 10.20, AIC= 168.58, TWh-2050= 164.42
AIC  ARIMA: (0, 1, 1), RMSE= 7.56, AIC= 155.52, TWh-2050= 178.93
AIC  ARIMA: (0, 1, 2), RMSE= 6.78, AIC= 151.78, TWh-2050= 192.79
RMSE ARIMA: (0, 1, 3), RMSE= 5.56, AIC= 151.55, TWh-2050= 215.53
RMSE ARIMA: (0, 1, 4), RMSE= 5.08, AIC= 149.96, TWh-2050= 230.73
RMSE ARIMA: (0, 1, 5), RMSE= 5.07, AIC= 148.09, TWh-2050= 260.43
RMSE ARIMA: (0, 1, 7), RMSE= 4.49, AIC= 147.83, TWh-2050= 313.42
AIC  ARIMA: (0, 2, 0), RMSE= 5.06, AIC= 128.70, TWh-2050= 1141.72
RMSE ARIMA: (0, 2, 3), RMSE= 4.48, AIC= 132.60, TWh-2050= 1083.48
RMSE ARIMA: (0, 2, 7), RMSE= 4.02, AIC= 134.36, TWh-2050= 1183.10
AIC  ARIMA: (0, 3, 1), RMSE= 5.13, AIC= 126.62, TWh-2050= 2007.65
RMSE ARIMA: (0, 3, 6), RMSE= 3.86, AIC= 129.39, TWh-2050= 3207.19
RMSE ARIMA: (0, 3, 7), RMSE= 3.77, AIC= 131.17, TWh-2050= 2969.54
RMSE ARIMA: (2, 3, 3), RMSE= 3.76, AIC= 128.36, TWh-2050= 2246.43
RMSE ARIMA: (3, 2, 3), RMSE= 3.72, AIC= 134.29, TWh-2050= 1602.79
AIC  ARIMA: (3, 3, 4), RMSE= 49.50, AIC= 16.00, TWh-2050= -46.01
RMSE ARIMA: (4, 2, 3), RMSE= 3.72, AIC= 135.96, TWh-2050= 1599.08
RMSE ARIMA: (5, 3, 6), RMSE= 3.66, AIC= 136.77, TWh-2050= 2551.67
RMSE ARIMA: (6, 3, 3), RMSE= 3.63, AIC= 133.48, TWh-2050= 2232.39
Best RMSE ARIMA: (6, 3, 3) RMSE= 3.63 AIC= 133.48, TWh-2050= 2232.39
Best AIC  ARIMA: (3, 3, 4) RMSE= 49.50 AIC= 16.00, TWh-2050= -46.01
```


United States Solar (TWh) ARIMA Forecast
Solar (TWh) -Historic
Solar (TWh) -ARIMA Best RMSE (6, 3, 3)
Solar (TWh) -ARIMA Best AIC (3, 3, 4)
TWh
Year
0
500
1000
1500
2000
2000
2010
2020
2030
2040
2050


United States Solar (TWh) -Transformed
ARIMA Best RMSE (6, 3, 3) -Test Projection
United States Solar (TWh) -Historic
United States Solar (TWh) -Transformed
ARIMA Best RMSE (6, 3, 3) -Test Projection
TWh
Year
0
25
50
75
100
125
150
2000
2004
2008
2012
2016
2020



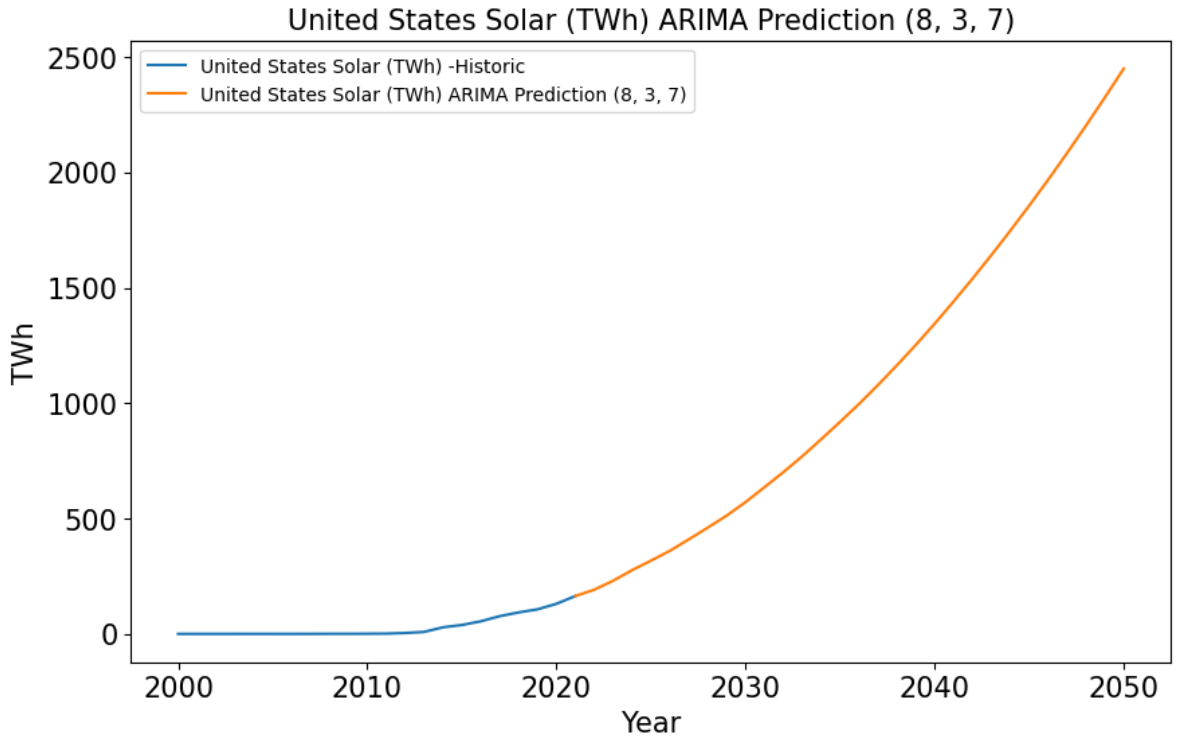
7.8.9 Model Selection

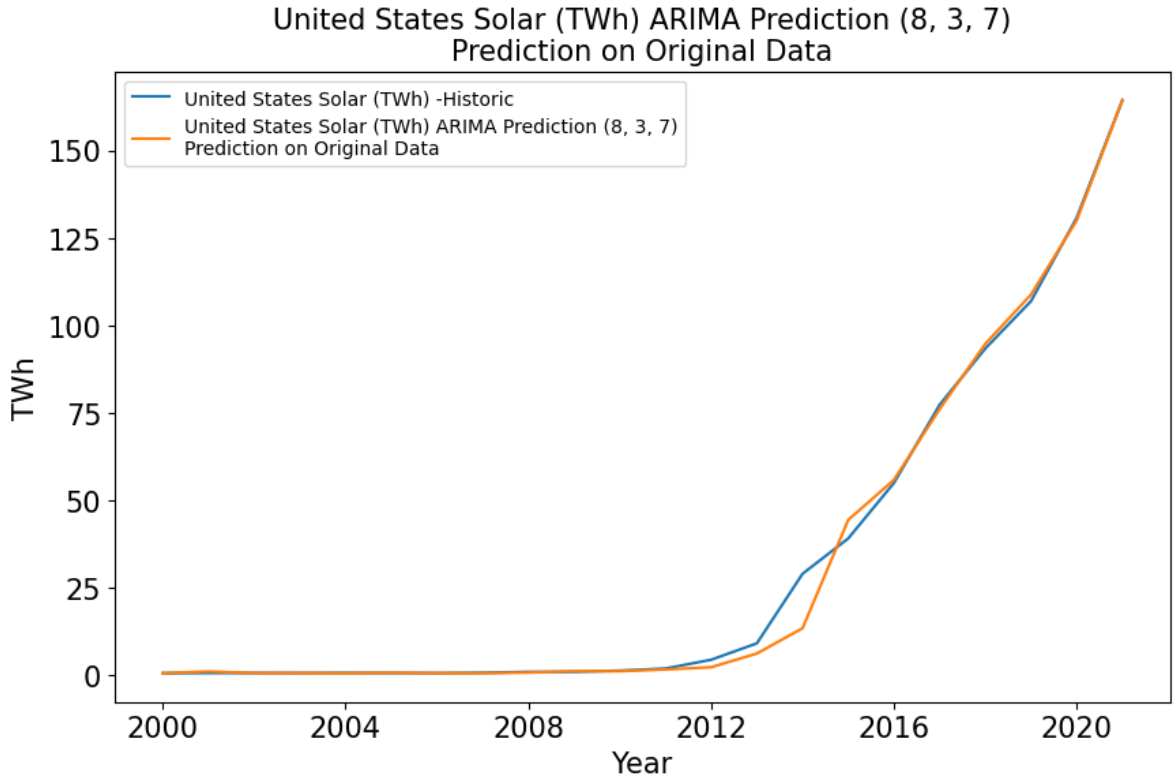
In [187]:
```
#Deleteme
stored_solar = selected_models.copy()
```

In [188]:
```
selected_models = stored_solar.copy() #deleteme
df_solar = model_data_t.iloc[:,14:15]

#                        Model,   df_o,    pdq,     tran, n, PDQs
selected_models.append(['ARIMA', df_solar, (8,3,7), None, 0, None])

m = 7
model_summary_no_tran(selected_models[m][1], selected_models[m][2], 22, 50);
```

```
ARIMA: (8, 3, 7), RMSE=3.70, AIC=143.70
**********************************************************************
**
United States Solar (TWh) -ARIMA Prediction (8, 3, 7) model results.

                                SARIMAX Results
==========================================================================
========
Dep. Variable:     United States Solar (TWh)   No. Observations:
22
Model:                         ARIMA(8, 3, 7)   Log Likelihood
-55.850
Date:                       Sat, 29 Apr 2023   AIC
143.701
Time:                               06:25:20   BIC
158.812
Sample:                           01-01-2000   HQIC
146.258
                                - 01-01-2021
Covariance Type:                         opg
==========================================================================
=
                 coef    std err          z      P>|z|      [0.025      0.97
5]
--------------------------------------------------------------------------
-
ar.L1         -1.4376    264.626     -0.005      0.996    -520.095     517.21
9
ar.L2         -1.0602    244.504     -0.004      0.997    -480.279     478.15
9
ar.L3         -0.4261    168.242     -0.003      0.998    -330.174     329.32
1
ar.L4         -0.2944     79.446     -0.004      0.997    -156.006     155.41
7
ar.L5         -0.6571    142.573     -0.005      0.996    -280.095     278.78
0
ar.L6         -0.7612    145.471     -0.005      0.996    -285.880     284.35
7
ar.L7         -0.4818    175.410     -0.003      0.998    -344.280     343.31
6
ar.L8         -0.0623     24.415     -0.003      0.998     -47.916      47.79
1
ma.L1         -0.0739   1.47e+04  -5.03e-06      1.000   -2.88e+04    2.88e+0
4
ma.L2         -0.3311   8528.933  -3.88e-05      1.000   -1.67e+04    1.67e+0
4
ma.L3          0.1668   3263.248   5.11e-05      1.000   -6395.681    6396.01
5
ma.L4          0.0409   3384.213   1.21e-05      1.000   -6632.894    6632.97
6
ma.L5         -0.3100   9541.480  -3.25e-05      1.000   -1.87e+04    1.87e+0
4
ma.L6          0.0141   1.38e+04   1.02e-06      1.000   -2.71e+04    2.71e+0
4
ma.L7          0.9358   8519.635      0.000      1.000   -1.67e+04    1.67e+0
4
```

```
sigma2        11.0310   1.02e+05      0.000      1.000      -2e+05       2e+0
5
================================================================================
======
Ljung-Box (L1) (Q):                   0.37   Jarque-Bera (JB):
71.82
Prob(Q):                              0.54   Prob(JB):
0.00
Heteroskedasticity (H):             544.14   Skew:
2.59
Prob(H) (two-sided):                  0.00   Kurtosis:
11.00
================================================================================
======

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (comple
x-step).
```





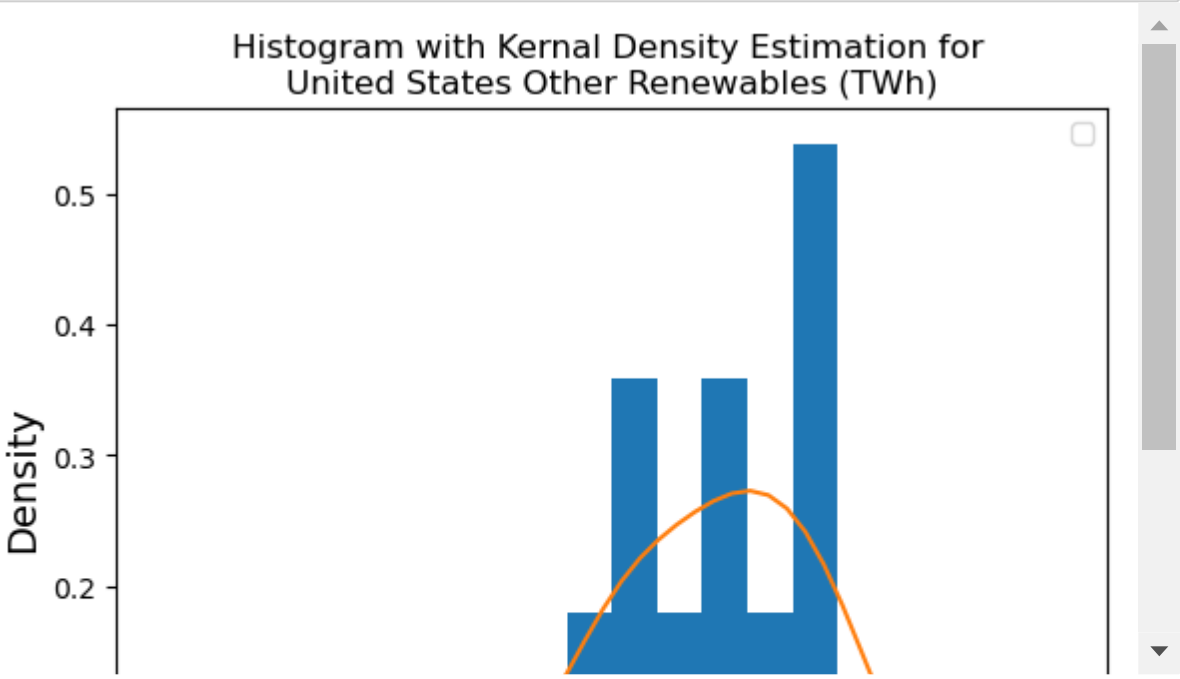
Order (8,3,8) seemed to be an outlier. Most of the solar curves guessed the level of TWh for solar by 2050 around 2500 TWh. (8,3,8) was the only one that pushed that number past 2600 that had a good RMSE, and to do it by a whole thousand? That seems wishful. Also, the RMSE between the best RMSE (8,3,8) and the second best (8,3,7) was less than a hundredth decimal place.

This one did not rely on transformations. All the graphs with transformation gave unreasonable high estimates, like solar had several times the power needs of the entire earth's economy by 2050. I'm happy with this graph as it is.

7.9 Other Renewables

7.9.1 Distribution Investigation

In [189]:
```
hist(model_data.iloc[:,8:9])
```



In [190]:
```
stats_block = s_block(model_data.iloc[:,8:9], stats_block)
stats_block
```

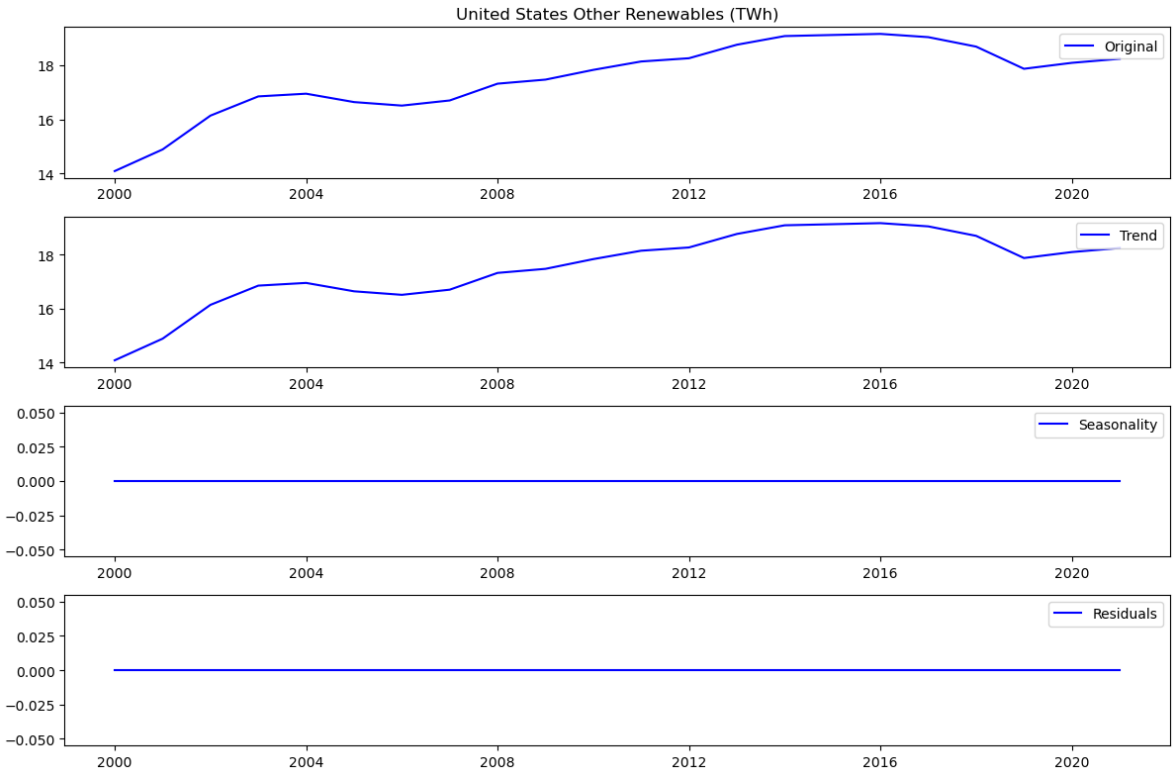
Out[190]:

	Dataset	Mean	Median	Standard Deviation	Skew	Fisher Kurtosis
0	United States Coal (TWh)	1,607.17	1,744.66	399.15	-0.68	-0.90
1	United States Gas (TWh)	1,060.39	1,000.70	325.82	0.29	-1.21
2	United States Oil (TWh)	67.20	45.97	36.59	0.89	-0.92
3	United States Nuclear (TWh)	790.04	790.04	15.55	-0.64	-0.53
4	United States Hydro (TWh)	264.36	263.82	21.08	-0.22	1.03
5	United States Bioenergy (TWh)	57.18	55.81	3.92	0.35	-0.98
6	United States Wind (TWh)	132.63	107.42	116.84	0.57	-0.95
7	United States Solar (TWh)	32.64	1.52	48.75	1.39	0.67
8	United States Other Renewables (TWh)	17.54	17.85	1.33	-0.88	0.30

This data is negatively skewed, but also the kurtosis is closest to zero, so it closely resembles standard deviation other than the negative skew.

7.9.2 Decomposing The Data

In [191]:
```
decomp_graph(model_data.iloc[:,8:9])
```

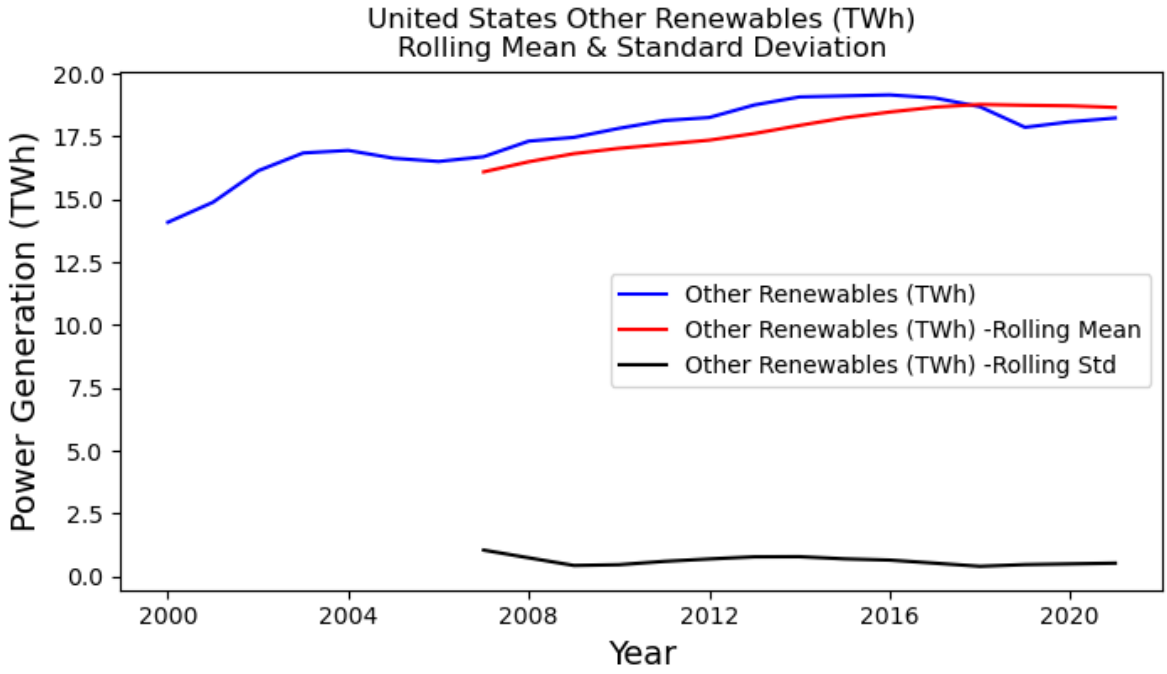


No seasonality or residuals. I'll check for stationarity.

7.9.3 Checking for Stationarity and Flattening

In [192]:
```
pd.set_option('display.max_rows', 30)
stationarity_check(model_data.iloc[:,8:9], s=14)
```

```
United States Other Renewables (TWh)
Results of Dickey-Fuller test:

Test Statistic                -2.38
p-value                        0.15
#Lags Used                     1.00
Number of Observations Used   20.00
Critical Value (1%)           -3.81
Critical Value (5%)           -3.02
Critical Value (10%)          -2.65
dtype: float64
```



This is close to stationary. After playing around with transformations, I found that if I took the plotted the difference and then subtracted the rolling average of six values, I can get it stationary.

In [193]:
```
stored = model_data_t.copy()
```

In [194]:
```
model_data_t = stored.copy()
t = subtract_roll_mean(difference(model_data.iloc[:,8:9]), n=6)
model_data_t.insert(17,t.columns[0],t)

model_data_t.iloc[:,17:18]
```

Out[194]:

Year	Subtract Rolling Mean of Annual Change of United States Other Renewables (TWh)
2000-01-01	NaN
2001-01-01	NaN
2002-01-01	NaN
2003-01-01	NaN
2004-01-01	NaN
2005-01-01	NaN
2006-01-01	-0.53
2007-01-01	-0.11
2008-01-01	0.42
2009-01-01	0.05
2010-01-01	0.21
2011-01-01	0.06
2012-01-01	-0.17
2013-01-01	0.16
2014-01-01	0.03
2015-01-01	-0.23
2016-01-01	-0.18
2017-01-01	-0.27
2018-01-01	-0.42
2019-01-01	-0.67
2020-01-01	0.38
2021-01-01	0.30

Before I move forward, I want to make sure I can transform this data back after modeling.

In [195]:
```
test = model_data_t.iloc[:,16:18].copy()
test.iloc[:,1:2] = inv_diff_inv_sub(test,n=6)
test
```

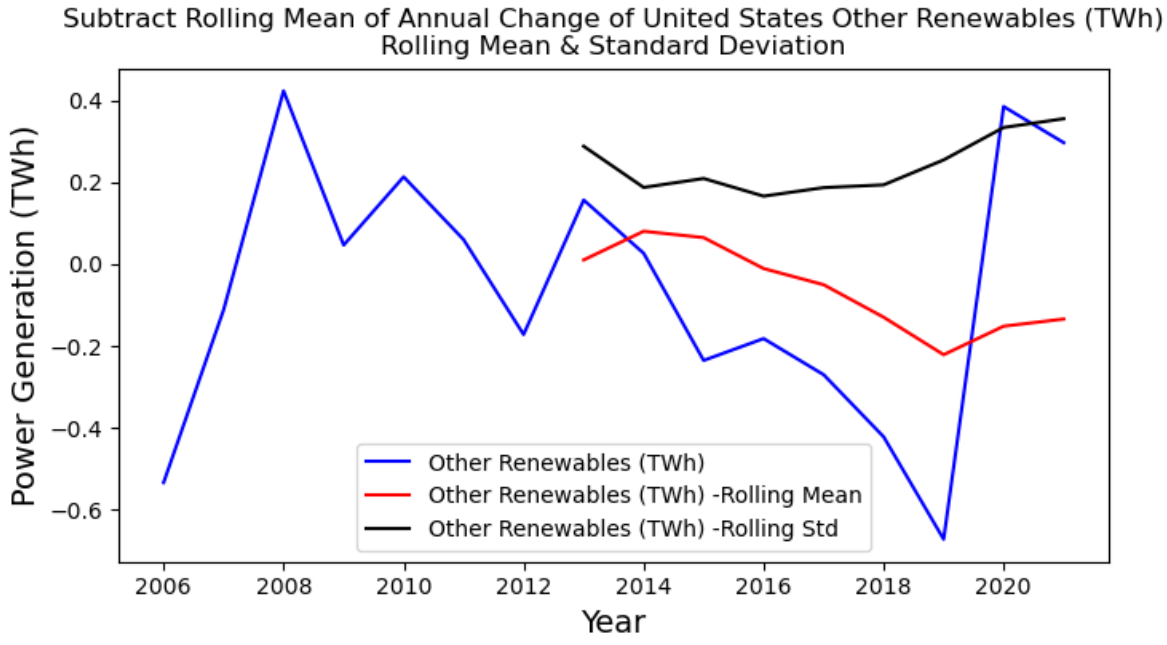
Out[195]:

Year	United States Other Renewables (TWh)	Subtract Rolling Mean of Annual Change of United States Other Renewables (TWh)
2000-01-01	14.09	14.09
2001-01-01	14.89	14.89
2002-01-01	16.14	16.14
2003-01-01	16.85	16.85
2004-01-01	16.95	16.95
2005-01-01	16.64	16.64
2006-01-01	16.51	16.51
2007-01-01	16.70	16.70
2008-01-01	17.32	17.32
2009-01-01	17.47	17.47
2010-01-01	17.83	17.83
2011-01-01	18.14	18.14
2012-01-01	18.26	18.26
2013-01-01	18.76	18.76
2014-01-01	19.08	19.08
2015-01-01	19.12	19.12
2016-01-01	19.16	19.16
2017-01-01	19.04	19.04
2018-01-01	18.69	18.69
2019-01-01	17.87	17.87
2020-01-01	18.09	18.09
2021-01-01	18.24	18.24

Transferring back is a possibility. Therefore, we can move forward. I'll again check for stationarity.

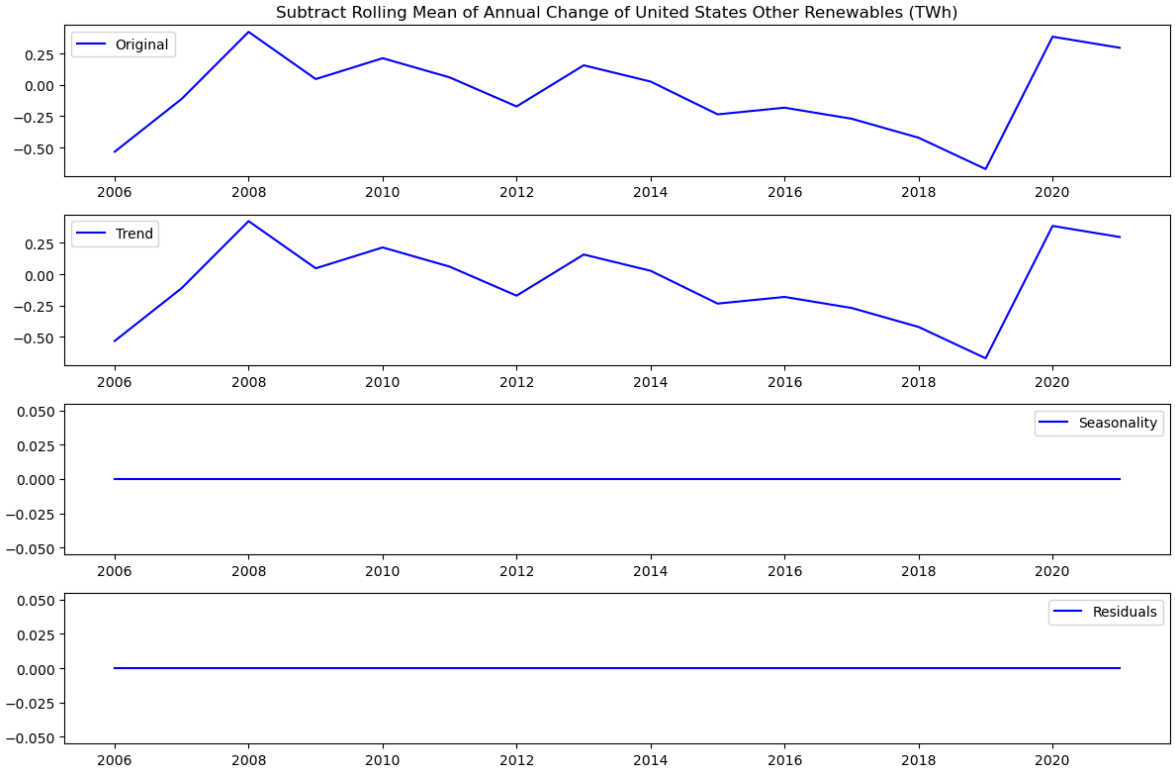
In [196]:
```
stationarity_check(model_data_t.iloc[:,17:18], s=56)
```

```
Subtract Rolling Mean of Annual Change of United States Other Renewables (TW
h)
Results of Dickey-Fuller test:

Test Statistic                -2.97
p-value                        0.04
#Lags Used                     0.00
Number of Observations Used   15.00
Critical Value (1%)           -3.96
Critical Value (5%)           -3.08
Critical Value (10%)          -2.68
dtype: float64
```



My p-value is less than .05. Stationarity achieved. I'll look at the decomposition graph again.

In [197]:
```
decomp_graph(model_data_t.iloc[:,17:18].dropna())
```

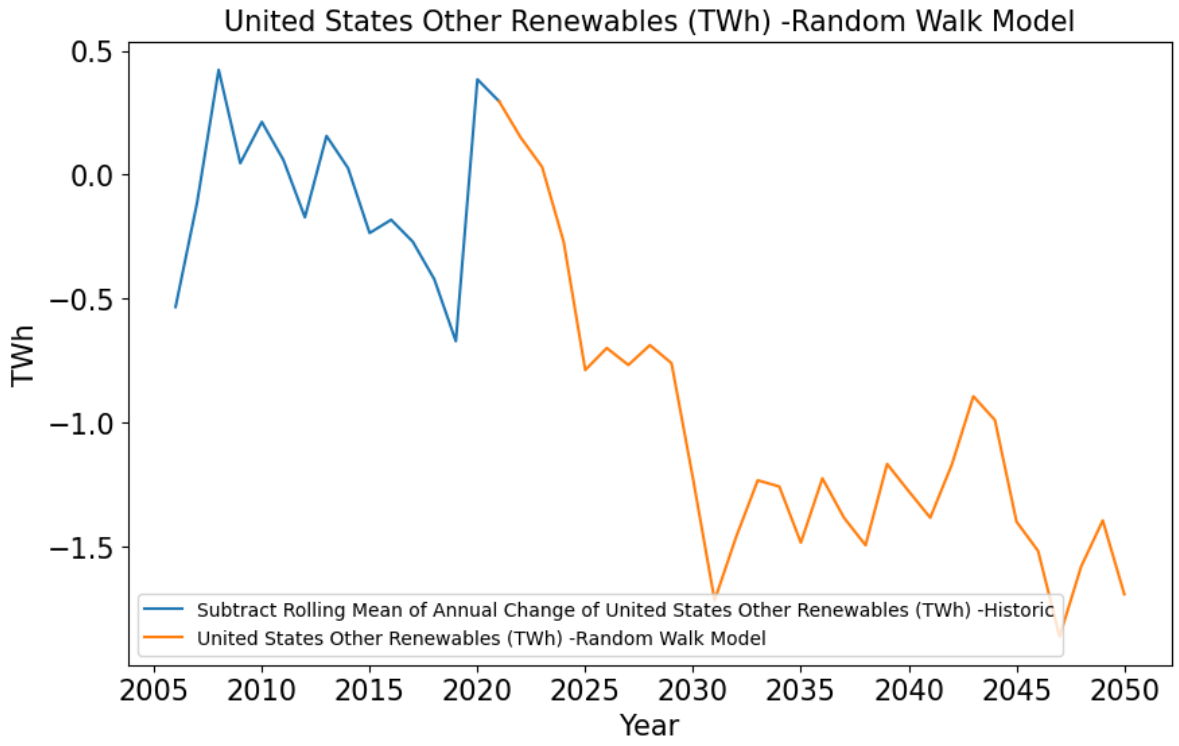


Looks like there's still a trend line, but it does pass stationarity, so I'm moving forward to the baseline random walk model.

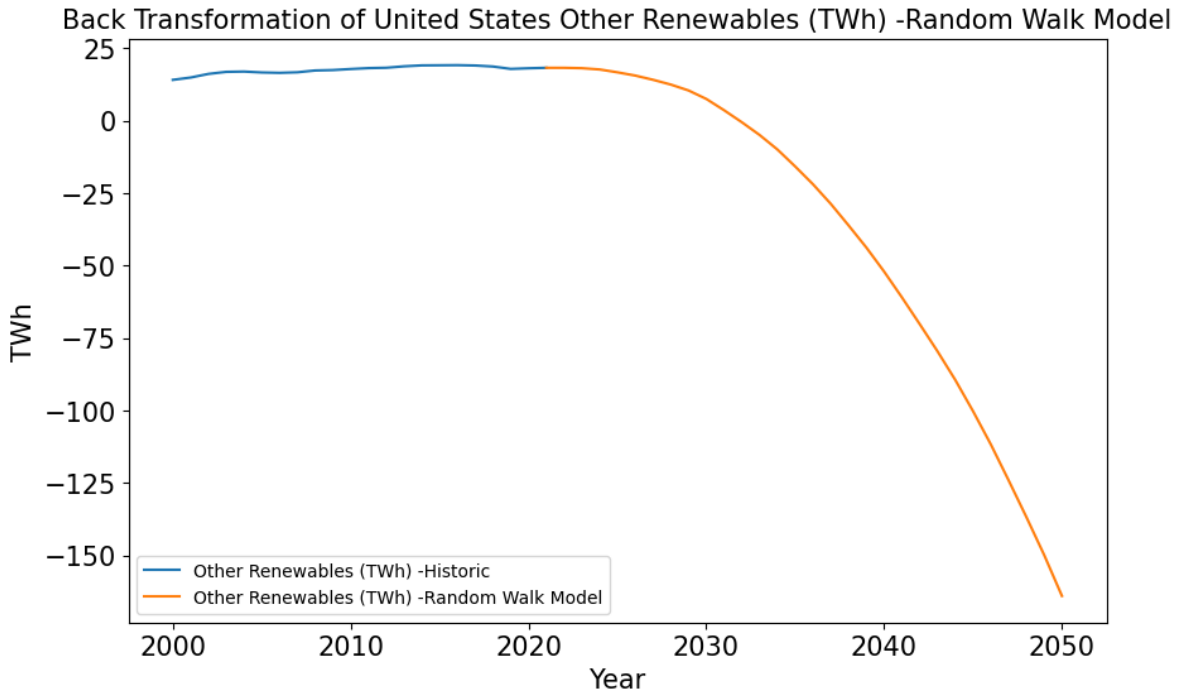
7.9.4 Random Walk Model

In [198]:
```
pd.set_option('display.max_rows',None)
y_hat_proj = random_walk(model_data_t.iloc[:,17:18])
y_hat_proj.columns = [model_data.columns[8]+' -Random Walk Model']

# Plot the transformed Random Walk
pred_graph(model_data_t.iloc[:,17:18], y_hat_proj.iloc[:,0:1])
```

```
Random Walk with Standard Deviation of Transformed Data set: 0.31
```



In [199]:
```
# Prepare the DataFrame for Back Transformation
ranplot = model_data_t.iloc[:,16:18].copy()
ranplot.rename(columns={str(ranplot.columns[1]): str(y_hat_proj.columns[0])},
               inplace=True)
ranplot = pd.concat([ranplot,y_hat_proj],axis=0)

# Perform Back Transformation
y_hat_proj = inv_diff_inv_sub(ranplot, n=6)

#Plot the new graph
pred_graph(model_data.iloc[:,8:9], y_hat_proj.iloc[22:,0:1], 14)
```

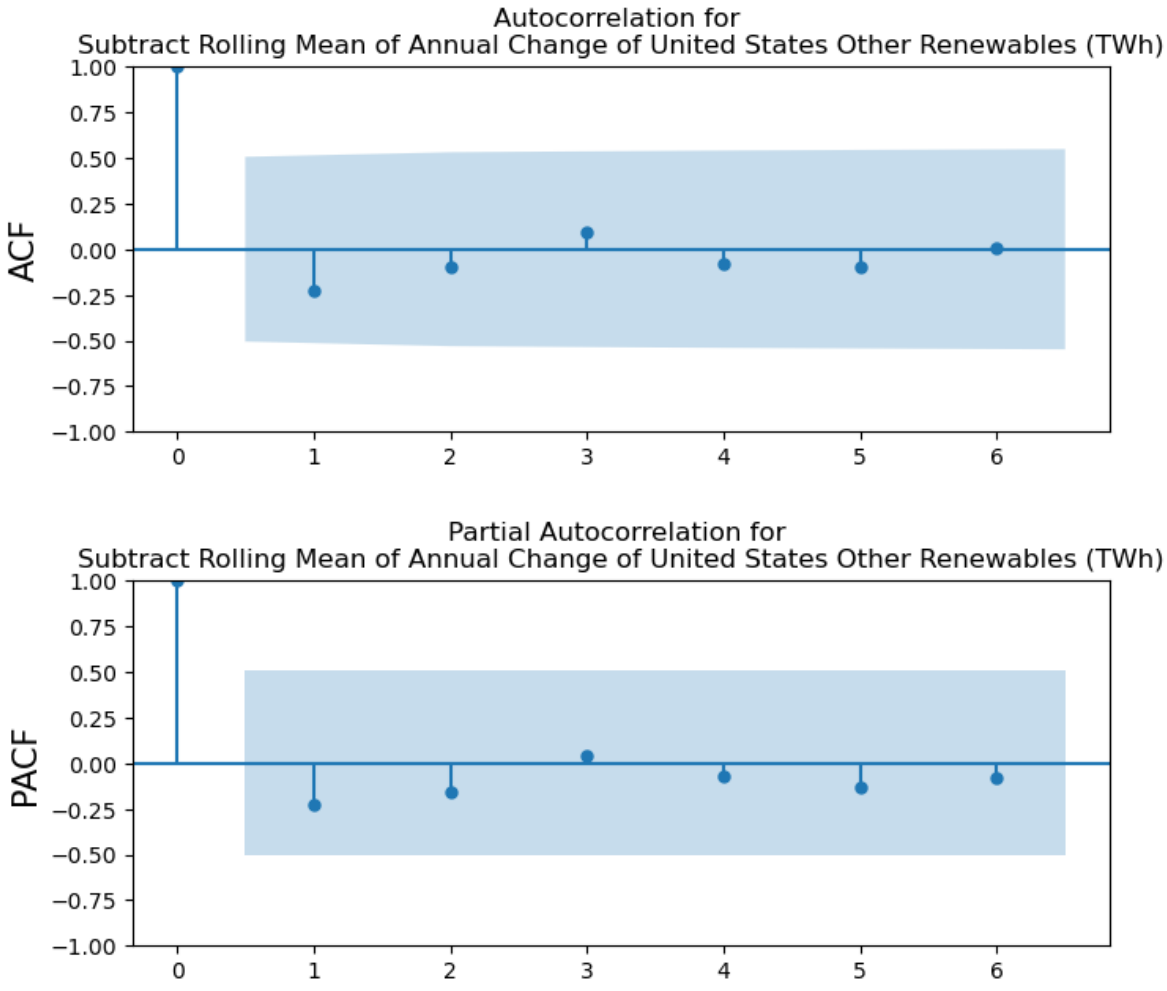


This model is as unreliable as it's name would imply, random walk. This one does ok because of the low standard deviation.

Now I'll move forward with an ARMA model. First, I need to evaluate the ACF and PACF plots to find the best Autoregressive and Moving Average terms.

7.9.5 Evaluating Initial AR and MA Values with ACF and PACF Plots

In [200]:
```
plotacf(model_data_t.iloc[:,17:18], lags = 6)
plotpacf(model_data_t.iloc[:,17:18], lags = 6)
```

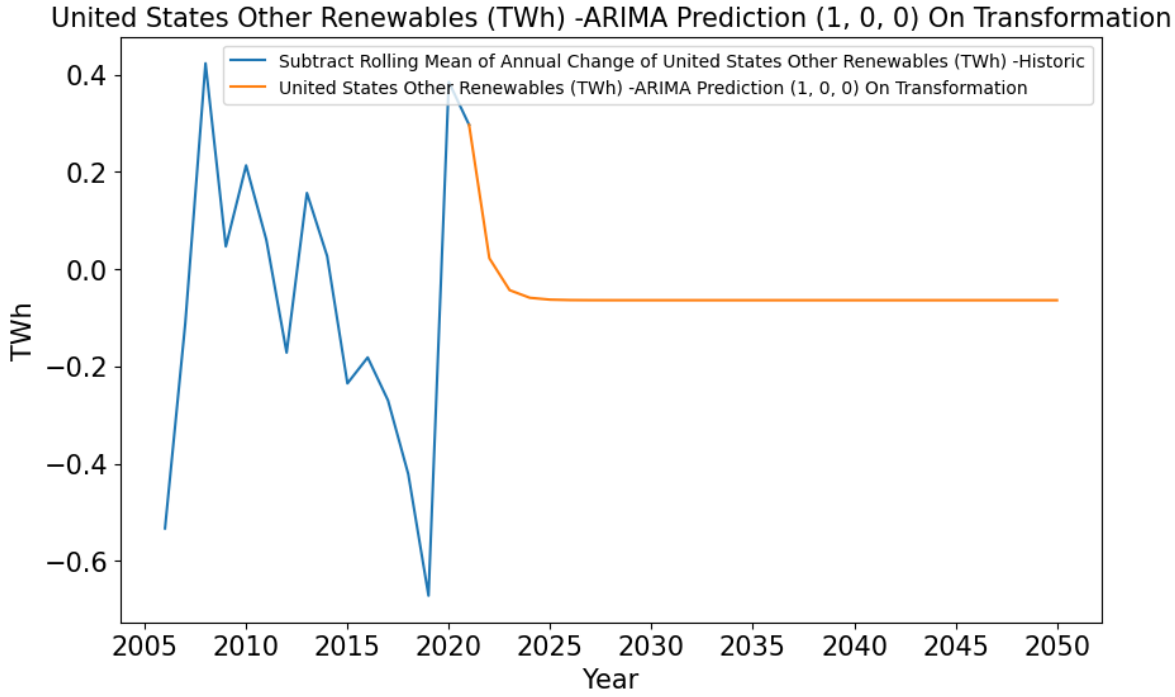


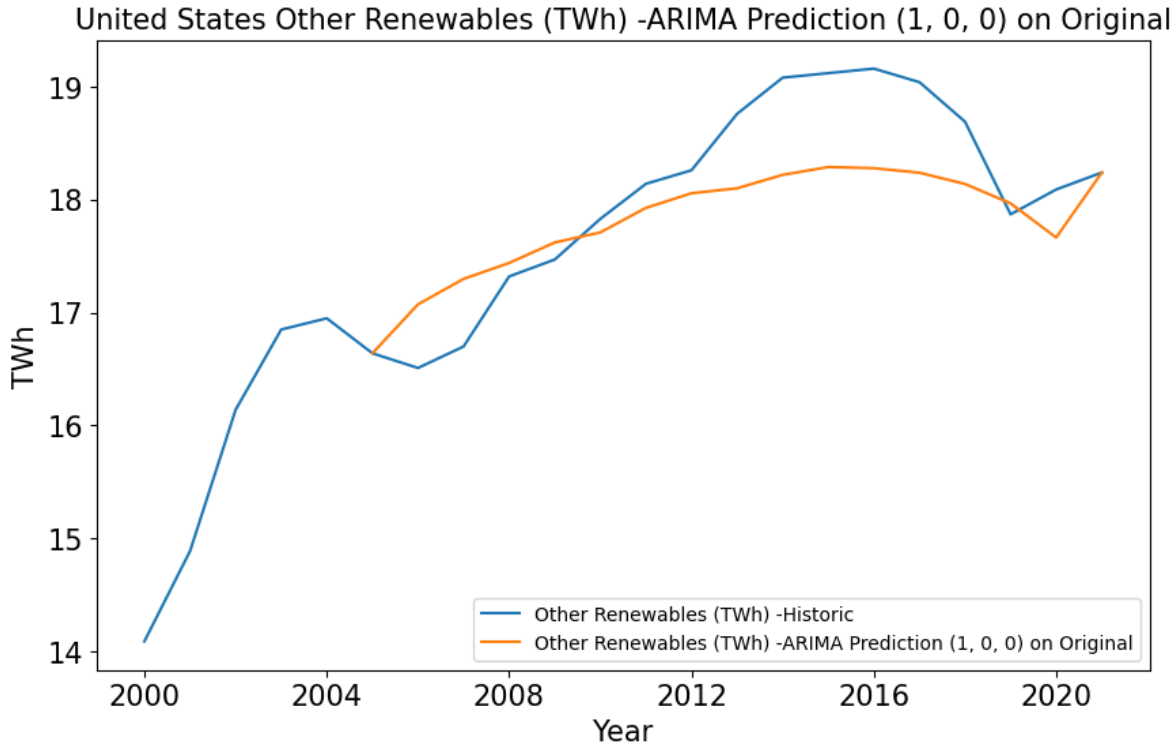
No breaches at all. I'll try (1,0,0) for the first model.

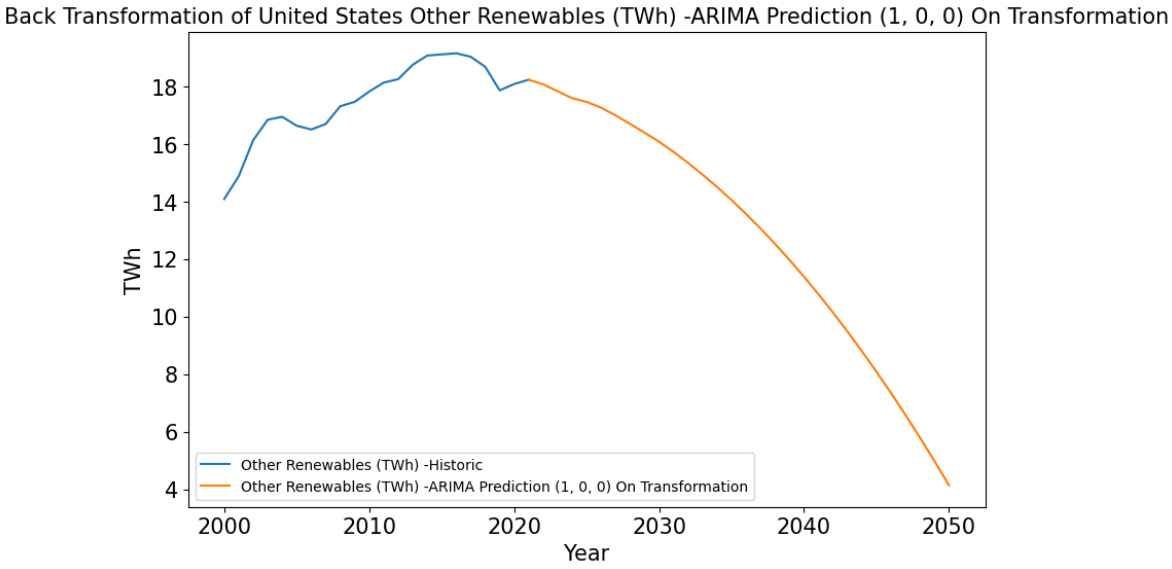
7.9.6 ARMA Model

In [201]:
```
pd.set_option('display.max_rows',60)
order = (1,0,0)
model_summary(model_data_t.iloc[:,16:18], order, 22, 50, 14,
          tran = 'inv_diff_inv_sub', n=6)
```

```
ARIMA: (1, 0, 0), RMSE=0.56, AIC=13.24
**********************************************************************
**
United States Other Renewables (TWh) model results.

                                                                SARIMAX Results
==============================================================================
==============================================================
Dep. Variable:     Subtract Rolling Mean of Annual Change of United States Ot
her Renewables (TWh)   No. Observations:                   22
Model:
ARIMA(1, 0, 0)   Log Likelihood                  -3.621
Date:
Sat, 29 Apr 2023   AIC                             13.242
Time:
06:25:24   BIC                              16.515
Sample:
01-01-2000   HQIC                             14.013

- 01-01-2021
Covariance Type:
opg
==============================================================================
=
                 coef    std err          z      P>|z|      [0.025      0.97
5]
------------------------------------------------------------------------------
-
const         -0.0639      0.120     -0.531      0.595      -0.300       0.17
2
ar.L1          0.2395      0.281      0.853      0.394      -0.311       0.79
0
sigma2         0.0917      0.041      2.249      0.025       0.012       0.17
2
==============================================================================
======
Ljung-Box (L1) (Q):                   0.04   Jarque-Bera (JB):
0.24
Prob(Q):                              0.84   Prob(JB):
0.89
Heteroskedasticity (H):               4.14   Skew:
0.20
Prob(H) (two-sided):                  0.08   Kurtosis:
3.33
==============================================================================
======

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (comple
x-step).
```


United States Other Renewables (TWh) -ARIMA Prediction (1, 0, 0) On Transformation
Subtract Rolling Mean of Annual Change of United States Other Renewables (TWh) -Historic
United States Other Renewables (TWh) -ARIMA Prediction (1, 0, 0) On Transformation
TWh
Year
0.4
0.2
0.0
−0.2
−0.4
−0.6
2005
2010
2015
2020
2025
2030
2035
2040
2045
2050


United States Other Renewables (TWh) -ARIMA Prediction (1, 0, 0) on Original
Other Renewables (TWh) -Historic
Other Renewables (TWh) -ARIMA Prediction (1, 0, 0) on Original
TWh
Year
19
18
17
16
15
14
2000
2004
2008
2012
2016
2020

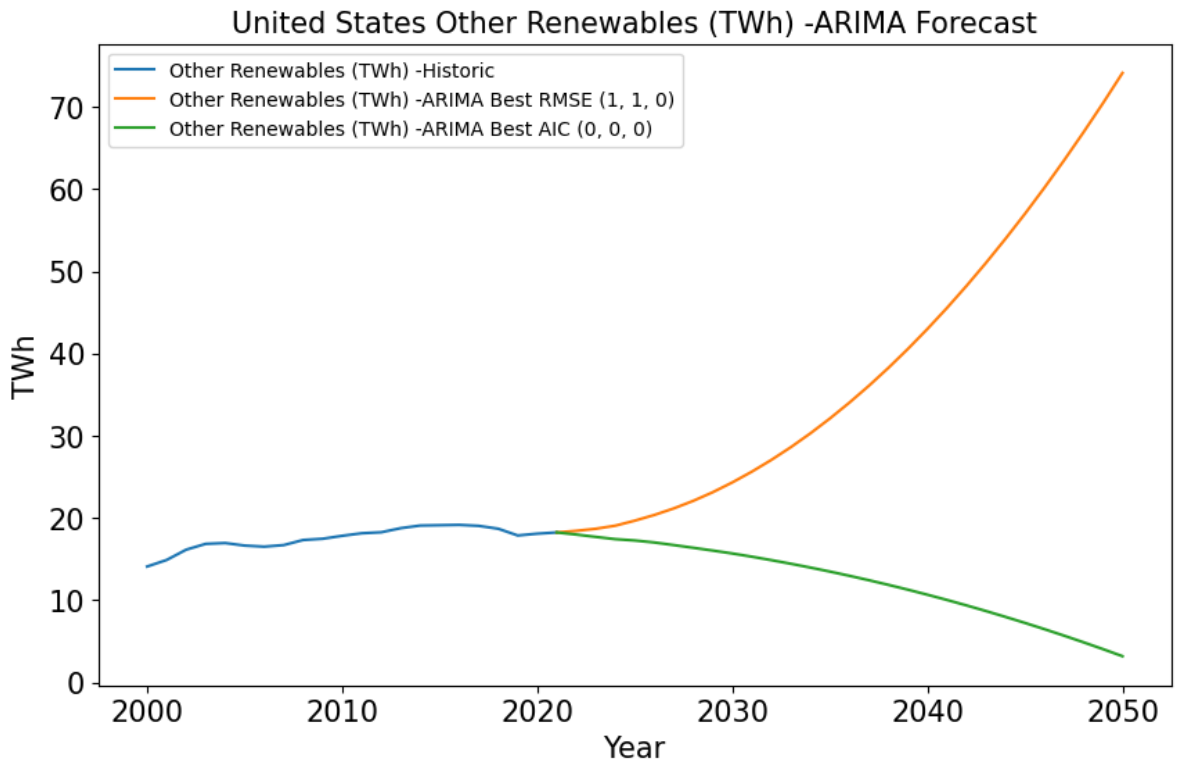


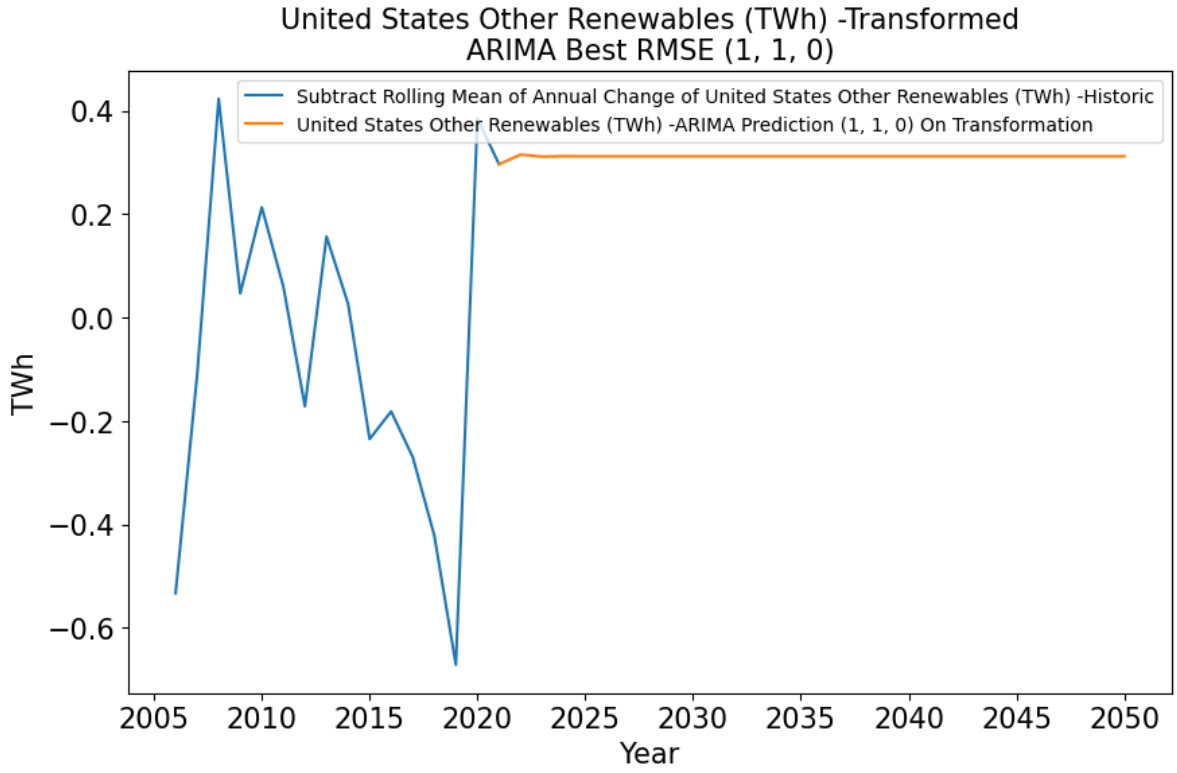
I don't think we'll see drop in non-wind/solar renewables. I'll see what the Grid Search yields.

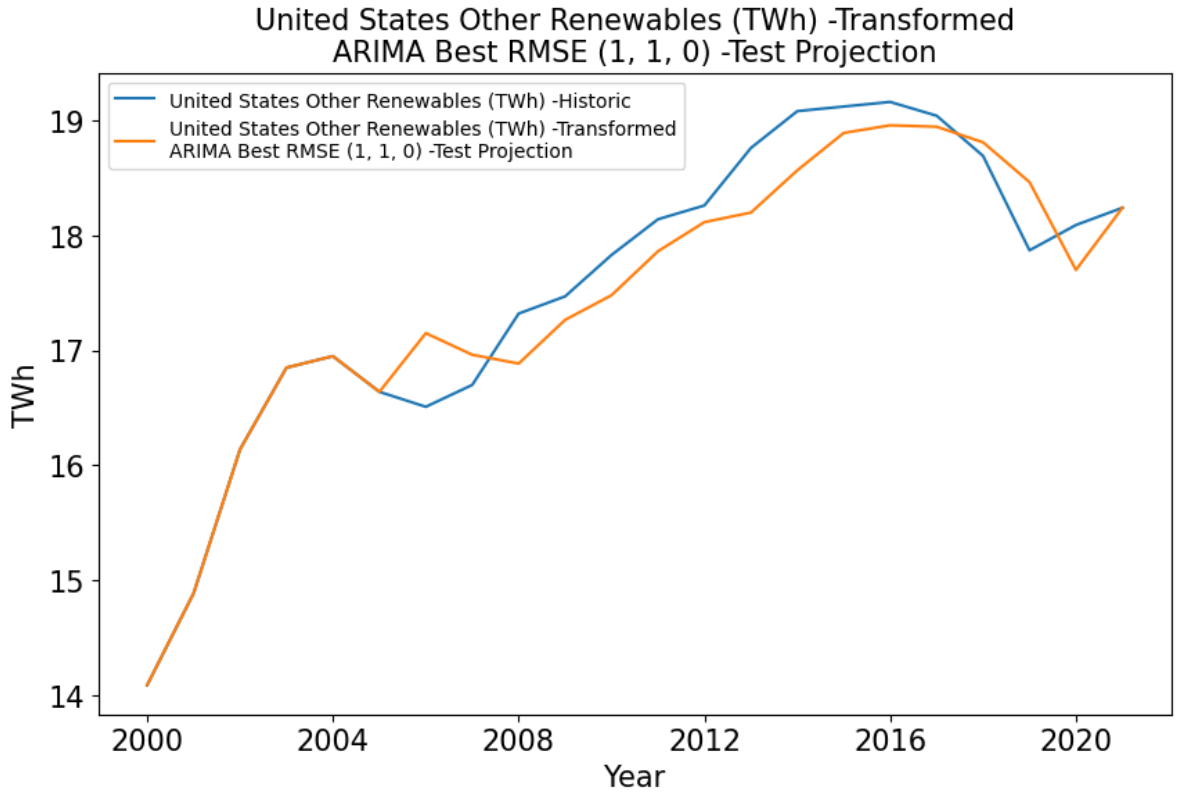
7.9.7 ARIMA Model and Grid Search

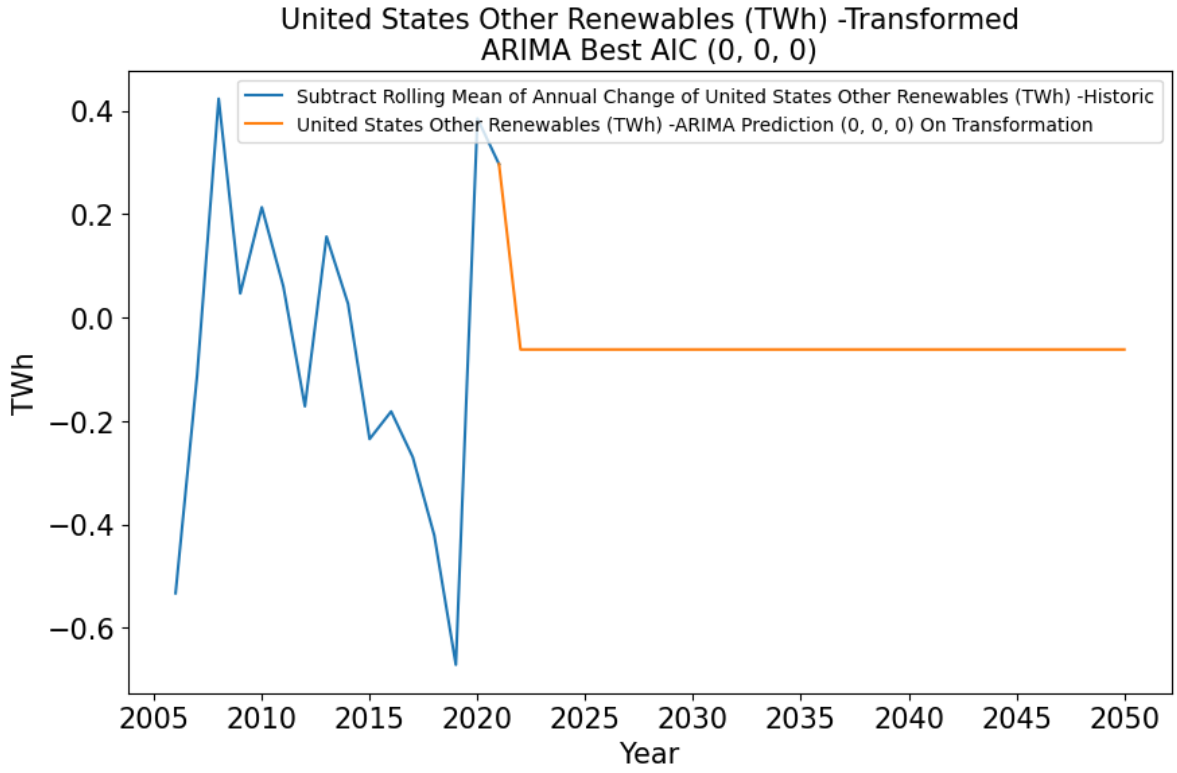
In [202]:
```
coal_rmse_cfg, coal_aic_cfg = arima_pdq(
                model_data_t.iloc[:,16:18], s=14,
                    tran = 'inv_diff_inv_sub', n=6)
```

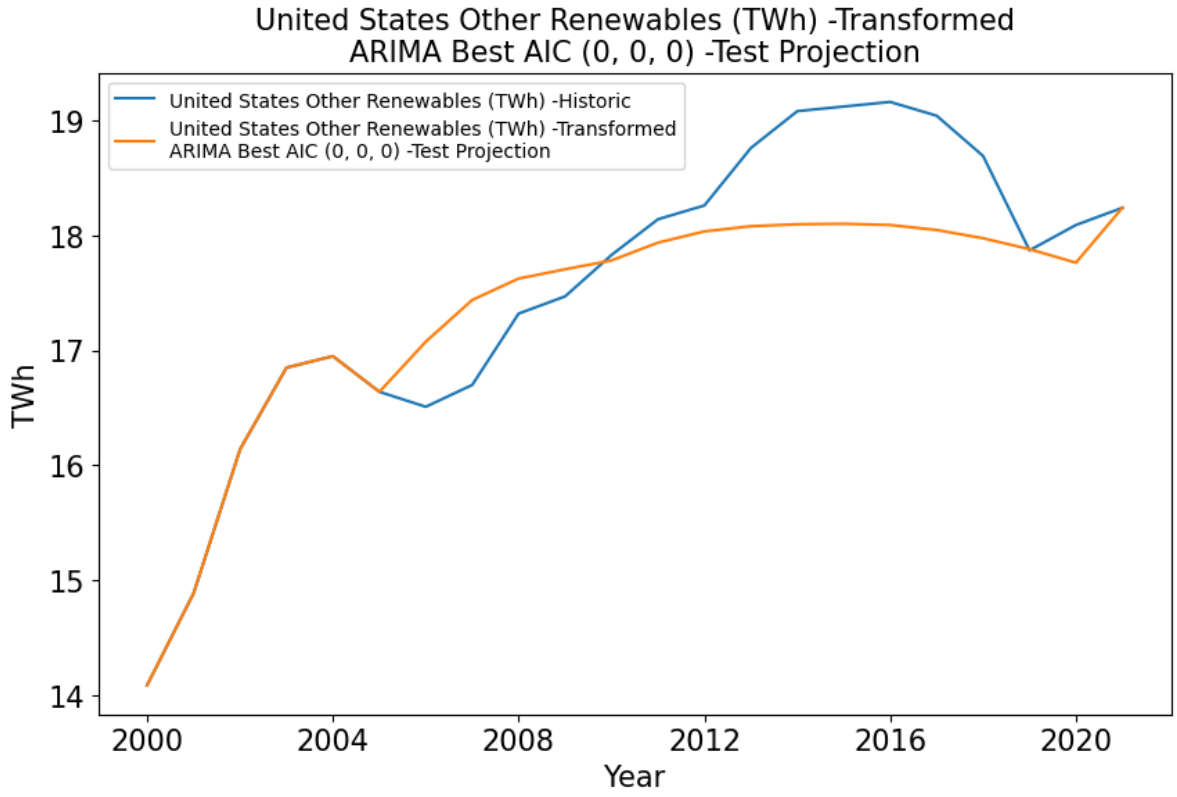
```
RMSE ARIMA: (0, 0, 0), RMSE= 0.65, AIC= 12.03, TWh-2050= 3.18
RMSE ARIMA: (0, 0, 1), RMSE= 0.55, AIC= 12.53, TWh-2050= 2.34
RMSE ARIMA: (0, 1, 0), RMSE= 0.40, AIC= 30.90, TWh-2050= 71.20
RMSE ARIMA: (1, 1, 0), RMSE= 0.39, AIC= 32.24, TWh-2050= 74.15
Best RMSE ARIMA: (1, 1, 0) RMSE= 0.39 AIC= 32.24, TWh-2050= 74.15
Best AIC  ARIMA: (0, 0, 0) RMSE= 0.65 AIC= 12.03, TWh-2050= 3.18
Graph_formatting produced error at (1, 1, 0) or (0, 0, 0)
```




United States Other Renewables (TWh) -Transformed
ARIMA Best RMSE (1, 1, 0)
Subtract Rolling Mean of Annual Change of United States Other Renewables (TWh) -Historic
United States Other Renewables (TWh) -ARIMA Prediction (1, 1, 0) On Transformation
TWh
0.4
0.2
0.0
−0.2
−0.4
−0.6
2005
2010
2015
2020
2025
2030
2035
2040
2045
2050
Year


United States Other Renewables (TWh) -Transformed
ARIMA Best RMSE (1, 1, 0) -Test Projection
United States Other Renewables (TWh) -Historic
United States Other Renewables (TWh) -Transformed
ARIMA Best RMSE (1, 1, 0) -Test Projection
TWh
19
18
17
16
15
14
2000
2004
2008
2012
2016
2020
Year





This graph may be more reasonable than it looks. I think it's strange that the RMSE values peaked before getting past a single digit increase for p, d, and q, but I double checked the numbers and it's right.

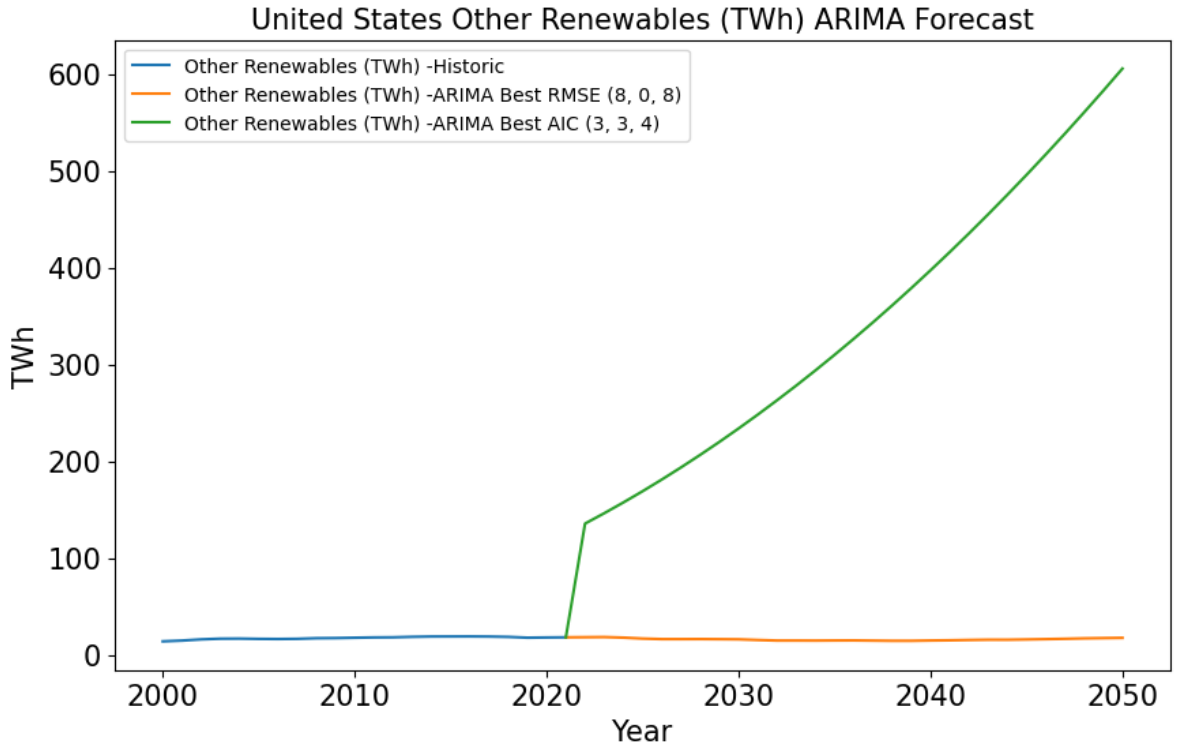
The other renewables category represent new sources of energy that are GHG emission free. I think we will see a lot of growth in this market in the next three decades. In fact, at peaking out at 70 TWhs, it's possible this may be conservative. This is where fusion will go if it ever gets off

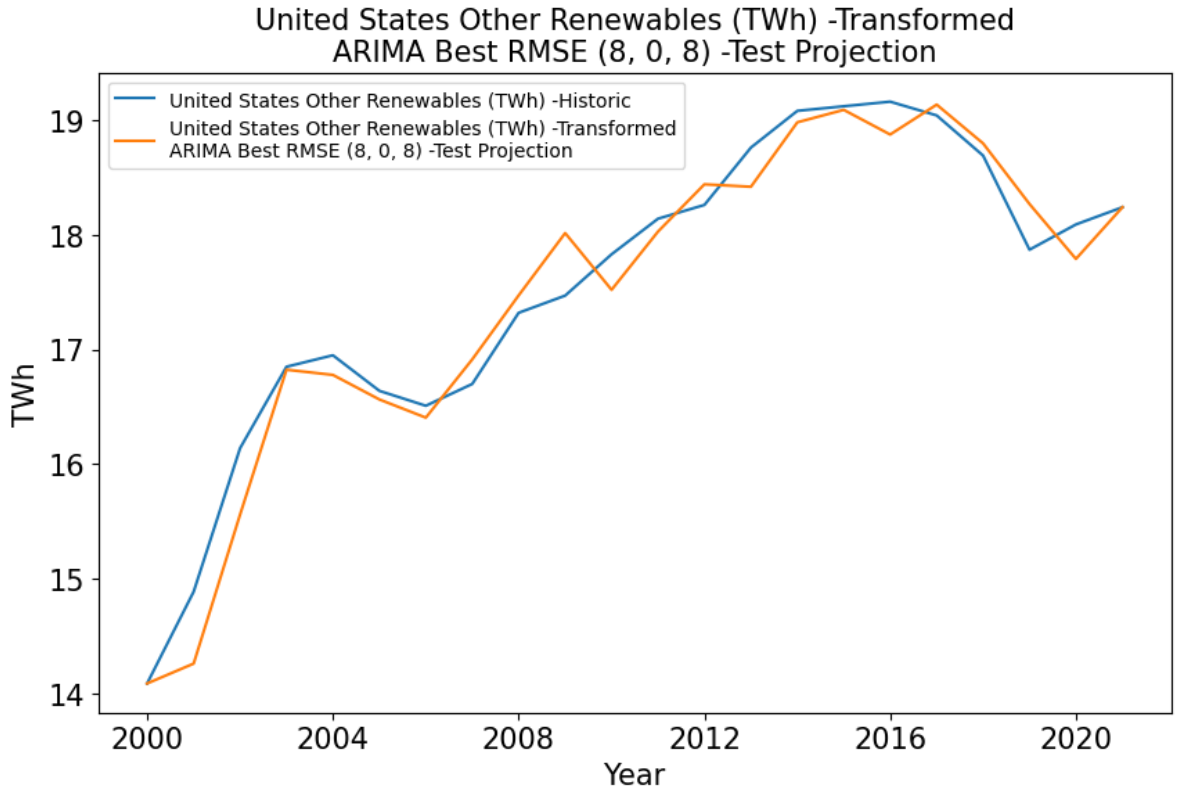
the ground, after all.

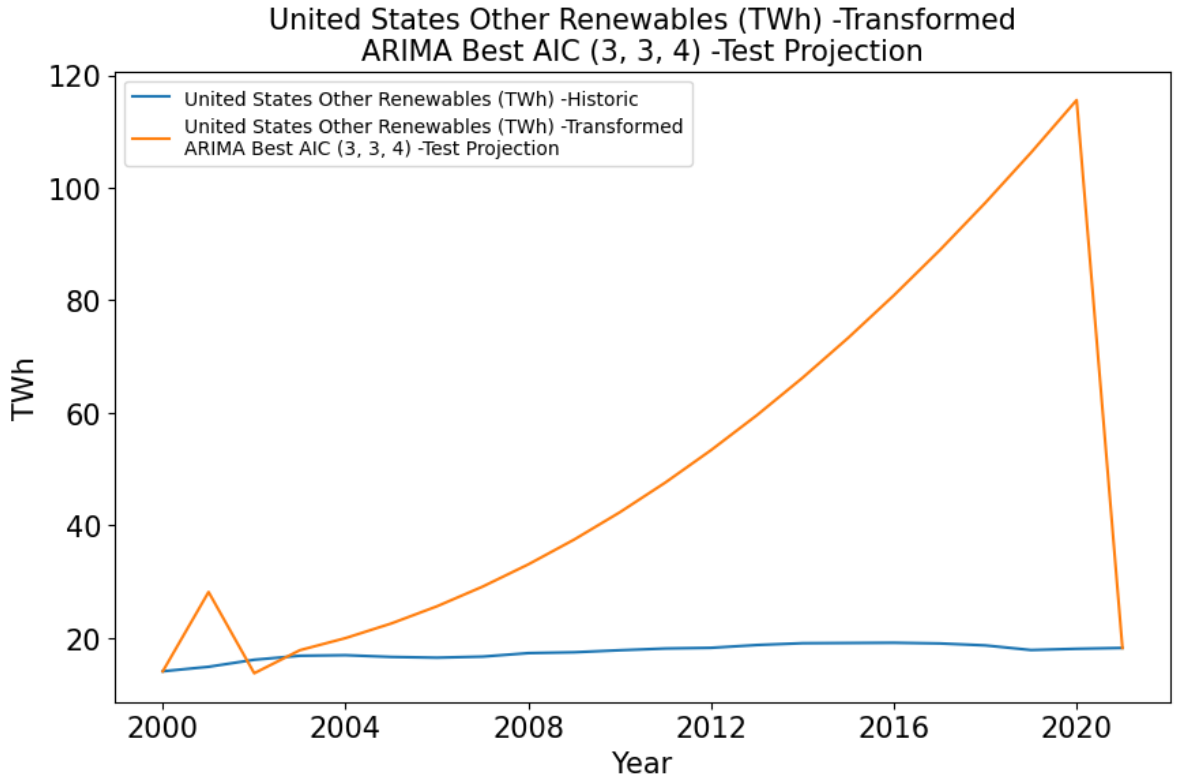
7.9.8 ARIMA Without Transformation

In [203]:
```
oth_rmse_cfg, oth_aic_cfg = arima_pdq_no_tran(model_data.iloc[:,8:9], 22, 50,
```

```
United States Other Renewables (TWh) Grid Search:
RMSE ARIMA: (0, 0, 0), RMSE= 1.09, AIC= 78.89, TWh-2050= 17.54
RMSE ARIMA: (0, 0, 1), RMSE= 0.56, AIC= 58.02, TWh-2050= 17.47
RMSE ARIMA: (0, 0, 2), RMSE= 0.37, AIC= 45.42, TWh-2050= 17.45
RMSE ARIMA: (0, 0, 3), RMSE= 0.34, AIC= 38.69, TWh-2050= 17.37
RMSE ARIMA: (0, 0, 4), RMSE= 0.31, AIC= 35.87, TWh-2050= 17.35
RMSE ARIMA: (0, 0, 5), RMSE= 0.31, AIC= 34.39, TWh-2050= 17.42
RMSE ARIMA: (0, 0, 8), RMSE= 0.31, AIC= 37.69, TWh-2050= 17.18
AIC  ARIMA: (0, 1, 0), RMSE= 0.46, AIC= 30.37, TWh-2050= 18.24
AIC  ARIMA: (0, 1, 1), RMSE= 0.38, AIC= 23.46, TWh-2050= 18.10
AIC  ARIMA: (0, 2, 0), RMSE= 1.39, AIC= 22.71, TWh-2050= 22.59
AIC  ARIMA: (1, 1, 0), RMSE= 0.38, AIC= 22.40, TWh-2050= 18.51
AIC  ARIMA: (3, 3, 4), RMSE= 44.13, AIC= 16.00, TWh-2050= 605.34
RMSE ARIMA: (7, 0, 7), RMSE= 0.29, AIC= 43.71, TWh-2050= 17.04
RMSE ARIMA: (8, 0, 8), RMSE= 0.28, AIC= 46.26, TWh-2050= 17.70
Best RMSE ARIMA: (8, 0, 8) RMSE= 0.28 AIC= 46.26, TWh-2050= 17.70
Best AIC  ARIMA: (3, 3, 4) RMSE= 44.13 AIC= 16.00, TWh-2050= 605.34
```




United States Other Renewables (TWh) -Transformed
ARIMA Best RMSE (8, 0, 8) -Test Projection
United States Other Renewables (TWh) -Historic
United States Other Renewables (TWh) -Transformed
ARIMA Best RMSE (8, 0, 8) -Test Projection
TWh
Year
14
15
16
17
18
19
2000
2004
2008
2012
2016
2020

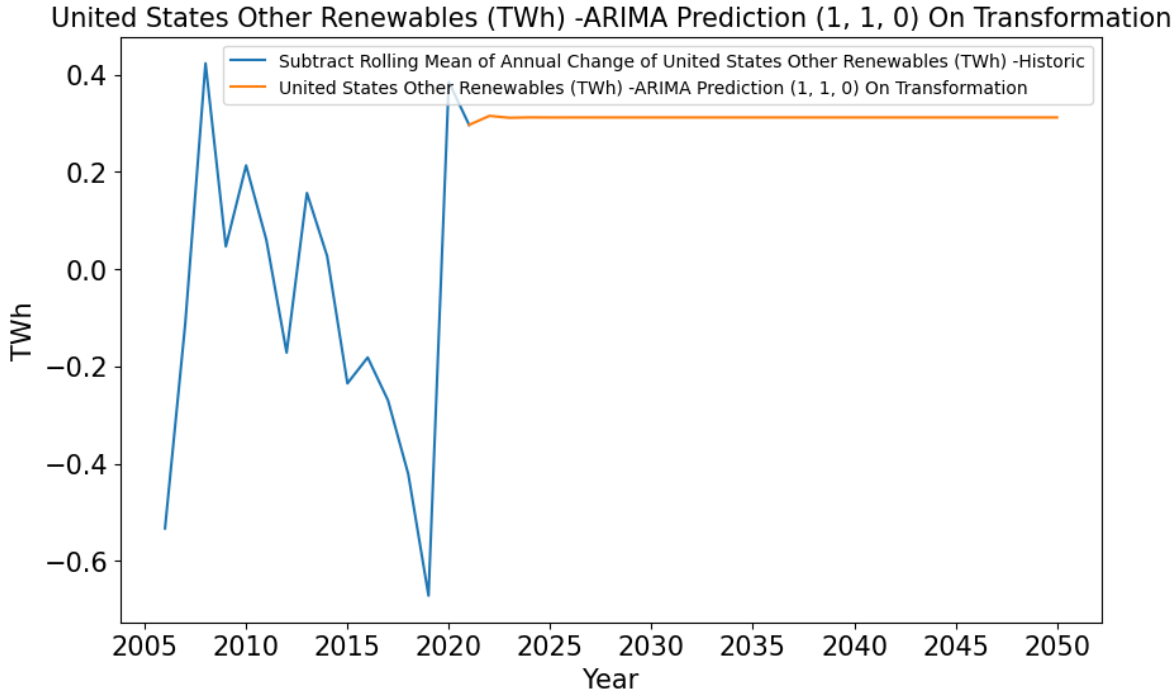

United States Other Renewables (TWh) -Transformed
ARIMA Best AIC (3, 3, 4) -Test Projection
United States Other Renewables (TWh) -Historic
United States Other Renewables (TWh) -Transformed
ARIMA Best AIC (3, 3, 4) -Test Projection
TWh
Year
20
40
60
80
100
120
2000
2004
2008
2012
2016
2020

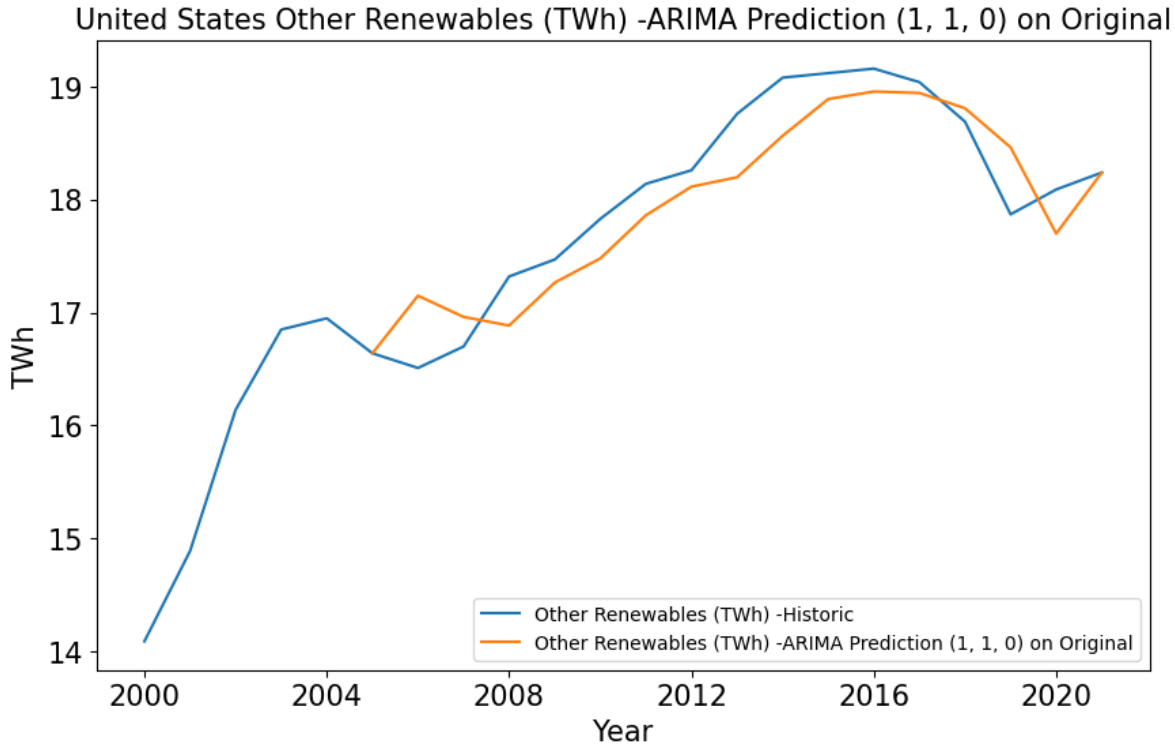
7.9.9 Model Selection

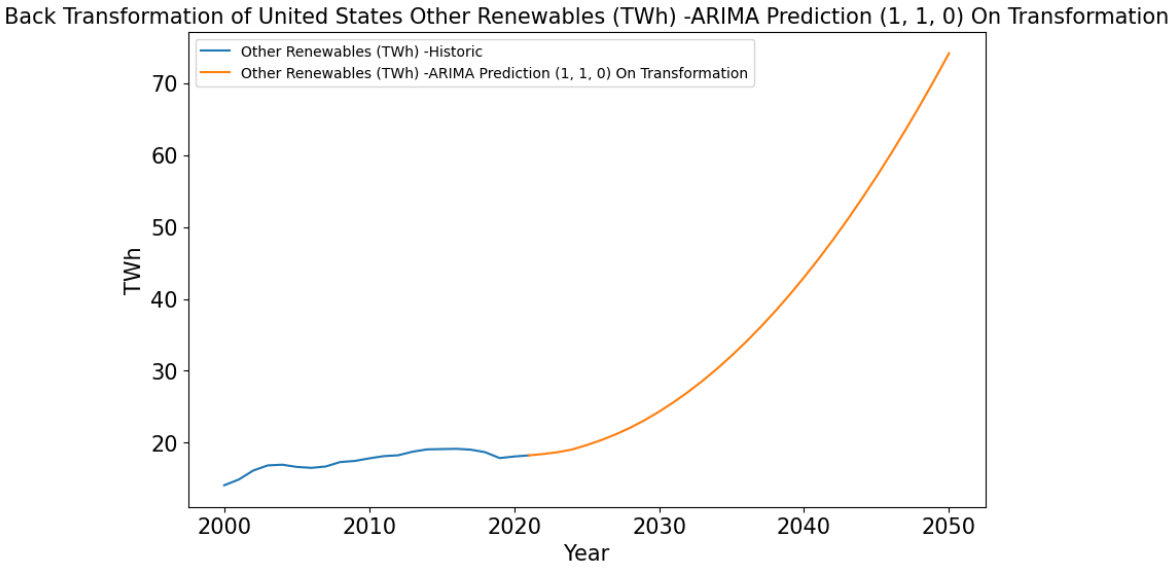
In [204]:
```
#Deleteme
stored_other = selected_models.copy()
```

In [205]:
```
selected_models = stored_other.copy() #deleteme
df_other = model_data_t.iloc[:,16:18]

#                         Model,       df_o,     pdq,
selected_models.append(['ARIMA_tran', df_other, (1,1,0),
                        'inv_diff_inv_sub', 6, None])

m = 8
#model_summary_no_tran(selected_models[m][1], selected_models[m][2], 22, 50);
# model_summary(df, o, st, en, s=0, tran=None, n=0):
model_summary(selected_models[m][1], selected_models[m][2], 22, 50, 14,
              selected_models[m][3], selected_models[m][4])
```

```
ARIMA: (1, 1, 0), RMSE=0.39, AIC=32.24
**********************************************************************
**
United States Other Renewables (TWh) model results.

                                                                SARIMAX Results
==============================================================================
==============================================================
Dep. Variable:     Subtract Rolling Mean of Annual Change of United States Ot
her Renewables (TWh)   No. Observations:                   22
Model:
ARIMA(1, 1, 0)   Log Likelihood                -14.118
Date:
Sat, 29 Apr 2023   AIC                           32.236
Time:
06:26:54   BIC                             34.325
Sample:
01-01-2000   HQIC                            32.690

- 01-01-2021
Covariance Type:
opg
==============================================================================
=
                 coef    std err          z      P>|z|      [0.025      0.97
5]
------------------------------------------------------------------------------
-
ar.L1         -0.2107      0.339     -0.621      0.535      -0.876       0.45
5
sigma2         0.1351      0.040      3.357      0.001       0.056       0.21
4
==============================================================================
======
Ljung-Box (L1) (Q):                   0.08   Jarque-Bera (JB):
14.15
Prob(Q):                              0.78   Prob(JB):
0.00
Heteroskedasticity (H):               7.20   Skew:
1.60
Prob(H) (two-sided):                  0.02   Kurtosis:
5.44
==============================================================================
======

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (comple
x-step).
```


United States Other Renewables (TWh) -ARIMA Prediction (1, 1, 0) On Transformation
Subtract Rolling Mean of Annual Change of United States Other Renewables (TWh) -Historic
United States Other Renewables (TWh) -ARIMA Prediction (1, 1, 0) On Transformation
TWh
Year


United States Other Renewables (TWh) -ARIMA Prediction (1, 1, 0) on Original
Other Renewables (TWh) -Historic
Other Renewables (TWh) -ARIMA Prediction (1, 1, 0) on Original
TWh
Year



As I wrote above, an exponential curve might be fight for other renewable sources of energy. This is research and there is a lot of money going into this market. It might even be conservative. This curve has the best RMSE.

8 Graphing Energy Production Sources

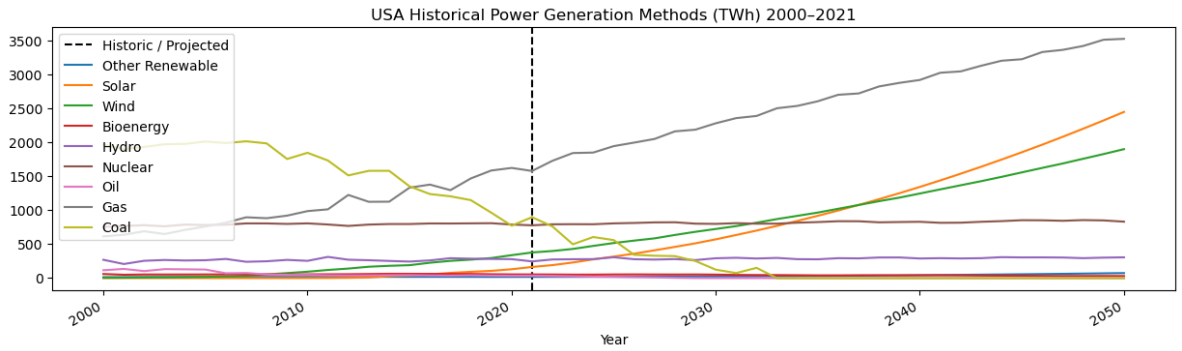
In [206]:
```
# This creates models as specified in the selected_models list, and returns
# the predicted values ready to graph.
def y_hats_sel_mod(sel_mod, st, en):
    y_hats = pd.DataFrame()
    for sm in sel_mod:
        if sm[0] == 'ARIMA':
            m, y_hat, yo = arima_no_tran(
                                sm[1], sm[2], st, en)

        elif sm[0] == 'ARIMA_tran':
            yt, df, y_hat = mod_pred_s(sm[1], sm[2], st, en, tran=sm[3],
                                       n=sm[4])

            # Setting the first point of y-hat to the last point of the
            # DataFrame to show continuious change on the graph.
            y_hat.loc[sm[1].index[-1], y_hat.columns] = sm[1].iloc[-1,0].copy(
            # Sorting index
            y_hat.sort_index(inplace=True)

        # Adding the y_hat to the DataFrame with all y_hats
        y_hats = pd.concat([y_hats,y_hat], axis=1)

    # Setting all negative y-hat values to zero.
    y_hats[y_hats < 0] = 0

    return y_hats
```

In [207]:
```
# This is the same as the stacked energy graph from earlier in this notebook,
# but it adds the ability to plot the projected data next to the historic.
def stacked_energy_proj(his_en, his_dem, pj_en, pj_dem, title, s=0):

    # Creating the labels for the legend.
    df_names = list(pj_en.columns.copy())
    if s>0:
        for i in range(len(df_names)):
            # In case the beginning of the column is "Back Transformation of "
            # but the graph doesn't need that in the index label.
            if df_names[i][:23] == 'Back Transformation of ':
                ss = s+23
            else:
                ss = s+0
            df_names[i] = str(df_names[i])[ss:]

    his_dem.name = his_dem.iloc[:,0:1].columns[0][s:]
    pj_dem.name = pj_dem.iloc[:,0:1].columns[0][s:]

    fig = plt.figure(figsize = (20,12))
    plt.title(title, size = 15)

    # Plotting the demand, both historic and projected
    plt.plot(his_dem.index, his_dem.iloc[:,0], '--',
             label = his_dem.name)
    plt.plot(pj_dem.index, pj_dem.iloc[:,0], '--', color = 'red',
             label = pj_dem.name)

    # Plotting the historical energy production
    his_en_lst = []
    for i in range(len(his_en.columns)):
        his_en_lst.append(his_en.iloc[:,i])

    # Plotting the projected energy production.
    pj_en_lst = []
    for i in range(len(pj_en.columns)):
        pj_en_lst.append(pj_en.iloc[:,i])

    # Printing the stacked energy production
    # Resetting colors to ensure color parity across predicted and historic.
    plt.gca().set_prop_cycle(None)
    plt.stackplot(his_en.index, his_en_lst)
    # Resetting colors to ensure color parity across predicted and historic.
    plt.gca().set_prop_cycle(None)
    plt.stackplot(pj_en.index, pj_en_lst, labels = df_names)

    # Line at x-axis showing historic vs projected
    plt.axvline(his_en.index[-1], color = 'black', linestyle = 'dashed',
                label = 'Historic / Projected')

    # Formatting
    plt.xlabel('Year', size = 15)
    plt.ylabel('TWh', size = 15)
    plt.xticks(size=15)
    plt.yticks(size=15)
    plt.legend(loc=2)
    # show the graph
```

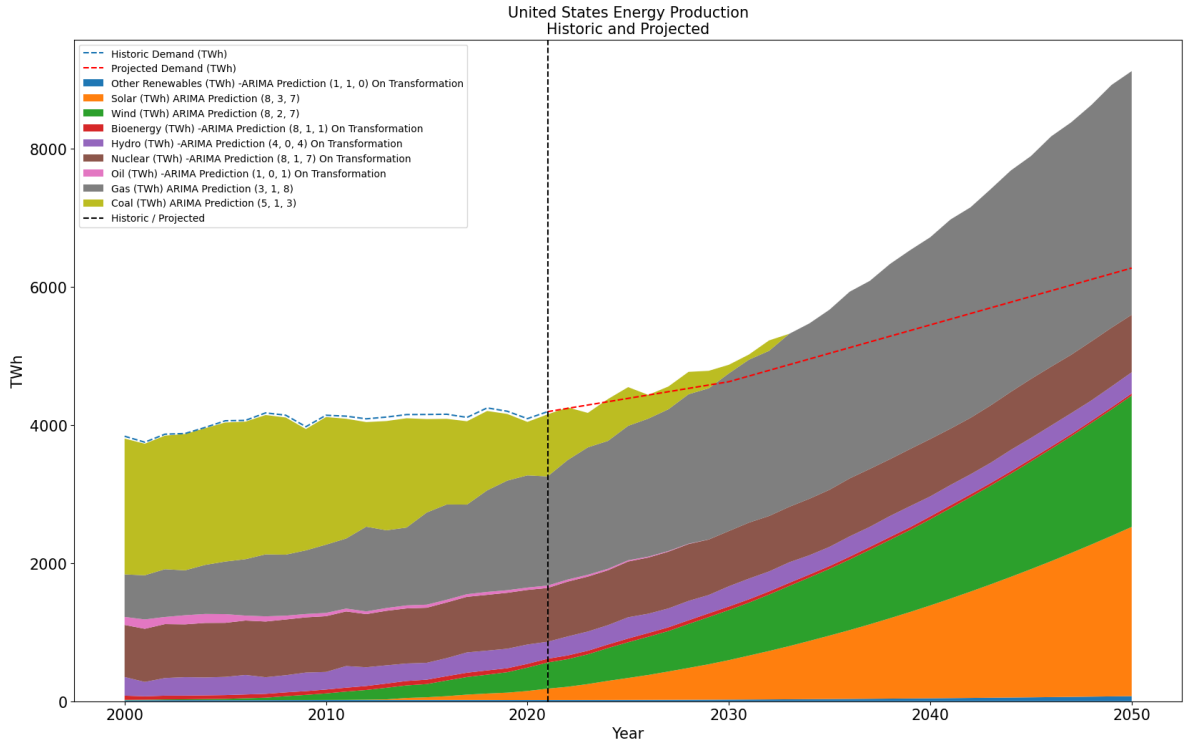
```
    plt.show()
```

In [208]:
```
y_hats = y_hats_sel_mod(selected_models, 22, 50)
```

In [209]:
```
# Line at x-axis showing historic vs projected
plt.axvline(df_hist_dem.index[-1], color = 'black', linestyle = 'dashed',
                label = 'Historic / Projected')

line8 =  model_data.iloc[:,8].plot(label = 'Other Renewable')
line7 =  model_data.iloc[:,7].plot(label = 'Solar')
line6 =  model_data.iloc[:,6].plot(label = 'Wind')
line5 =  model_data.iloc[:,5].plot(label = 'Bioenergy')
line4 =  model_data.iloc[:,4].plot(label = 'Hydro')
line3 =  model_data.iloc[:,3].plot(label = 'Nuclear')
line2 =  model_data.iloc[:,2].plot(label = 'Oil')
line1 =  model_data.iloc[:,1].plot(label = 'Gas')
line0 =  model_data.iloc[:,0].plot(figsize=(15,4),label = 'Coal')

plt.legend(loc = 6)

# Resetting colors to ensure color parity across predicted and historic.
plt.gca().set_prop_cycle(None)

line8_pred = y_hats.iloc[:,8].plot()
line7_pred = y_hats.iloc[:,7].plot()
line6_pred = y_hats.iloc[:,6].plot()
line5_pred = y_hats.iloc[:,5].plot()
line4_pred = y_hats.iloc[:,4].plot()
line3_pred = y_hats.iloc[:,3].plot()
line2_pred = y_hats.iloc[:,2].plot()
line1_pred = y_hats.iloc[:,1].plot()
line0_pred = y_hats.iloc[:,0].plot()


plt.title('USA Historical Power Generation Methods (TWh) 2000-2021');
```



In [210]:
```
# Energy use projected into the future.
stacked_energy_proj(model_data.iloc[:,::-1], df_hist_dem.iloc[:,8:9],
                    y_hats.iloc[:,::-1], df_proj_dem_stat.iloc[:,8:9],
                'United States Energy Production\nHistoric and Projected', 14)
```



The graph shows the projected energy production until the year 2050. To meet the Paris Agreement, we have to diminish coal, oil and natural gas. As the graph shows, coal is already on a trend out, as is oil. Natural gas will be difficult to trim as its rise will make up nearly half of all energy production by 2050. That's my task in the next section.

9 Reduction in Natural Gas Energy Production to Meet Paris Agreement Targets

In [211]:
```
# Create a new DataFrame for the reduction of natural gas, and increase of
# renewables.
pd.set_option('display.max_rows',6)

rdc_y_hats = y_hats.copy()
rdc_y_hats.columns = model_data.columns
df_pts = rdc_y_hats.copy()
rdc_y_hats
```

Out[211]:

	United States Coal (TWh)	United States Gas (TWh)	United States Oil (TWh)	United States Nuclear (TWh)	United States Hydro (TWh)	United States Bioenergy (TWh)	United States Wind (TWh)	United States Solar (TWh)	United States Other Renewables (TWh)
2021-01-01	897.89	1,579.36	35.20	778.19	246.47	54.25	378.20	164.42	18.24
2022-01-01	761.47	1,729.42	30.41	793.20	275.07	53.52	399.61	191.81	18.43
2023-01-01	498.83	1,843.62	26.62	794.83	278.37	51.62	430.35	230.51	18.69
...	...	...	...	...	...	...	...	...	...
2048-01-01	0.00	3,421.31	0.00	855.37	294.31	31.40	1,759.11	2,201.84	66.92
2049-01-01	0.00	3,513.38	0.00	850.49	301.92	31.60	1,828.65	2,324.00	70.48
2050-01-01	0.00	3,526.45	0.00	832.07	306.50	30.84	1,900.81	2,449.58	74.15

30 rows × 9 columns

Nuclear plants take a long, long time to build. For example, Wyoming has a Nuclear power plant under construction now that is scheduled to open in 2030. To meet the 2030 target, we need to deploy renewable energy production at a much faster rate. The only way to do this is to expand wind and solar technology.

At current construction rates, it will be difficult to replace the reduction we need in natural gas with wind and solar. Instead, I'll say we continue natural gas production for three years, then start the rapid reduction.

In [212]:
```
# Getting the natural gas column ready for interpolation, starting with three
# years into the projected values.
rdc_y_hats.iloc[4:,1] = np.nan

rdc_y_hats.rename(columns = {rdc_y_hats.columns[1] :
                             rdc_y_hats.columns[1] + ' -Targets'},
                              inplace=True)
rdc_y_hats.iloc[:,1:2]
```

Out[212]:

	United States Gas (TWh) -Targets
2021-01-01	1,579.36
2022-01-01	1,729.42
2023-01-01	1,843.62
...	...
2048-01-01	NaN
2049-01-01	NaN
2050-01-01	NaN

30 rows × 1 columns

In [213]:
```
# Getting the 2030 reduction target number for gas. We need to be at 45%
# of 2010's production level.
display(model_data.iloc[10:11,1:2])
gas_2030_target = model_data.iloc[10,1]*.45
print('The IPCC target for gas is 45% of the 2010 production levels.',
      '\nForty five percent of {:.2f} TWh is {:.2f} TWh.'.format(
          model_data.iloc[10,1], gas_2030_target))
```

Year	United States Gas (TWh)
2010-01-01	987.70

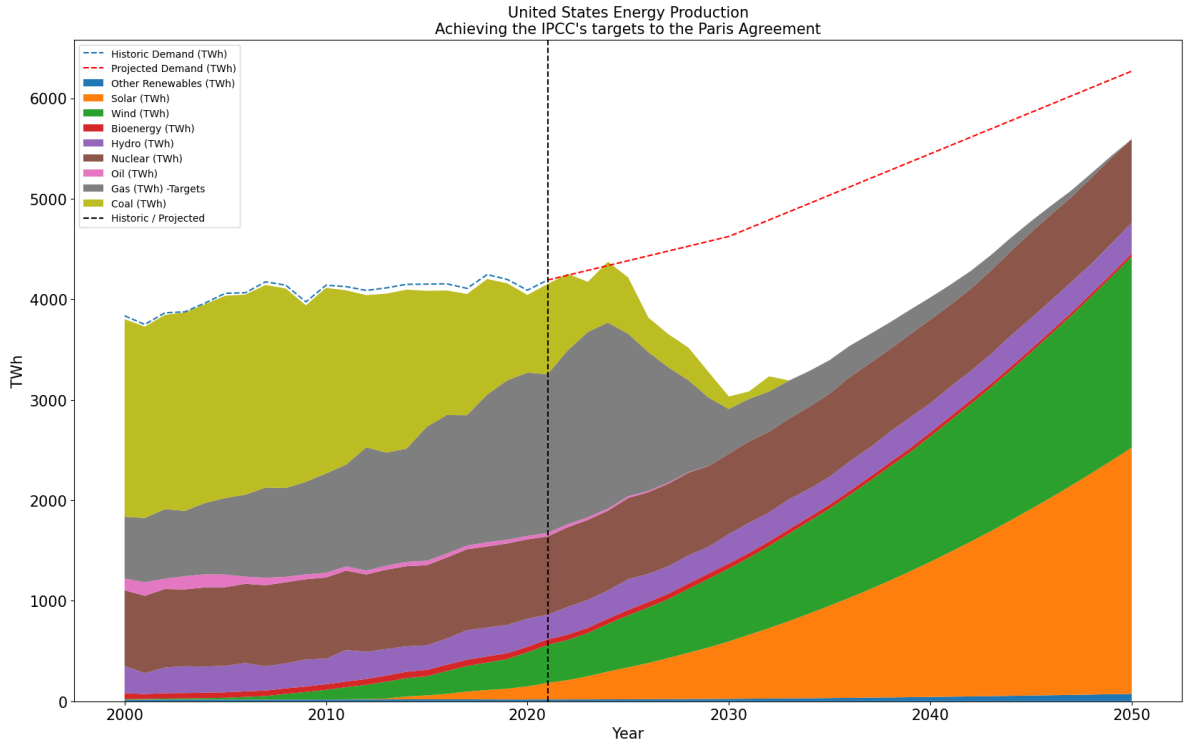
```
The IPCC target for gas is 45% of the 2010 production levels.
Forty five percent of 987.70 TWh is 444.47 TWh.
```

In [214]:
```
pd.set_option('display.max_rows', None)
# Setting gas targets for 2030 and 2050
# 2030 is row 9.
rdc_y_hats.iloc[9,1] = gas_2030_target
rdc_y_hats.iloc[-1,1] = 0

# Interpolate the remaining values.
rdc_y_hats.interpolate('linear', inplace=True)
display(rdc_y_hats.iloc[:,1:2])
```

	United States Gas (TWh) -Targets
2021-01-01	1,579.36
2022-01-01	1,729.42
2023-01-01	1,843.62
2024-01-01	1,850.38
2025-01-01	1,616.06
2026-01-01	1,381.74
2027-01-01	1,147.42
2028-01-01	913.10
2029-01-01	678.78
2030-01-01	444.47
2031-01-01	422.24
2032-01-01	400.02
2033-01-01	377.80
2034-01-01	355.57
2035-01-01	333.35
2036-01-01	311.13
2037-01-01	288.90
2038-01-01	266.68
2039-01-01	244.46
2040-01-01	222.23
2041-01-01	200.01
2042-01-01	177.79
2043-01-01	155.56
2044-01-01	133.34
2045-01-01	111.12
2046-01-01	88.89
2047-01-01	66.67
2048-01-01	44.45
2049-01-01	22.22
2050-01-01	0.00

Now I'll reprint the slide with the natural gas reduction.

In [215]:
```
# Printing the future energy projections again.
stacked_energy_proj(model_data.iloc[:,::-1], df_hist_dem.iloc[:,8:9],
                    rdc_y_hats.iloc[:,::-1], df_proj_dem_stat.iloc[:,8:9],
                ('United States Energy Production\n' +
                 "Achieving the IPCC's targets to the Paris Agreement"), 14)
```



That's a lot of energy that the United States needs to make up for. The first thing I need to do is calculate the 2030 deficit.

In [216]:
```
# This cell calculates the necessary numbers and explains what they mean.

energy_produced_2030 = np.sum(rdc_y_hats.iloc[9,:])
energy_demand_2030 = df_proj_dem_stat.iloc[1,8]
energy_deficit_2030 = energy_demand_2030 - energy_produced_2030

print('IEA Projected Energy Demand USA 2030: ',
      '{:.2f} KWh'.format(energy_demand_2030))
print('Production with natural gas reduction:',
      '{:.2f} KWh'.format(energy_produced_2030))
print('Deficit of energy production:         ',
      '{:.2f} KWh\n'.format(energy_deficit_2030))

print('In the year 2030, the IEA projects that United States will ' +
      'need {:.2f} KWh of'.format(energy_demand_2030),
      '\nenergy. If, by then, the United States has reduced Natural Gas',
      'emissions to',
      '\nIPCC recommended levels, then the United States will have an energy',
      'deficit\nof {:.2f} KWh.'.format(energy_deficit_2030),
      'The best chance the United States has to making up this deficit',
      '\nis to construct more wind and solar power.')
```

```
IEA Projected Energy Demand USA 2030:  4624.97 KWh
Production with natural gas reduction: 3032.61 KWh
Deficit of energy production:          1592.36 KWh

In the year 2030, the IEA projects that United States will need 4624.97 KWh o
f
energy. If, by then, the United States has reduced Natural Gas emissions to
IPCC recommended levels, then the United States will have an energy deficit
of 1592.36 KWh. The best chance the United States has to making up this defic
it
is to construct more wind and solar power.
```

In [217]:
```
# Calculating the annual increase for wind and solar power.
diffs = rdc_y_hats.iloc[:,6:8].diff()
# I will set the rate of change of the first year as the initial production
# rate.
prod_rate = diffs.iloc[1]

# Double capacity of wind and solar production annually for three years.
for i in range(1,4):
    prod_rate = prod_rate*2
    rdc_y_hats.iloc[i:i+2,6:8] = rdc_y_hats.iloc[i-1,6:8] + prod_rate

# Keep that production rate steady perpetually.
for i in range(4,len(rdc_y_hats)):
    rdc_y_hats.iloc[i:i+1,6:8] = rdc_y_hats.iloc[i-1:i,6:8] + prod_rate

print('Proposed annual production rate of wind and solar:')
display(prod_rate)

rdc_y_hats.iloc[:,6:8]
```

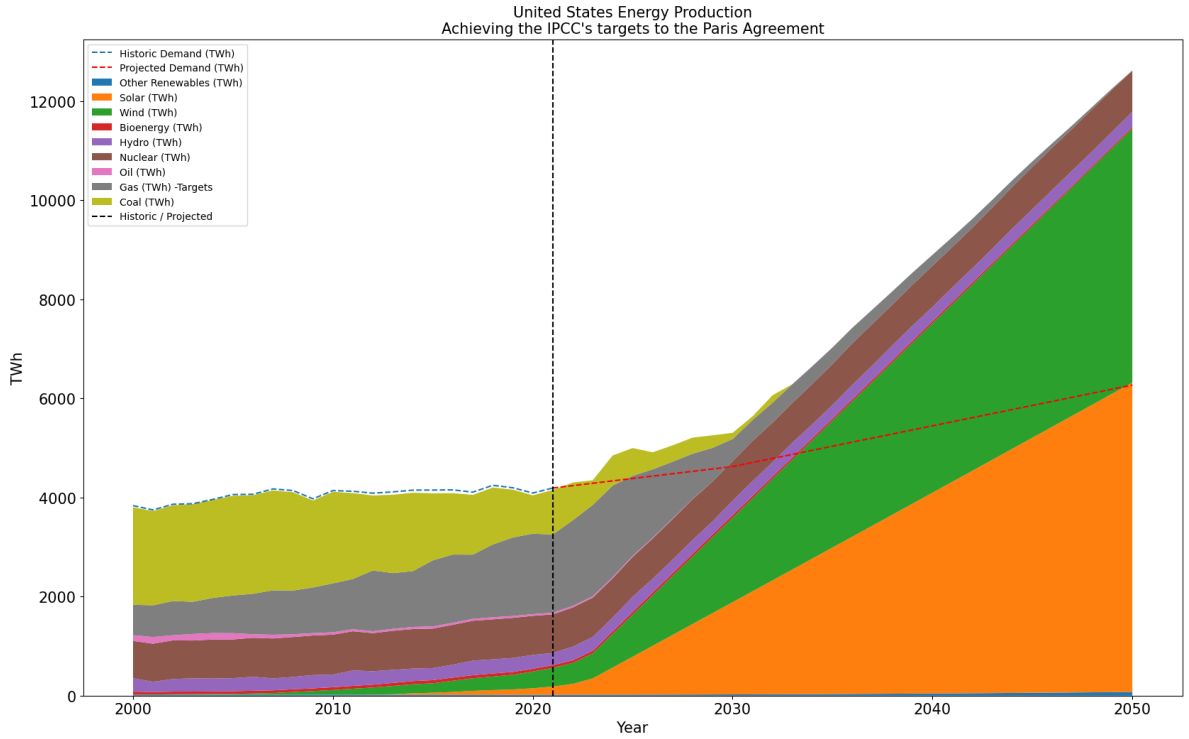
```
Proposed annual production rate of wind and solar:

United States Wind (TWh)    171.27
United States Solar (TWh)   219.15
Name: 2022-01-01 00:00:00, dtype: float64
```

Out[217]:

	United States Wind (TWh)	United States Solar (TWh)
2021-01-01	378.20	164.42
2022-01-01	421.02	219.21
2023-01-01	506.66	328.78
2024-01-01	677.93	547.92
2025-01-01	849.20	767.07
2026-01-01	1,020.48	986.21
2027-01-01	1,191.75	1,205.36
2028-01-01	1,363.03	1,424.50
2029-01-01	1,534.30	1,643.65
2030-01-01	1,705.58	1,862.80
2031-01-01	1,876.85	2,081.94
2032-01-01	2,048.12	2,301.09
2033-01-01	2,219.40	2,520.23
2034-01-01	2,390.67	2,739.38
2035-01-01	2,561.95	2,958.52
2036-01-01	2,733.22	3,177.67
2037-01-01	2,904.50	3,396.81
2038-01-01	3,075.77	3,615.96
2039-01-01	3,247.05	3,835.10
2040-01-01	3,418.32	4,054.25
2041-01-01	3,589.59	4,273.39
2042-01-01	3,760.87	4,492.54
2043-01-01	3,932.14	4,711.68
2044-01-01	4,103.42	4,930.83
2045-01-01	4,274.69	5,149.97
2046-01-01	4,445.97	5,369.12
2047-01-01	4,617.24	5,588.26
2048-01-01	4,788.51	5,807.41
2049-01-01	4,959.79	6,026.55
2050-01-01	5,131.06	6,245.70

Now, I'll plot how these changes will affect the future.

In [218]:
```
# Energy use projected into the future.
stacked_energy_proj(model_data.iloc[:,::-1], df_hist_dem.iloc[:,8:9],
                    rdc_y_hats.iloc[:,::-1], df_proj_dem_stat.iloc[:,8:9],
                ('United States Energy Production\n' +
                 "Achieving the IPCC's targets to the Paris Agreement"), 14)
```



That achieves the objectives set out by the IPCC to meet Paris Agreement commitments. Natural gas use is down in to the required levels. Oil, and nearly coal, both are diminished entirely by 2030, and renewable production is powering the United States. The good news is the massive surplus the United States has made for itself can be exported to other nations in need of clean power themselves. This is, after all, a global problem.

10 Conclusion

10.1 Limitations of Scope / Provided Data

This analysis is limited to just the United States, and also doesn't clarify how the reduction in natural gas production will affect the earth's temperature at all. I have data available that will allow me to:

1. **Add All Nations:** Project the energy generation of other nations of the world, and perform a similar analysis for how they can reduce their reliance on fossil fuels.
2. **Estimate Proposed GHG Emissions Reductions:** Convert the power generation sources into estimates on GHG emissions. I have equations available that can accomplish this.
3. **Combine This Analysis With the SSP Scenarios:** Compare how reductions across the world will reduce projected GHG emissions through the different Shared Socioeconomic Pathway (SSP) scenarios proposed by the IPCC. The SSPs are used to measure what

global temperatures will be like under different circumstances. I have emissions data available for each. I can subtract my emission reductions from each to see how that will effect the overall picture of each SSP.

4. **Correlate Emissions with Atmospheric Concentration:** Create time series analysis that correlate CO2 and CH4 emissions into the atmospheric concentration of both substances. The emissions data will be from the SSP scenarios as specified above. If I can correlate that with atmospheric concentration, then I can...
5. **Determine Radiative Forcings:** Convert the CO2 and CH4 atmospheric concentrations into radiative forcing with a calculation provided by NOAA and the IPCC. Radiative forcings are the common unit that allows scientists to analyze different chemical's heat absorption from the sun, which has lead to this global crisis.
6. **Global Annual Average Temperature** Convert the radiative forcing into estimated average annual temperature change.

In short, these steps will allow me to measure how the change in electricity production fuel sources will correlate with the change in average annual temperature for the world. This is my dream scope for this project. I've gathered all the data already. The only missing factor is time

10.2 Future Research Opportunities

- **Enact the Dream Scope:** I would like to analyze all the countries and regions of the world in the same way I did the United States. I would like to connect the dots to draw a direct correlation between power generation sources and the average annual global temperature.
- **Add 2022's Data:** I would also like to add 2022's electricity generation data so I don't have to begin projecting after 2021.
- **Better Emissions Data:** Though I didn't use it for this analysis, as part of the dream scope, I found emissions data on all countries, but it had one frustrating omission. I couldn't find emissions data exclusively for the power generation sector. The data I could find on this issue combined emissions from power generation with emissions from all fuel combustion, including automobiles. I would like to further research this so I could better learn the scope of emissions for the power generation sector.
- **Primary Power:** This analysis focused exclusively on Secondary Power, which is making the fuel at the power plant. It doesn't acknowledge emissions from extracting natural gas, oil and coal from the earth. We know that a lot of carbon emissions occurs before the primary fuel source enters the power plant ready to be burned. It would be interesting to see how much reducing reliance on these fuel sources decreases emissions at their extraction.
- **Transmission Losses:** Energy is loss between power plants and the businesses and homes they service just through the transmission across the power lines. It would be interesting to add these considerations to the study.
- **Atmospheric Concentration of Methane (CH4):** Before I limited my scope to America and power generation reduction to meet IPCC targets, I experimented with building a time series model to correlate GHG emissions with atmospheric concentration. I successfully modeled the atmospheric concentrations with CO2 emissions, but I could not correlate Methane emissions with atmospheric concentration. This will take more research. (To view this work, look at the Multivariate Jupyter Notebook found in the same directory as this notebook.)

10.3 Recommendations

This analysis yields to three recommendations:

1. **Reduce Natural Gas:** We are fortunate that market forces are driving coal and oil power production out of existence. The same is not true for natural gas. It is already the primary source of electricity generation in America. We must actively prepare for its replacement as soon as possible.
2. **Double Production of Wind and Solar Power:** This is critical. To meet the future challenges of reducing our GHG emissions in the power generation sector, we must replace it with renewable energy that does not emit GHG at all. At this time, the United States is not producing enough new wind turbines and solar panels to account for the reductions we need. We must double capacity, the double capacity, then double capacity a third time for the next three years. Then we will have the production capacity to eliminate natural gas power generation.
3. **Export Wind and Solar:** If we ramp up our solar and wind capabilities to meet the challenge we face, there will be no reason to cease operating at this capacity. Sell the excess solar panels and wind turbines to other nations. This will be good business for the United States, and it will help meet the challenges we face with the climate crisis. Acting alone will not be enough to preserve the world we live in today. We must do everything we can to encourage other nations to switch to renewable energy sources as soon as possible.

Here are three more recommendations that are not detailed by this analysis, but are related to the problem:

1. **Invest in Nuclear ASAP:** I did not consider new nuclear power plants as a replacement for natural gas for two reasons: 1) Nuclear is the most expensive fuel source at this time and 2) Nuclear plants take about a decade to build.

   However, there are multiple companies rapidly trying to change this, or as rapidly as they can. TerraPower (https://www.terrapower.com/), owned by Bill Gates, is currently building a power plant in Wyoming, but it doesn't open until 2030. Even then, the amount of electricity it will provide is not enough to significantly replace natural gas. It is a test project to see if Nuclear Power can be done safer and cheaper. If it works, it theoretically could be mass produced across the country, and the world, but... now we're talking about 2040 at the earliest. We need solutions between now and then. Those solutions are the use of wind and solar. It's the best we've got at this time.
2. **Invest in Educating the Public:** Years ago, the United States forced cigarette manufacturers to spend as much money on raising awareness of the ill effects of smoking as they did on advertising. It was very effective. Americans do not smoke as much as many other countries. Why not do the same thing with the dangers of fossil fuels? Part of the reason the climate crisis has been so difficult to manage politically is the public is barely aware of the dangers or the exact causes. Help the public understand the effects of emissions, including the eight million people who die every year of pollution born diseases.
3. **Buy Natural Gas Power Companies:** American capitalists will fight he reduction in natural gas as hard as they possibly can because it is their paycheck. Instead of the government playing the role of regulator and legislating natural gas out of existence, take a page from

the time-honored tradition of buying them out. Answer the cries of communism with a promise to sell the companies back to some willing capitalist a specified number of years after their emissions have dropped to zero.

That's a wrap! Thanks for hiring Greg Osborne and I hope to work with you more in the future.