In [1]:
```
import numpy as np
import pandas as pd
import warnings
from datetime import datetime, timedelta
from matplotlib import pyplot as plt


warnings.filterwarnings('ignore')
```

Import Data

In [2]:
```
telescopes = ['12-meter','alma','apex','aste','iram','jcmt','lmt','sma','
```

In [3]:
```
starttime = datetime(2019,10,3,6)
endtime = datetime(2019,10,14,0) # not included
timestamps = np.arange(starttime, endtime,
                       timedelta(hours=6)).astype(datetime)
databook = {}
for ts in telescopes:
    databook[ts] = dict.fromkeys(timestamps)
```

In [4]:
```
for ts in telescopes:
    for t in timestamps:
        filepath = "data/"+ ts +"/"+ t.strftime("%Y%m%d_%H:%M:%S")
        try:
            df = pd.read_csv(filepath, delim_whitespace=True, skiprows =
            df.columns = ["date", "tau225", "Tb[k]", "pwv[mm]", "lwp[kg*m
            df['date'] = pd.to_datetime(df['date'], format = "%Y%m%d_%H:%
            databook[ts][t] = df
        except FileNotFoundError:
            databook[ts][t] = None
# databook is a dictionary of dictionaries of dataframes
# keys: telescope names
# values: dictionaries of dataframes for one telescope
# databook[telescope_name] is a dictionary of dataframes for one telescop
# keys: timestamps when the forecast is made
# values: forecast dataframe (None if missing)
```

All our available data is stored in `databook` (a dict of dictionaries):

`databook` indexing:

`databook[telescope_name][timestamp]` is a dataframe of the 16-day forward forecast made at `timestamp` for `telescope_name`. See example below:

In [5]:
```
print(telescopes[0],timestamps[1])
(databook[telescopes[0]][timestamps[1]]).head()
```

```
12-meter 2019-10-03 12:00:00
```

Out[5]:

	date	tau225	Tb[k]	pwv[mm]	lwp[kg*m^-2]	iwp[kg*m^-2]	o3[DU]
0	2019-10-03 12:00:00	0.76472	154.45	12.945	0.000000	0.000000	252.87
1	2019-10-03 13:00:00	0.76153	154.05	12.879	0.000000	0.000000	253.39
2	2019-10-03 14:00:00	0.76019	153.85	12.872	0.000000	0.000000	253.79
3	2019-10-03 15:00:00	0.00000	0.00	0.000	0.000000	0.000000	0.00
4	2019-10-03 16:00:00	0.90973	171.53	15.391	0.006263	0.000001	253.53

Visualize forecast

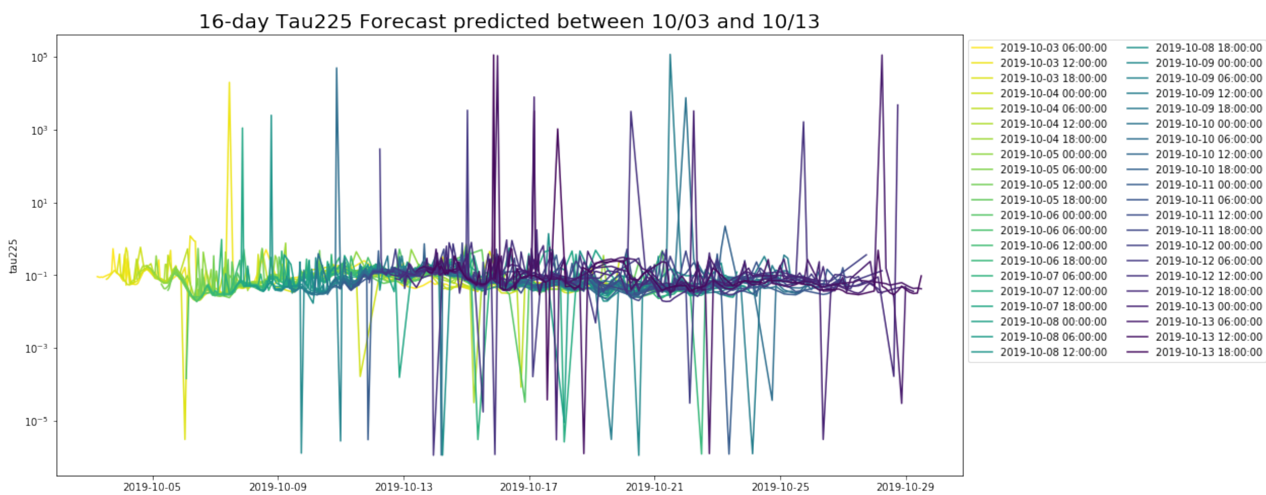
For the EDA below, we use the data for telescope('apex') as an example.

In [6]:
```
data_telescope = databook['apex']
```

In [26]:

```
import matplotlib.cm as cm
#cycle through colors from a continuous color map
plt.rcParams["axes.prop_cycle"] = plt.cycler("color", plt.cm.viridis(np.l

plt.figure(figsize = (16,8))
for t in data_telescope:
    df = data_telescope[t]
    if df is not None:
        mask = df['tau225']!=0
        plt.plot(df['date'][mask], df['tau225'][mask], label=t)
plt.legend(ncol = 2,bbox_to_anchor=(1, 1))
plt.title("16-day Tau225 Forecast predicted between 10/03 and 10/13",font
plt.ylabel("tau225")
plt.yscale('log')
plt.show()
```



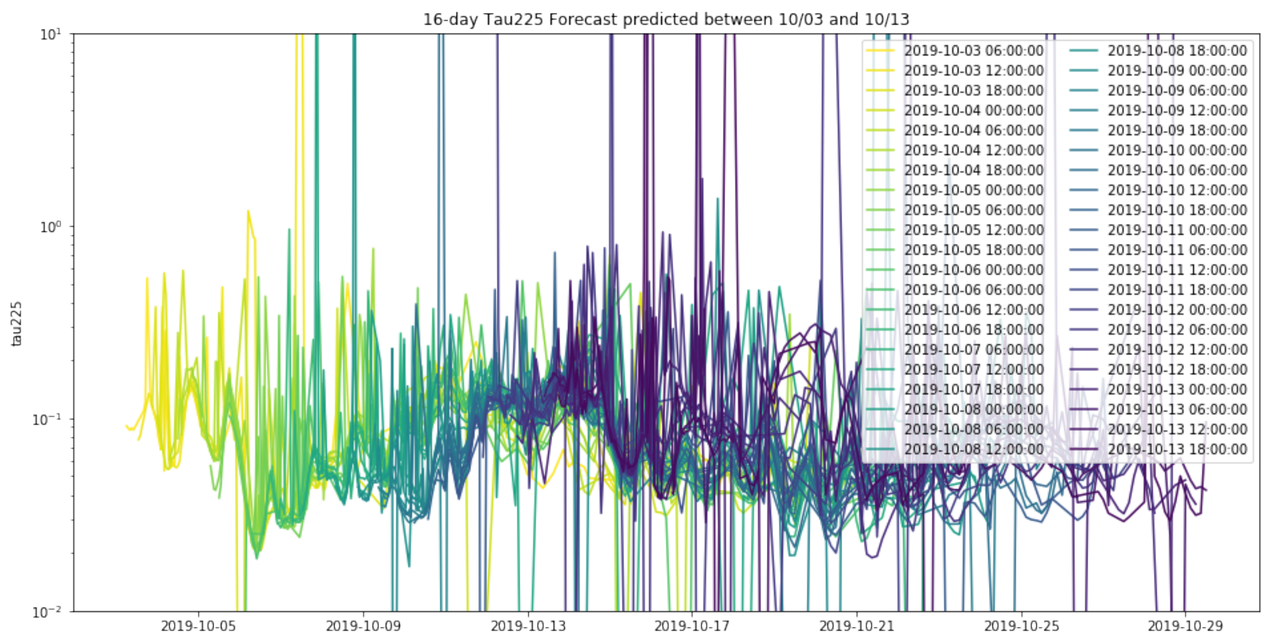
With the use of color map, we denotes the more recent the prediction the deeper

As we can see, there are some really high spikes. Are these outliers? (need to be validated)

Zoom in a little bit:

In [15]:

```
plt.figure(figsize = (16,8))
for t in data_telescope:
    df = data_telescope[t]
    if df is not None:
        mask = df['tau225']!=0
        plt.plot(df['date'][mask], df['tau225'][mask], label=t)
plt.legend(ncol = 2)
plt.title("16-day Tau225 Forecast predicted between 10/03 and 10/13")
plt.ylabel("tau225")
plt.yscale('log')
plt.ylim(0.01,10)
plt.show()
```



It seems that the forecast has some variance.

In [ ]:

```
```

Compute variance

1. Variance v.s. date of prediction

For each date/hour, we have been making prediction of it since 16 days before that date/hour. How does those different predictions about the atmosphere condition at the same datetime vary? Knowing that may give us an overall sense of how accurate the forecast is.

We still use the weather prediction for 'apex' as the example to compute the variance.

In [30]:
```
df_all = pd.concat([data_telescope[t] for t in data_telescope], axis =0)
df_tau_all = df_all.groupby('date').agg({'tau225':lambda x: list(x)})

df_tau_all['tau225_mean'] = df_tau_all['tau225'].apply(lambda x: np.mean(
df_tau_all['tau225_min'] = df_tau_all['tau225'].apply(lambda x: np.min(x)
df_tau_all['tau225_max'] = df_tau_all['tau225'].apply(lambda x: np.max(x)
df_tau_all['tau225_true'] = df_tau_all['tau225'].apply(lambda x: x[-1])
df_tau_all['tau225_latest'] = df_tau_all['tau225'].apply(lambda x: x[-2]

df_tau_all.reset_index(inplace = True)
```

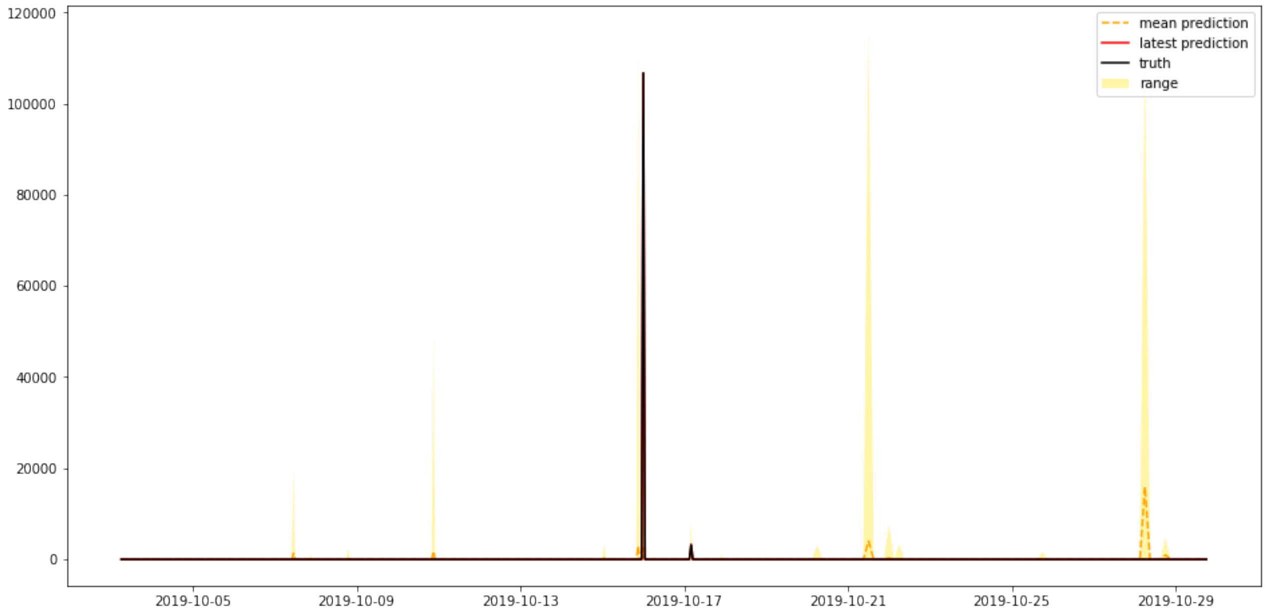
In [240]:
```
df_tau_all.iloc[20:25]
```

Out[240]:

	date	tau225	tau225_mean	tau225_min	tau225_max	tau225_true	tau225_l
20	2019-10-04 02:00:00	[0.0, 0.068749, 0.0, 0.081689]	0.037610	0.000000	0.081689	0.081689	0.00
21	2019-10-04 03:00:00	[0.35504, 0.37931, 0.069635]	0.267995	0.069635	0.379310	0.069635	0.37
22	2019-10-04 04:00:00	[0.0, 0.053272, 0.56715]	0.206807	0.000000	0.567150	0.567150	0.05
23	2019-10-04 05:00:00	[0.0, 0.2847, 0.43477, 0.055701]	0.193793	0.000000	0.434770	0.055701	0.43
24	2019-10-04 06:00:00	[0.0, 0.0, 0.054242999999999999, 0.0]	0.013561	0.000000	0.054243	0.000000	0.05

Here we re-arrange the dataframe. Each row collected all the prediction for the time in 'date' as a list (here we only use 'tau225' for now. We also calculate the mean, range(min, max) and use the last prediction value as the true weather condition(predicting the weather on the time we are on now), and the second last prediction as the latest prediction.

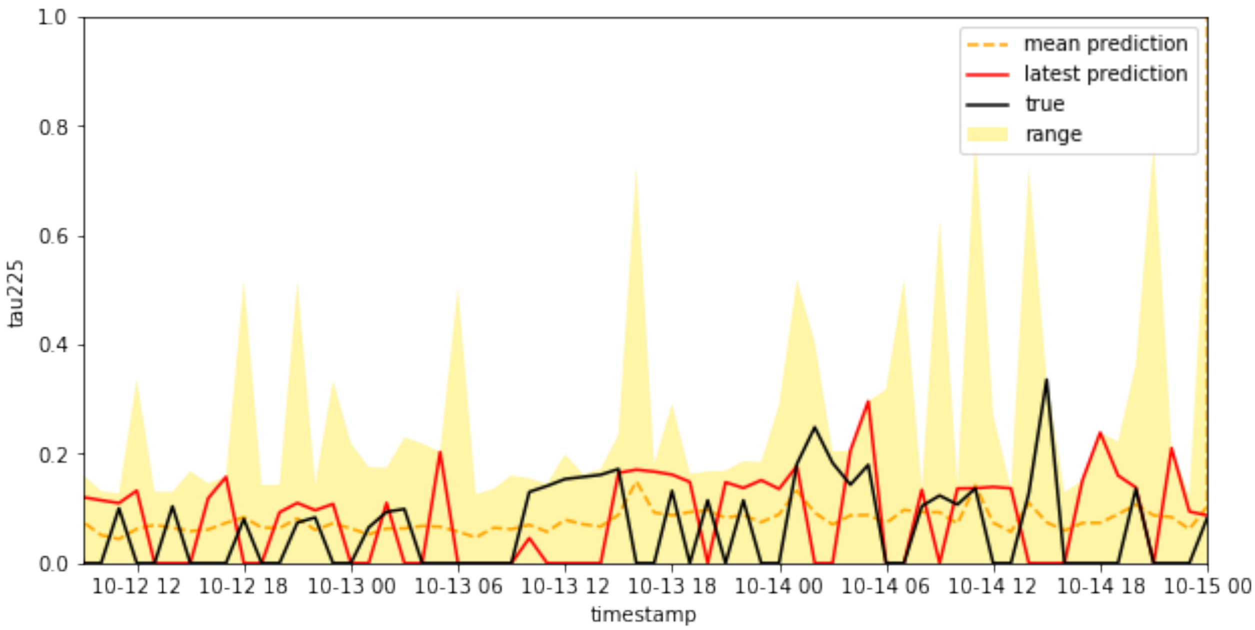
In [259]:
```
plt.figure(figsize = (16,8))
plt.plot(df_tau_all.date, df_tau_all.tau225_mean, c = 'orange', ls = '--'
plt.plot(df_tau_all.date, df_tau_all.tau225_latest, c = 'r', label = 'lat
plt.plot(df_tau_all.date, df_tau_all.tau225_latest, c = 'black', label = 
plt.fill_between(df_tau_all.date, df_tau_all.tau225_max, df_tau_all.tau22
plt.legend();
```



Here we use the yellow area to denote the range, blue represents the lastest prediction and red represnts the weather predictions. Since here are some really strong spikes, it's better for us to zoom in a little bit to see the relationship between those three.

We chose a period in the middle (10/12 09:00 to 10/15 00:00) as it have enough predictions and are well-representative.

In [258]:
```
plt.figure(figsize = (10,5))
plt.plot(df_tau_all.date, df_tau_all.tau225_mean, c = 'orange', ls = '--',
plt.plot(df_tau_all.date, df_tau_all.tau225_latest, c = 'r', label = 'late
plt.plot(df_tau_all.date, df_tau_all.tau225_true, c = 'black', label = 'tr
plt.fill_between(df_tau_all.date, df_tau_all.tau225_max, df_tau_all.tau225
plt.ylim(0,1)
plt.xlim('2019-10-12 9:00:00', '2019-10-15 00:00:00')
plt.ylim()
plt.legend(fontsize = 10)
plt.ylabel('tau225', fontsize = 10)
plt.xlabel('timestamp', fontsize = 10);
```



From the plot above we can see that, we got a range of predictions for each time. Also, the latest prediction for each time also differs from the mean prediction greatly, and also differs from the true weather prediction.

One other thing also noticable is that there are several zeros for the true weather condition.

Small Take-away

- Things needs further check:
  - the spikes
  - the zeros
- There are lot of freedom in how to choose the prediction and how to measure the uncertainty.

2. Variance v.s. number of days forward

It makes sense that the further future the GFS model is trying to predict, the less certain it will be. We will estimate the variance of forecast v.s. number of days forward. This could be used as the uncertainty measure for our scheduling strategy.

In [112]:

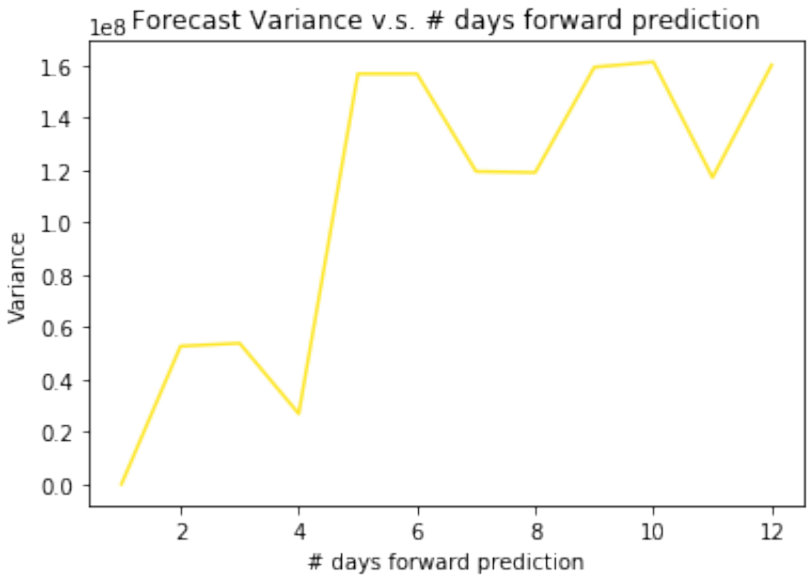
```
# concatenate all tau225 for one telescope(apex)
# columns are the timestamps when predictions were made, indices are time
df_tau225 = pd.concat([data_telescope[t].set_index('date')['tau225'] for
df_tau225.columns = [t for t in data_telescope if data_telescope[t] is no
df_tau225

# use df_mask to record the number of days forward (index - col) for each
df_mask = pd.DataFrame(index = df_tau225.index, columns = df_tau225.colum
for col in df_tau225.columns:
    df_mask[col] = (df_tau225.index - col).days

# get 'true' tau225 value, which is the closest prediction within a day
true_tau225 = pd.Series(index = df_tau225.index)
for idx in df_tau225.index:
    values = df_tau225.loc[idx]
    true_tau225[idx] = values.values[~values.isna()][-1]

# compute variance v.s. # days forward
df_diff = (df_tau225.T - true_tau225).T
var = []
for i in range(1, df_mask.max().max()+1):
    var.append(np.nanmean(df_diff[df_mask==i].values ** 2))
```

In [113]:
```python
plt.plot(range(1,13), var[:12])
plt.title("Forecast Variance v.s. # days forward prediction")
plt.ylabel("Variance")
plt.xlabel("# days forward prediction")
plt.show()
```



Overall, it makes sense that the further we want to predict, the larger variance we got. However, we would expect this plot to be a smooth line having upward trend. This ill-behavior might have something to do with the extreme values (zeros and large values) in the data we pulled. As shown in the plot below, the difference beween one-day-forward predictions and the true values is either 0 or very large. This could distort our estimation of the true variance of the forecast.

In [123]:
```
# one day forward prediction
one_day_diff = df_diff[df_mask==1].values
one_day_diff = one_day_diff.reshape(-1)
one_day_diff = one_day_diff[~np.isnan(one_day_diff)]
plt.hist(one_day_diff)
plt.title("Distribution of difference between one-day-forward prediction
plt.ylabel("Frequency")
plt.xlabel("Difference")
plt.show()
```