Comparing Ampleforth and Tellor's Oracles

AMPL/USD Feed

Using web3 to get Tellor's data, ABI via etherscan. Looking to see the update times and price accuracy of Tellor and Ampleforth's AMPL/USD feeds

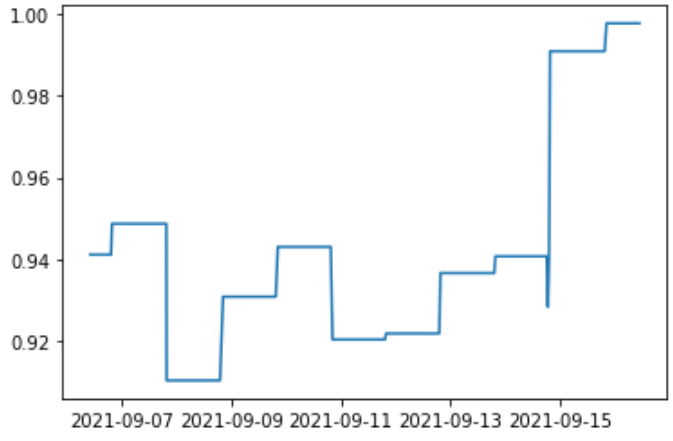
In [1]:
```
from web3 import Web3
infura_link = 'https://mainnet.infura.io/v3/3e5c6b2a48494d9a921a52ec1cc0a8ff'
w3 = Web3(Web3.HTTPProvider(infura_link))
```

In [2]:
```
tellor_add = "0xb2b6c6232d38fae21656703cac5a74e5314741d4" # id 10
ampl_add = "0x99C9775E076FDF99388C029550155032Ba2d8914"

tellor_abi = '[{"inputs":[{"internalType":"address payable","name":"_master","type":"adc
ampl_abi = '[{"constant":true,"inputs":[],"name":"reportDelaySec","outputs":[{"name":"",

contract_t = w3.eth.contract(address = w3.toChecksumAddress(tellor_add), abi = tellor_ab
tellor_data = (contract_t.functions.getCurrentValue(10).call())
```

In [3]:
```
from datetime import datetime, timedelta
all_prices = []
all_timestamps = []

initial_data = tellor_data
old_date = datetime.timestamp(datetime.now() - timedelta(days = 10))
all_prices.append(tellor_data[1]/ 1000000)
all_timestamps.append(datetime.fromtimestamp(int(tellor_data[2])))
while (old_date < initial_data[2]):
    initial_data = contract_t.functions.getDataBefore(10, initial_data[2]).call()
    #print(initial_data)
    all_prices.append(initial_data[1] / 1000000)
    all_timestamps.append(datetime.fromtimestamp(int(initial_data[2])))
```

In [4]:
```
import matplotlib.pyplot as plt
%matplotlib inline

plt.plot(all_timestamps, all_prices)
```

Out[4]: [<matplotlib.lines.Line2D at 0x1be38cedb48>]



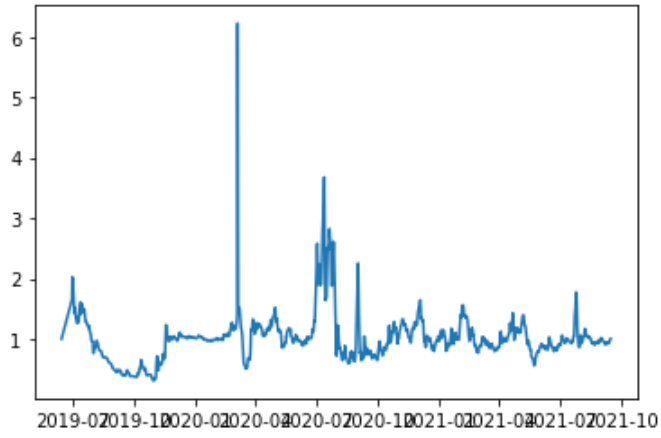
Ampleforth Data

Gotten from an API and parsed with JSON, ampleforth's data comes outside of normal contract connection. Using pandas, dictionaries and the requests library, I got the data in and looked at it on a plot to make sure it was accurate, then trimmed and scaled it to compare to Tellor's

In [5]:
```
import json
import requests
import pandas as pd

url = 'https://web-api.ampleforth.org/eth/oracle-histroy'

r = requests.get(url)
files = r.json()
ampl_dict = files["rateOracleProviderHistory"]["reports"]["ampleforth.org"]

ampl_df = pd.DataFrame(ampl_dict)
payload = list(ampl_df['payload'])
timestamps = list(ampl_df['timestampSec'])
new_timestamps = [datetime.fromtimestamp(i) for i in timestamps]
```

In [6]:
```
plt.plot(new_timestamps, payload)
```

Out[6]:
```
[<matplotlib.lines.Line2D at 0x1be3b26a988>]
```



Since we only wanted to look at the last 10 days, I cut off the data using a datetime conditional list comprehension.

In [7]:
```
end_date = datetime(2021,9,6,0,0,0)
timestamps_2 = [i for i in new_timestamps if i > end_date]
new_length = len(new_timestamps) - len(timestamps_2)
```

Get unique values from tellor data to avoid "stair step" shape of function. Reversed lists since original was in descending order going backwards through time

In [8]:
```
streamline_tx = []
streamline_ty = []

timestamps_rev = all_timestamps[::-1]
prices_rev = all_prices[::-1]

for i in range(1, len(timestamps_rev) - 1):
    if prices_rev[i] != prices_rev[i-1]:
        streamline_ty.append(prices_rev[i + 1])
        streamline_tx.append(timestamps_rev[i])
```

Chainlink Data

Used smart contract and web3 to pull data back 10 days.

In [15]:
```
chainlink_add = "0xe20CA8D7546932360e37E9D72c1a47334af57706"
chainlink_add_cs = w3.toChecksumAddress(chainlink_add)
chainlink_abi = '[{"inputs":[{"internalType":"address","name":"_aggregator","type":"addr
contract_cl = w3.eth.contract(address = chainlink_add_cs, abi = chainlink_abi)
```

In [79]:
```
latestData = contract_cl.functions.latestRoundData().call()
scale = 1e18

round_id_0 = latestData[0]
price_0 = latestData[1]
update_time_0 = latestData[3]

round_ids = []
prices = []
update_times_utc = []

round_ids.append(round_id_0)
prices.append(price_0 / scale)
update_times_utc.append(datetime.utcfromtimestamp(update_time_0))
```

In [80]:
```
current_date = str(datetime.utcfromtimestamp(update_time_0))
current_date = current_date.split()[0]
end_date = '2021-09-07'

def time_convert(raw_time):
    return datetime.utcfromtimestamp(raw_time)

def inc_rounds(curr_round_data, end_date, time_arr):

    curr_round_id = curr_round_data[0]
    current_date = str(time_convert(curr_round_data[3])).split()[0]

    while current_date != end_date:
        curr_round_id = curr_round_id - 1
        historicalData = contract_cl.functions.getRoundData(curr_round_id).call()
        update_time_raw = historicalData[3]
        time_arr.append(time_convert(update_time_raw))

        #round_ids.append(historicalData[0])
        prices.append(historicalData[1] / scale)
        #update_times_utc.append(time_convert(update_time_raw))


        current_date = str(time_convert(update_time_raw)).split()[0]
        print("time: ",time_convert(update_time_raw), "price: ", historicalData[1] / (s
```

In [81]:
```
inc_rounds(latestData, end_date, update_times_utc)
```

```
time:  2021-09-15 00:01:07 price:  1.0081175584297437 round ID:  36893488147419105617
time:  2021-09-14 00:00:37 price:  0.948489905 round ID:  36893488147419105616
time:  2021-09-13 00:00:15 price:  0.9552808618845301 round ID:  36893488147419105615
time:  2021-09-12 00:00:06 price:  0.91241296 round ID:  36893488147419105614
time:  2021-09-11 00:08:29 price:  0.90852506 round ID:  36893488147419105613
time:  2021-09-11 00:00:25 price:  0.9089066877963939 round ID:  36893488147419105612
time:  2021-09-10 00:01:22 price:  0.9352815598109805 round ID:  36893488147419105611
time:  2021-09-09 00:00:10 price:  0.938577 round ID:  36893488147419105610
time:  2021-09-08 00:03:09 price:  0.9303749540281897 round ID:  36893488147419105609
time:  2021-09-07 00:01:08 price:  0.9464961626016553 round ID:  36893488147419105608
```

Comparison Plot

Plot data on same figure. Since x axis is datetime objects, it will match up evenly and we will be able to see changes in data.

In [83]:
```
plt.figure(figsize = (15, 12))
plt.plot(timestamps_2, payload[new_length:], label = 'ampl')
#plt.plot(all_timestamps, all_prices, label = 'tellor' )
plt.plot(streamline_tx, streamline_ty, label = 'tellor')
plt.plot(update_times_utc, prices, label = 'chainlink')
plt.title("Tellor and Ampleforth Oracle Prices")
plt.xlabel("date")
plt.ylabel("AMPL price (USD)")
plt.xticks(rotation = 90)
plt.legend()
```

Out[83]: <matplotlib.legend.Legend at 0x1be3df39148>