CE 596 - Project 1

Thomas Thelen

May 2022 flooding in Carolina Beach

notebook file path: C:\Users\ththelen\OneDrive - North Carolina State University\CarolinaBeach\SMS\20221103_NC9rev2.1+0.15m-May2022\ADCIRC\CE596_Proj1\plotting

Question 1

Why did you select this storm? What region was impacted by this storm?
The sequential offshore pressure system and king tides in May 2022 caused flooding throughout coastal North Carolina. The project team developed new tidal, atmospheric and riverine forcings for the simulation. We will review water levels at and around Carolina Beach, NC as part of work to support Sunny Day Flooding Project model development.

Does the wind files contain enough parameters for the wind model within ADCIRC? What timeframe will you simulate?
We use the North American Mesoscale (NAM) analyses product to obtain wind and pressure forcings for ADCRIC. The NAM files are converted to a OWI hindcast format that is recognizable by ADCIRC. The simulation is run in two legs, both of which are forced by atmospheric, tidal, and riverine forcings. The first leg is a nine day simulation ramp period that spans Apr 30 - May 9. The second leg from May 9 - May 23 has full forcings (tidal, atmospheric, riverine) and includes the period of flooding in Carolina Beach.

What observation data are available for validation of your model results? What NOAA stations are available for validation of tide and surge levels along the coast? What NDBC buoys are available for validation of winds and waves in open water?

Coastal water levels: NOAA stations at Wilmington and Wrightsville Beach, FIMAN gauge at Carolina Beach

Open water waves: NBDC bouys at Frying Pan Shoals, Wilmington Harbor, Masonboro Inlet, Onslow Bay

Open water winds: NBDC bouys at Frying Pan Shoals, Wrightsville Beach Offshore and Neashore, Onslow Bay

Coastal winds: Stations at Wrightville Beach Pier, Masonboro Island (north and south)

Question 2

Compare the ADCIRC results to the measured tide levels at your NOAA stations. How well do the predictions match to the measurements? What could be done to improve this comparison? Is it okay to proceed with the storm simulation?
Because the ramp portion of the simulation includes tidal, atmospheric, and riverine forcing per TT discussion with CD to epxedite model runs, we do not have a tides-only simulation to compare with NOAA predicted water levels. If the research team thinks that this simulation would be useful to better understand results, we could look at setting up a tides-only spinup in a future model run.

Note that this simulation was run on mesh NC9rev2.1. See Project 2 for a complete description of this mesh and boundary conditions.

Question 3

Using the methodology developed in HW 4, develop a simulation with winds and waves during the storm. Hot-start this simulation from the end of the previous simulation.

In [107…

```
""" Import packages to script. """

import os
import pandas as pd
import csv
import numpy as np
import matplotlib.pyplot as plt
from datetime import datetime, timedelta
import requests
import seaborn as sns
import matplotlib.dates as mdates

from erddapy import ERDDAP
import xarray as xr
import netCDF4
import gc
import glob

def pathChecker(prompt):
    '''Takes a string prompt for path input and does not continue until path esits'''
```

```
    inPath = input(prompt)
    inPath = inPath.strip('"')
    while os.path.exists(inPath) == False:
        print("Oops, looks like your path name contained an error. Try again.")
        inPath = input(prompt)
        inPath = inPath.strip('"')
    return(inPath)
```

Read mesh and element from ADCIRC file.

- Parameter (var, units) - mesh
- water level (zeta, m) - fort.63.nc
- wind speed (windx/windy, m/s) - fort.74.nc
- water velocity (u-vel/v-vel, m/s) - fort.64.nc
- significant wave height (swan_HS, m) - swan_HS.nc

User provides simulation start and time in UTC

e.g., results from a 14-day simulation starting 0000 UTC 1 Nov and ending 0000 UTC 15 Nov. ADCIRC writes its first output after the first interval in the simulation, so the first snap in the result files will correspond to 0100 UTC 1 Nov. Then they are hourly after that.

In [108…

```
m_start = "2022-05-09 01:00"
#m_start = input("Enter date and time in UTC corresponding to first ADCIRC output time step (YYYY-MM-DD HH:MM)."\
#                "\nNote this is one hour after the simulation start time: ")
dt_start_utc = datetime.strptime(m_start, "%Y-%m-%d %H:%M")
dt_start_date = dt_start_utc.date()

print(dt_start_utc)
print(dt_start_date)
```

```
2022-05-09 01:00:00
2022-05-09
```

Question 3.1 - water levels

Compare the ADCIRC results to the measured water levels at your NOAA stations. How well do the predictions match to the measurements? What could be done to improve this comparison?

Provide end date to bound NOAA WL API request.

In [109…
```
m_end = '2022-05-22'
#m_end = input("Simulation start date is {}\nInput the date through which you want to pull NOAA data (YYYY-MM-DD): ".f
dt_end_utc = datetime.strptime(m_end, "%Y-%m-%d")
dt_end_date = dt_end_utc.date()

dt_end_date
```

Out[109]:
```
datetime.date(2022, 5, 22)
```

Water level comparison: Wrightsville Beach

Import water levels from NOAA API

In [110…
```
#station = input("What NOAA station do you want to pull data from (wright or wilm)? Alternatively, enter a station ID:
station = 'wright'
if station == 'wright':
    sta_ID = '8658163'
elif station == 'wilm':
    sta_ID = '8658120'
elif int(station) > 0:
    sta_ID = station

NOAA_start_date = dt_start_date.strftime('%Y%m%d')
NOAA_end_date = dt_end_date.strftime('%Y%m%d')

NOAA_url = "https://api.tidesandcurrents.noaa.gov/api/prod/datagetter?begin_date={}&end_date={}&station={}&product=hou
"&datum=NAVD&time_zone=gmt&units=metric&format=json".format(NOAA_start_date, NOAA_end_date,sta_ID)
NOAA_response = requests.get(NOAA_url)
NOAA_json = NOAA_response.json()
NOAA_metadata = NOAA_json['metadata']
NOAA_df = pd.DataFrame(NOAA_json['data'])
NOAA_df = NOAA_df.drop(columns=['s', 'f'])
NOAA_df.columns = ['NOAA Timestep (UTC)', 'NOAA WL (m NAVD88)']
NOAA_df.index = pd.to_datetime(NOAA_df['NOAA Timestep (UTC)']).dt.tz_localize('UTC')
#NOAA_df['NOAA Timestep (UTC)'] = NOAA_df['NOAA Timestep (UTC)'].dt.tz_localize('UTC')
```

```
NOAA_df['NOAA WL (m NAVD88)'] = NOAA_df['NOAA WL (m NAVD88)'].astype(float)
NOAA_df['NOAA WL (ft NAVD88)'] = NOAA_df['NOAA WL (m NAVD88)'] * 3.281
print(NOAA_metadata)
NOAA_df
```

```
{'id': '8658163', 'name': 'Wrightsville Beach', 'lat': '34.2133', 'lon': '-77.7867'}
```

Out[110]:

NOAA Timestep (UTC)	NOAA Timestep (UTC)	NOAA WL (m NAVD88)	NOAA WL (ft NAVD88)
2022-05-09 00:00:00+00:00	2022-05-09 00:00	-0.238	-0.780878
2022-05-09 01:00:00+00:00	2022-05-09 01:00	-0.155	-0.508555
2022-05-09 02:00:00+00:00	2022-05-09 02:00	0.015	0.049215
2022-05-09 03:00:00+00:00	2022-05-09 03:00	0.254	0.833374
2022-05-09 04:00:00+00:00	2022-05-09 04:00	0.537	1.761897
...	...	...	...
2022-05-22 19:00:00+00:00	2022-05-22 19:00	0.358	1.174598
2022-05-22 20:00:00+00:00	2022-05-22 20:00	0.171	0.561051
2022-05-22 21:00:00+00:00	2022-05-22 21:00	-0.088	-0.288728
2022-05-22 22:00:00+00:00	2022-05-22 22:00	-0.425	-1.394425
2022-05-22 23:00:00+00:00	2022-05-22 23:00	-0.585	-1.919385

336 rows × 3 columns

Import ADCIRC water levels from text file interpolated from ADCIRC fort.63.nc output

In [111…

```
m_data_wright = "C:/Users/ththelen/OneDrive - North Carolina State University/CarolinaBeach/SMS/20221103_NC9rev2.1+0.1
#m_data_wright = pathChecker("Enter full path of pullT text file: ")

m_df_wright =  pd.read_csv(m_data_wright, sep=' ', header=None)
                    #engine='python', name=['ADCRIC Timestep (hr)', 'ADCIRC WL (m)'])
m_df_wright.columns = ['ADCRIC Timestep (hr)', 'WL (m NAVD88)']
m_df_wright['WL (ft NAVD88)'] = m_df_wright['WL (m NAVD88)'] * 3.281
m_df_wright['Time Delta (hrs)'] = pd.to_timedelta(m_df_wright['ADCRIC Timestep (hr)'], unit='h')
m_df_wright['ADCIRC Timestep (UTC)'] = dt_start_utc + m_df_wright['Time Delta (hrs)']
m_df_wright['ADCIRC Timestep (UTC)'] = m_df_wright['ADCIRC Timestep (UTC)'].dt.tz_localize('UTC')
```

```
m_df_wright.index = m_df_wright['ADCIRC Timestep (UTC)']

m_df_wright
```

Out[111]:

ADCIRC Timestep (UTC)	ADCRIC Timestep (hr)	WL (m NAVD88)	WL (ft NAVD88)	Time Delta (hrs)	ADCIRC Timestep (UTC)
2022-05-09 01:00:00+00:00	0	-0.018076	-0.059308	0 days 00:00:00	2022-05-09 01:00:00+00:00
2022-05-09 02:00:00+00:00	1	0.100843	0.330867	0 days 01:00:00	2022-05-09 02:00:00+00:00
2022-05-09 03:00:00+00:00	2	0.240762	0.789939	0 days 02:00:00	2022-05-09 03:00:00+00:00
2022-05-09 04:00:00+00:00	3	0.395762	1.298496	0 days 03:00:00	2022-05-09 04:00:00+00:00
2022-05-09 05:00:00+00:00	4	0.578310	1.897434	0 days 04:00:00	2022-05-09 05:00:00+00:00
...	...	...	...	...	...
2022-05-22 20:00:00+00:00	331	0.041639	0.136617	13 days 19:00:00	2022-05-22 20:00:00+00:00
2022-05-22 21:00:00+00:00	332	-0.223471	-0.733209	13 days 20:00:00	2022-05-22 21:00:00+00:00
2022-05-22 22:00:00+00:00	333	-0.470212	-1.542766	13 days 21:00:00	2022-05-22 22:00:00+00:00
2022-05-22 23:00:00+00:00	334	-0.611521	-2.006399	13 days 22:00:00	2022-05-22 23:00:00+00:00
2022-05-23 00:00:00+00:00	335	-0.627017	-2.057242	13 days 23:00:00	2022-05-23 00:00:00+00:00

336 rows × 5 columns

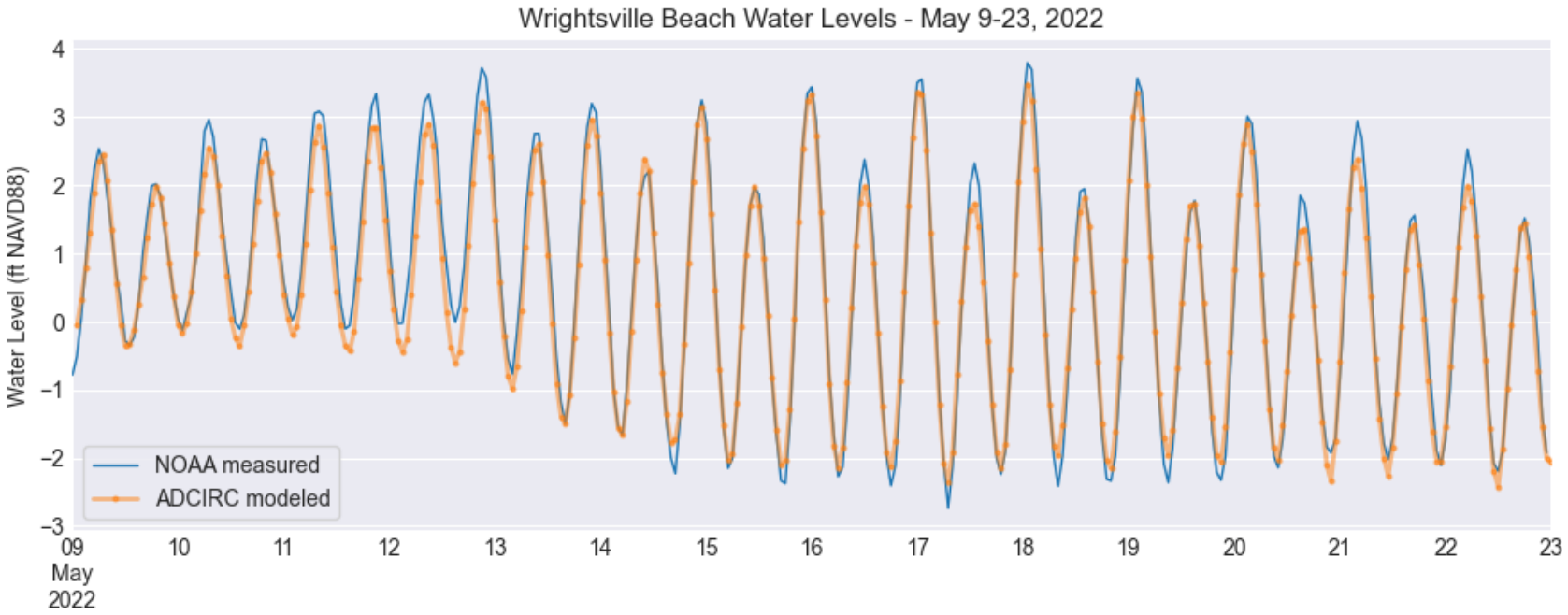
In [112…

```
pltUnit = input("Plot in feet or meters (ft or m)? ")

""" Plot comparison lines on chart. """
fig, ax = plt.subplots(figsize=(12, 4))

NOAA_df[f'NOAA WL ({pltUnit} NAVD88)'].plot(ax = ax, linewidth = 1, ls = '-', marker ='', markersize = 1)

m_df_wright[f'WL ({pltUnit} NAVD88)'].plot(ax = ax, linewidth = 2, ls = '-', marker = '.', markersize = 4, alpha = 0.5

ax.set_xlabel(' ')
plt.style.use('seaborn-darkgrid')

ax.set_ylabel('Water Level ({} NAVD88)'.format(pltUnit))
```

```
ax.set_title('Wrightsville Beach Water Levels - May 9-23, 2022')
ax.legend(['NOAA measured', 'ADCIRC modeled'], frameon = True)
```

```
Plot in feet or meters (ft or m)? ft
```

Out[112]: <matplotlib.legend.Legend at 0x1e9da35ae20>



Water level comparison: Wilmington

Import water levels from NOAA API

In [113…

```
#station = input("What NOAA station do you want to pull data from (wright or wilm)? Alternatively, enter a station ID:
station = 'wilm'
if station == 'wright':
    sta_ID = '8658163'
elif station == 'wilm':
    sta_ID = '8658120'
elif int(station) > 0:
    sta_ID = station

NOAA_start_date = dt_start_date.strftime('%Y%m%d')
```

```
NOAA_end_date = dt_end_date.strftime('%Y%m%d')

NOAA_url_wilm = "https://api.tidesandcurrents.noaa.gov/api/prod/datagetter?begin_date={}&end_date={}&station={}&produc
"&datum=NAVD&time_zone=gmt&units=metric&format=json".format(NOAA_start_date, NOAA_end_date,sta_ID)
NOAA_response_wilm = requests.get(NOAA_url_wilm)
NOAA_json_wilm = NOAA_response_wilm.json()
NOAA_metadata_wilm = NOAA_json_wilm['metadata']
NOAA_df_wilm = pd.DataFrame(NOAA_json_wilm['data'])
NOAA_df_wilm = NOAA_df_wilm.drop(columns=['s', 'f'])
NOAA_df_wilm.columns = ['NOAA Timestep (UTC)', 'NOAA WL (m NAVD88)']
NOAA_df_wilm.index = pd.to_datetime(NOAA_df_wilm['NOAA Timestep (UTC)']).dt.tz_localize('UTC')
#NOAA_df_wilm['NOAA Timestep (UTC)'] = NOAA_df['NOAA Timestep (UTC)'].dt.tz_localize('UTC')
NOAA_df_wilm['NOAA WL (m NAVD88)'] = NOAA_df_wilm['NOAA WL (m NAVD88)'].astype(float)
NOAA_df_wilm['NOAA WL (ft NAVD88)'] = NOAA_df_wilm['NOAA WL (m NAVD88)'] * 3.281
print(NOAA_metadata_wilm)
NOAA_df_wilm
```

```
{'id': '8658120', 'name': 'Wilmington', 'lat': '34.2275', 'lon': '-77.9536'}
```

Out[113]:

NOAA Timestep (UTC)	NOAA Timestep (UTC)	NOAA WL (m NAVD88)	NOAA WL (ft NAVD88)
2022-05-09 00:00:00+00:00	2022-05-09 00:00	-0.249	-0.816969
2022-05-09 01:00:00+00:00	2022-05-09 01:00	-0.401	-1.315681
2022-05-09 02:00:00+00:00	2022-05-09 02:00	-0.487	-1.597847
2022-05-09 03:00:00+00:00	2022-05-09 03:00	-0.429	-1.407549
2022-05-09 04:00:00+00:00	2022-05-09 04:00	-0.186	-0.610266
...	...	...	...
2022-05-22 19:00:00+00:00	2022-05-22 19:00	0.579	1.899699
2022-05-22 20:00:00+00:00	2022-05-22 20:00	0.688	2.257328
2022-05-22 21:00:00+00:00	2022-05-22 21:00	0.670	2.198270
2022-05-22 22:00:00+00:00	2022-05-22 22:00	0.536	1.758616
2022-05-22 23:00:00+00:00	2022-05-22 23:00	0.182	0.597142

336 rows × 3 columns

Import ADCIRC water levels from text file interpolated from ADCIRC fort.63.nc output

In [114…

```
m_data_wilm = "C:/Users/ththelen/OneDrive - North Carolina State University/CarolinaBeach/SMS//20221103_NC9rev2.1+0.15
#m_data_wilm = pathChecker("Enter full path of pullT text file: ")

m_df_wilm =  pd.read_csv(m_data_wilm, sep=' ', header=None)
                   #engine='python', name=['ADCRIC Timestep (hr)', 'ADCIRC WL (m)'])
m_df_wilm.columns = ['ADCRIC Timestep (hr)', 'WL (m NAVD88)']
m_df_wilm['WL (ft NAVD88)'] = m_df_wilm['WL (m NAVD88)'] * 3.281
m_df_wilm['Time Delta (hrs)'] = pd.to_timedelta(m_df_wilm['ADCRIC Timestep (hr)'], unit='h')
m_df_wilm['ADCIRC Timestep (UTC)'] = dt_start_utc + m_df_wilm['Time Delta (hrs)']
m_df_wilm['ADCIRC Timestep (UTC)'] = m_df_wilm['ADCIRC Timestep (UTC)'].dt.tz_localize('UTC')

m_df_wilm.index = m_df_wilm['ADCIRC Timestep (UTC)']

m_df_wilm
```

Out[114]:

ADCIRC Timestep (UTC)	ADCRIC Timestep (hr)	WL (m NAVD88)	WL (ft NAVD88)	Time Delta (hrs)	ADCIRC Timestep (UTC)
2022-05-09 01:00:00+00:00	0	-0.210866	-0.691853	0 days 00:00:00	2022-05-09 01:00:00+00:00
2022-05-09 02:00:00+00:00	1	-0.259717	-0.852131	0 days 01:00:00	2022-05-09 02:00:00+00:00
2022-05-09 03:00:00+00:00	2	-0.238452	-0.782360	0 days 02:00:00	2022-05-09 03:00:00+00:00
2022-05-09 04:00:00+00:00	3	-0.082714	-0.271385	0 days 03:00:00	2022-05-09 04:00:00+00:00
2022-05-09 05:00:00+00:00	4	0.110101	0.361242	0 days 04:00:00	2022-05-09 05:00:00+00:00
…	…	…	…	…	…
2022-05-22 20:00:00+00:00	331	0.605907	1.987980	13 days 19:00:00	2022-05-22 20:00:00+00:00
2022-05-22 21:00:00+00:00	332	0.556788	1.826823	13 days 20:00:00	2022-05-22 21:00:00+00:00
2022-05-22 22:00:00+00:00	333	0.302055	0.991042	13 days 21:00:00	2022-05-22 22:00:00+00:00
2022-05-22 23:00:00+00:00	334	0.063019	0.206764	13 days 22:00:00	2022-05-22 23:00:00+00:00
2022-05-23 00:00:00+00:00	335	-0.169600	-0.556458	13 days 23:00:00	2022-05-23 00:00:00+00:00

336 rows × 5 columns

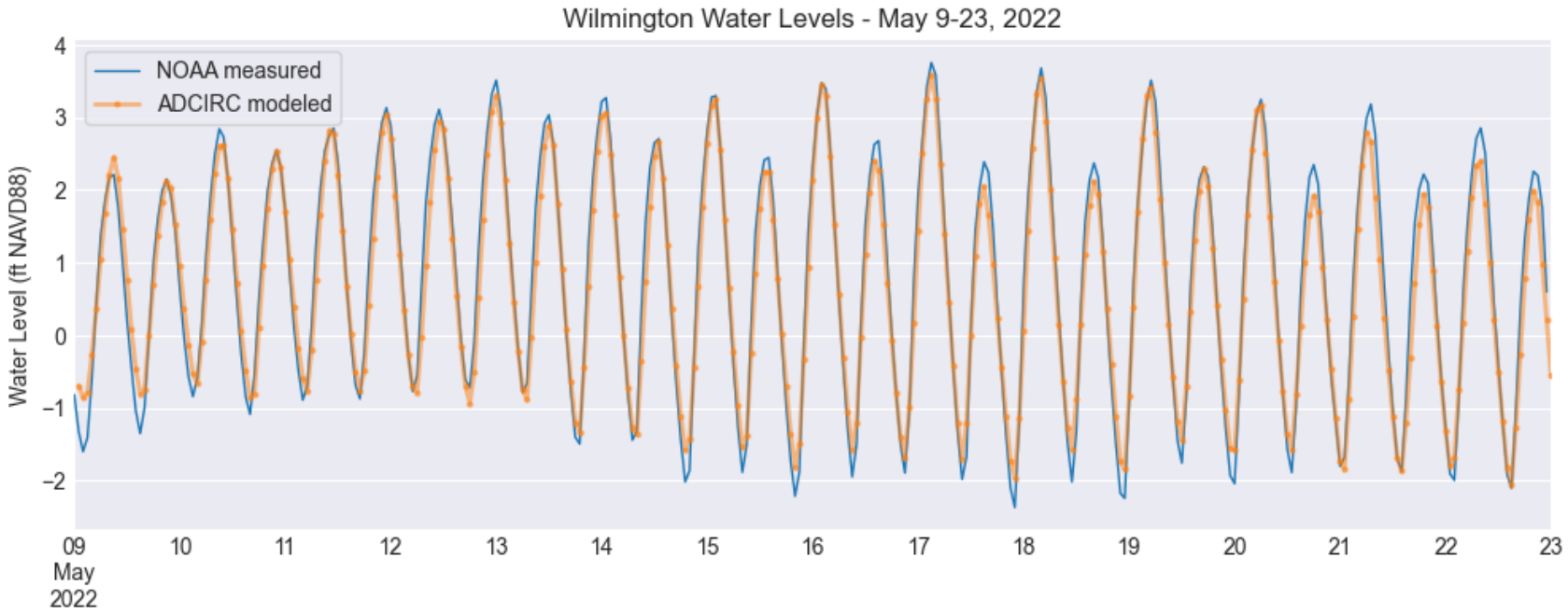
In [115…

```
pltUnit = input("Plot in feet or meters (ft or m)? ")

""" Plot comparison lines on chart. """
fig, ax = plt.subplots(figsize=(12, 4))

NOAA_df_wilm[f'NOAA WL ({pltUnit} NAVD88)'].plot(ax = ax, linewidth = 1, ls = '-', marker ='', markersize = 1)

m_df_wilm[f'WL ({pltUnit} NAVD88)'].plot(ax = ax, linewidth = 2, ls = '-', marker = '.', markersize = 4, alpha = 0.5)


ax.set_xlabel(' ')
#ax.xaxis.set_major_locator(mdates.DayLocator(interval=1))
#ax.xaxis.set_major_formatter(mdates.DateFormatter("%b %d"))
plt.style.use('seaborn-darkgrid')

ax.set_ylabel('Water Level ({} NAVD88)'.format(pltUnit))
ax.set_title('Wilmington Water Levels - May 9-23, 2022')
ax.legend(['NOAA measured', 'ADCIRC modeled'], frameon = True)
```

```
Plot in feet or meters (ft or m)? ft
```

Out[115]:

```
<matplotlib.legend.Legend at 0x1e9dabd07c0>
```



Water level comparison: Carolina Beach

Import ADCIRC water levels from text file interpolated from ADCIRC fort.63.nc output

In [116…

```
m_data = "C:/Users/ththelen/OneDrive - North Carolina State University/CarolinaBeach/SMS/20221103_NC9rev2.1+0.15m-May2
#m_data = pathChecker("Enter full path of pullT text file: ")

m_df =  pd.read_csv(m_data, sep=' ', header=None)
                    #engine='python', name=['ADCRIC Timestep (hr)', 'ADCIRC WL (m)'])
m_df.columns = ['ADCRIC Timestep (hr)', 'ADCIRC WL (m NAVD88)']
m_df['ADCIRC WL (ft NAVD88)'] = m_df['ADCIRC WL (m NAVD88)'] * 3.281
m_df['Time Delta (hrs)'] = pd.to_timedelta(m_df['ADCRIC Timestep (hr)'], unit='h')
m_df['ADCIRC Timestep (UTC)'] = dt_start_utc + m_df['Time Delta (hrs)']
m_df['ADCIRC Timestep (UTC)'] = m_df['ADCIRC Timestep (UTC)'].dt.tz_localize('UTC')

m_df.index = m_df['ADCIRC Timestep (UTC)']

dt_end_utc = m_df.iat[-1, 4]
dt_end_date = dt_end_utc.date() # Calculate simulation end date to provide for NOAA value lookup

m_df
```

Out[116]:

ADCIRC Timestep (UTC)	ADCRIC Timestep (hr)	ADCIRC WL (m NAVD88)	ADCIRC WL (ft NAVD88)	Time Delta (hrs)	ADCIRC Timestep (UTC)
2022-05-09 01:00:00+00:00	0	-0.101705	-0.333695	0 days 00:00:00	2022-05-09 01:00:00+00:00
2022-05-09 02:00:00+00:00	1	-0.051072	-0.167567	0 days 01:00:00	2022-05-09 02:00:00+00:00
2022-05-09 03:00:00+00:00	2	0.057224	0.187752	0 days 02:00:00	2022-05-09 03:00:00+00:00
2022-05-09 04:00:00+00:00	3	0.200301	0.657189	0 days 03:00:00	2022-05-09 04:00:00+00:00
2022-05-09 05:00:00+00:00	4	0.385292	1.264144	0 days 04:00:00	2022-05-09 05:00:00+00:00
...	...	...	...	...	...
2022-05-22 20:00:00+00:00	331	0.363019	1.191064	13 days 19:00:00	2022-05-22 20:00:00+00:00
2022-05-22 21:00:00+00:00	332	0.205489	0.674208	13 days 20:00:00	2022-05-22 21:00:00+00:00
2022-05-22 22:00:00+00:00	333	-0.022858	-0.074997	13 days 21:00:00	2022-05-22 22:00:00+00:00
2022-05-22 23:00:00+00:00	334	-0.276838	-0.908306	13 days 22:00:00	2022-05-22 23:00:00+00:00
2022-05-23 00:00:00+00:00	335	-0.476892	-1.564682	13 days 23:00:00	2022-05-23 00:00:00+00:00

336 rows × 5 columns

Input FIMAN WLs from .csv file downloaded from Contrail data repository and convert timestamp to UTC

In [117…

```
FIMAN_data = "C:/Users/ththelen/OneDrive - North Carolina State University/CarolinaBeach/SMS/20221103_NC9rev2.1+0.15m-
#FIMAN_data = pathChecker("Enter full path of FIMAN .csv file: ")

FIMAN_df =  pd.read_csv(FIMAN_data, sep=',')
FIMAN_df['Measurement Time (US/Eastern)'] = pd.to_datetime(FIMAN_df['Reading'], format = '%Y-%m-%d %H:%M:%S', errors =
```

```
FIMAN_df['Measurement Time (US/Eastern)'] = FIMAN_df['Measurement Time (US/Eastern)'].dt.tz_localize('US/Eastern', amb
FIMAN_df['FIMAN Timestep (UTC)'] = FIMAN_df['Measurement Time (US/Eastern)'].dt.tz_convert('UTC')

FIMAN_df['FIMAN WL (ft NAVD88)'] = FIMAN_df['Value'].astype(float)
FIMAN_df['FIMAN WL (m NAVD88)'] = FIMAN_df['FIMAN WL (ft NAVD88)'] / 3.281

measWLm = FIMAN_df['FIMAN WL (m NAVD88)']
measWLft = FIMAN_df['FIMAN WL (ft NAVD88)']
measTime = FIMAN_df['FIMAN Timestep (UTC)']

FIMAN_df.index = FIMAN_df['FIMAN Timestep (UTC)']

FIMAN_df = FIMAN_df.drop(columns=['Reading', 'Receive', 'Value', 'Unit', 'Data Quality'])

FIMAN_df
```

Out[117]:

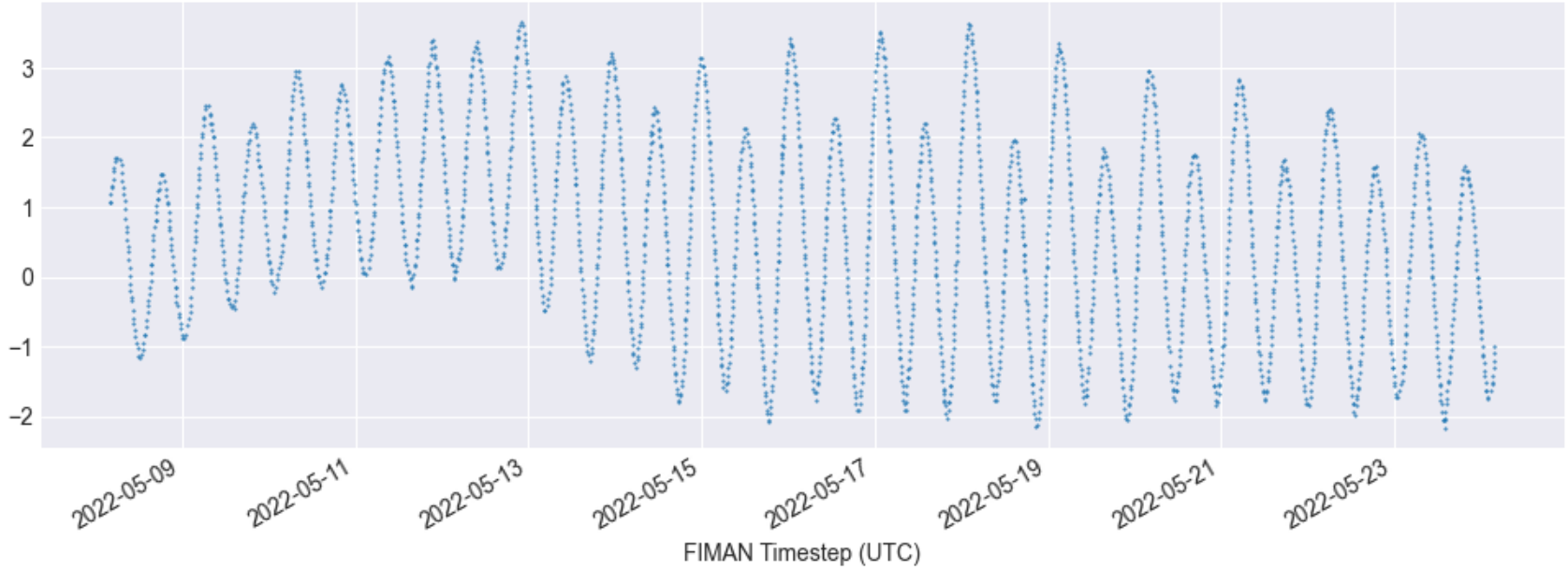
FIMAN Timestep (UTC)	Measurement Time (US/Eastern)	FIMAN Timestep (UTC)	FIMAN WL (ft NAVD88)	FIMAN WL (m NAVD88)
2022-05-24 03:55:03+00:00	2022-05-23 23:55:03-04:00	2022-05-24 03:55:03+00:00	-1.00	-0.304785
2022-05-24 03:47:09+00:00	2022-05-23 23:47:09-04:00	2022-05-24 03:47:09+00:00	-1.10	-0.335264
2022-05-24 03:36:10+00:00	2022-05-23 23:36:10-04:00	2022-05-24 03:36:10+00:00	-1.20	-0.365742
2022-05-24 03:29:08+00:00	2022-05-23 23:29:08-04:00	2022-05-24 03:29:08+00:00	-1.29	-0.393173
2022-05-24 03:20:02+00:00	2022-05-23 23:20:02-04:00	2022-05-24 03:20:02+00:00	-1.41	-0.429747
...	...	...	...	...
2022-05-08 04:36:02+00:00	2022-05-08 00:36:02-04:00	2022-05-08 04:36:02+00:00	1.31	0.399269
2022-05-08 04:22:08+00:00	2022-05-08 00:22:08-04:00	2022-05-08 04:22:08+00:00	1.29	0.393173
2022-05-08 04:14:03+00:00	2022-05-08 00:14:03-04:00	2022-05-08 04:14:03+00:00	1.18	0.359646
2022-05-08 04:06:09+00:00	2022-05-08 00:06:09-04:00	2022-05-08 04:06:09+00:00	1.08	0.329168
2022-05-08 04:05:07+00:00	2022-05-08 00:05:07-04:00	2022-05-08 04:05:07+00:00	1.06	0.323072

2604 rows × 4 columns

Clean FIMAN data by dropping rows with noise. Note that we only have one noisy data point in this record.

In [118…
```
FIMAN_df_clean = pd.concat([FIMAN_df[:'2022-05-18 17:15:01+00:00'],FIMAN_df['2022-05-18 17:13:07+00:00':]])
fig, ax = plt.subplots(figsize=(12, 4))

FIMAN_df_clean[f'FIMAN WL ({pltUnit} NAVD88)'].plot(ax = ax, linewidth = .5, ls = '', marker ='.', markersize = 1)
```

Out[118]:
```
<AxesSubplot:xlabel='FIMAN Timestep (UTC)'>
```



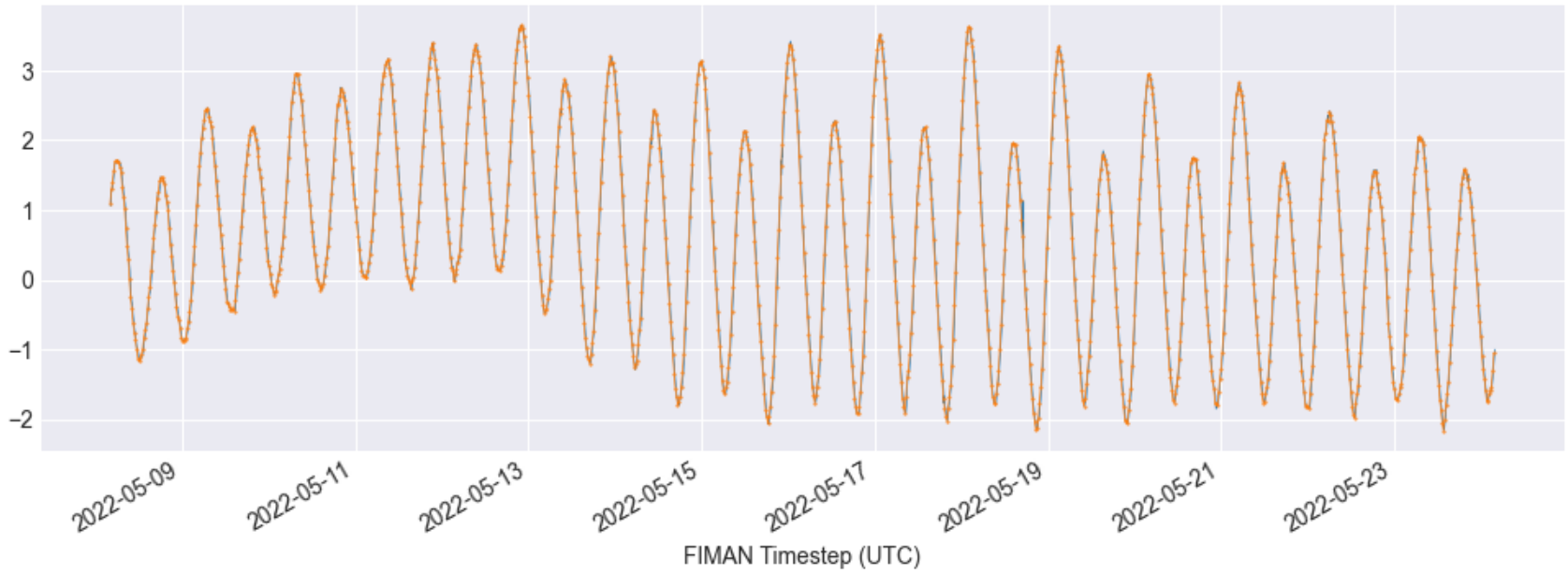
Smooth and resample FIMAN data. This step makes it easier to calculate RMSE in the next step.

In [119…
```
FIMAN_df_smooth = FIMAN_df_clean.resample('20T').mean()
FIMAN_df_smooth = FIMAN_df_smooth.interpolate(method='time')

fig, ax = plt.subplots(figsize=(12, 4))

FIMAN_df[f'FIMAN WL ({pltUnit} NAVD88)'].plot(ax = ax, linewidth = 0.5, ls = '-', marker ='', markersize = 1)
FIMAN_df_smooth[f'FIMAN WL ({pltUnit} NAVD88)'].plot(ax = ax, linewidth = .5, ls = '-', marker ='.', markersize = 1)
FIMAN_df_smooth
```

Out[119]:

FIMAN Timestep (UTC)	FIMAN WL (ft NAVD88)	FIMAN WL (m NAVD88)
2022-05-08 04:00:00+00:00	1.106667	0.337296
2022-05-08 04:20:00+00:00	1.300000	0.396221
2022-05-08 04:40:00+00:00	1.410000	0.429747
2022-05-08 05:00:00+00:00	1.576667	0.480545
2022-05-08 05:20:00+00:00	1.700000	0.518135
...	...	...
2022-05-24 02:20:00+00:00	-1.640000	-0.499848
2022-05-24 02:40:00+00:00	-1.582500	-0.482322
2022-05-24 03:00:00+00:00	-1.525000	-0.464797
2022-05-24 03:20:00+00:00	-1.300000	-0.396221
2022-05-24 03:40:00+00:00	-1.050000	-0.320024

1152 rows × 2 columns



Calculate RMSE of FIMAN data compared to model results

In [120…

```
diffFIMAN = m_df['ADCIRC WL (m NAVD88)'] - FIMAN_df_smooth['FIMAN WL (m NAVD88)']
diffFIMAN_df = pd.DataFrame(diffFIMAN)
diffFIMAN_df.columns = ['ADCIRC - FIMAN (m)']
diffFIMAN_df['(ADCIRC - FIMAN (m))^2'] = np.square(diffFIMAN_df['ADCIRC - FIMAN (m)'])

diffFIMAN_df = diffFIMAN_df.dropna()

diffFIMANsqaure_sum = diffFIMAN_df['(ADCIRC - FIMAN (m))^2'].sum()
n = diffFIMAN_df.shape[0]

rmseFIMAN_m = np.sqrt(diffFIMANsqaure_sum/n)
rmseFIMAN_ft = rmseFIMAN_m*3.281

print(f'RMSE (m) ADCIRC NC9rev2.1 vs FIMAN is: {rmseFIMAN_m}\nRMSE (ft) ADCIRC NC9rev2.1 vs FIMAN is: {rmseFIMAN_ft}')

diffFIMAN_df
```

```
RMSE (m) ADCIRC NC9rev2.1 vs FIMAN is: 0.10147979217250883
RMSE (ft) ADCIRC NC9rev2.1 vs FIMAN is: 0.33295519811800145
```

Out[120]:

	ADCIRC - FIMAN (m)	(ADCIRC - FIMAN (m))^2
2022-05-09 01:00:00+00:00	0.151266	0.022882
2022-05-09 02:00:00+00:00	0.087605	0.007675
2022-05-09 03:00:00+00:00	-0.011352	0.000129
2022-05-09 04:00:00+00:00	-0.123787	0.015323
2022-05-09 05:00:00+00:00	-0.173988	0.030272
...	...	...
2022-05-22 20:00:00+00:00	0.003372	0.000011
2022-05-22 21:00:00+00:00	0.018554	0.000344
2022-05-22 22:00:00+00:00	0.052576	0.002764
2022-05-22 23:00:00+00:00	0.022867	0.000523
2022-05-23 00:00:00+00:00	-0.001427	0.000002

336 rows × 2 columns

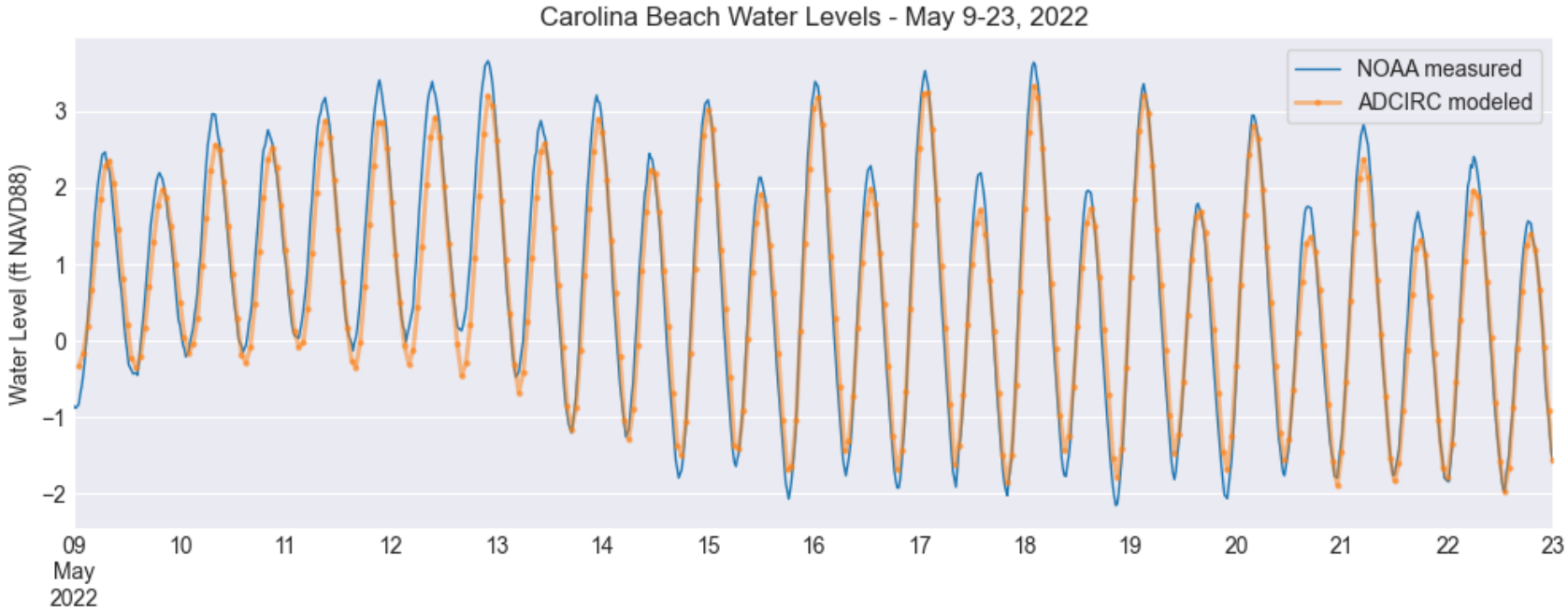
Plot Carolina Beach water level time series

In [134…

```
pltUnit = input("Plot in feet or meters (ft or m)? ")

""" Plot comparison lines on chart. """
fig, ax = plt.subplots(figsize=(12, 4))

#FIMAN_df[f'FIMAN WL ({pltUnit} NAVD88)'].plot(ax = ax, linewidth = .5, ls = '', marker ='.', markersize = 1)
FIMAN_df_smooth[f'FIMAN WL ({pltUnit} NAVD88)'].plot(ax = ax, linewidth = 1, ls = '-')

m_df[f'ADCIRC WL ({pltUnit} NAVD88)'].plot(ax = ax, linewidth = 2, ls = '-', marker = '.', markersize = 4, alpha = 0.5

ax.set_xlabel(' ')
ax.set_xlim([datetime(2022, 5, 9), datetime(2022, 5, 23)])
plt.style.use('seaborn-darkgrid')

ax.set_ylabel('Water Level ({} NAVD88)'.format(pltUnit))
ax.set_title('Carolina Beach Water Levels - May 9-23, 2022')
ax.legend(['NOAA measured', 'ADCIRC modeled'], frameon = True)
```

```
Plot in feet or meters (ft or m)? ft
```

Out[134]: <matplotlib.legend.Legend at 0x1e9df199b50>



How well do the predictions match to the measurements? What could be done to improve this comparison?

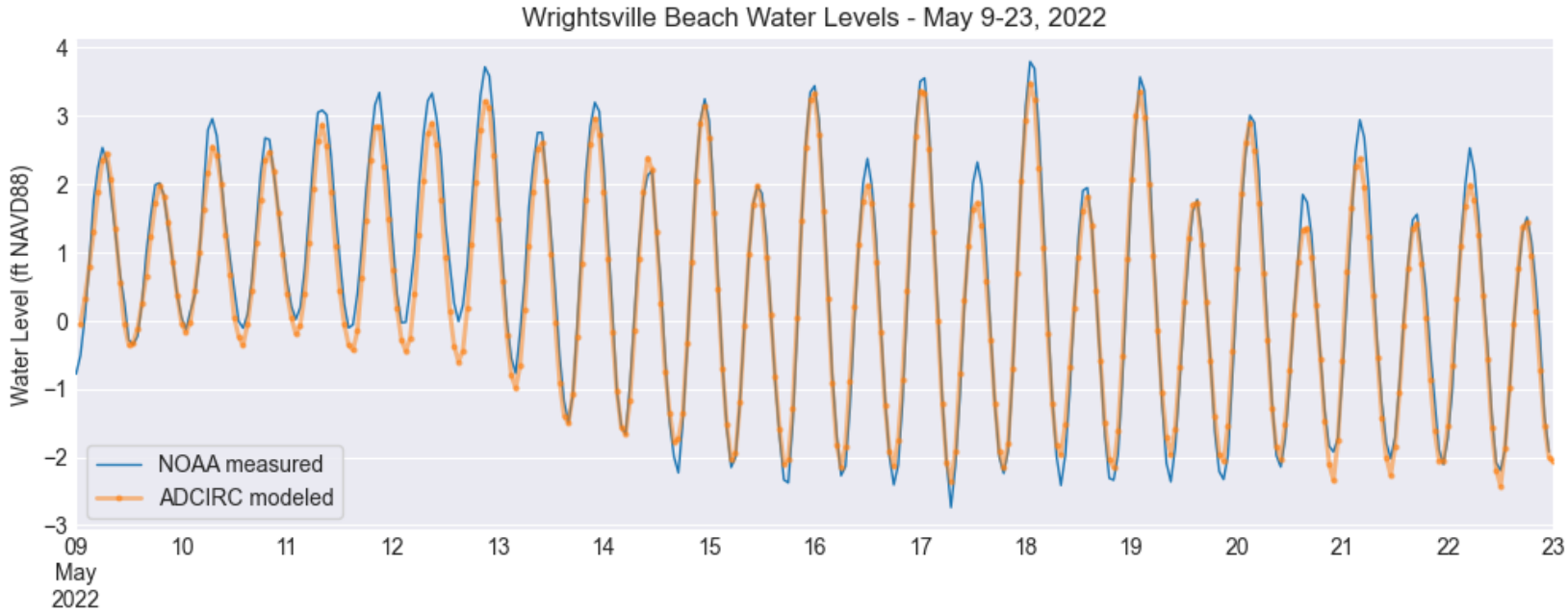
The three water level plots are included below for comparison. The RMSE between the Carolina Beach measurements and FIMAN data is slightly larger for the May 2022 simulations (0.33 ft) than for Nov 2021 simulations (0.28 ft). However, the model agreement with measured data is still in the ~0.3 ft neighborhood that is considered a good result for ADCIRC simulations.

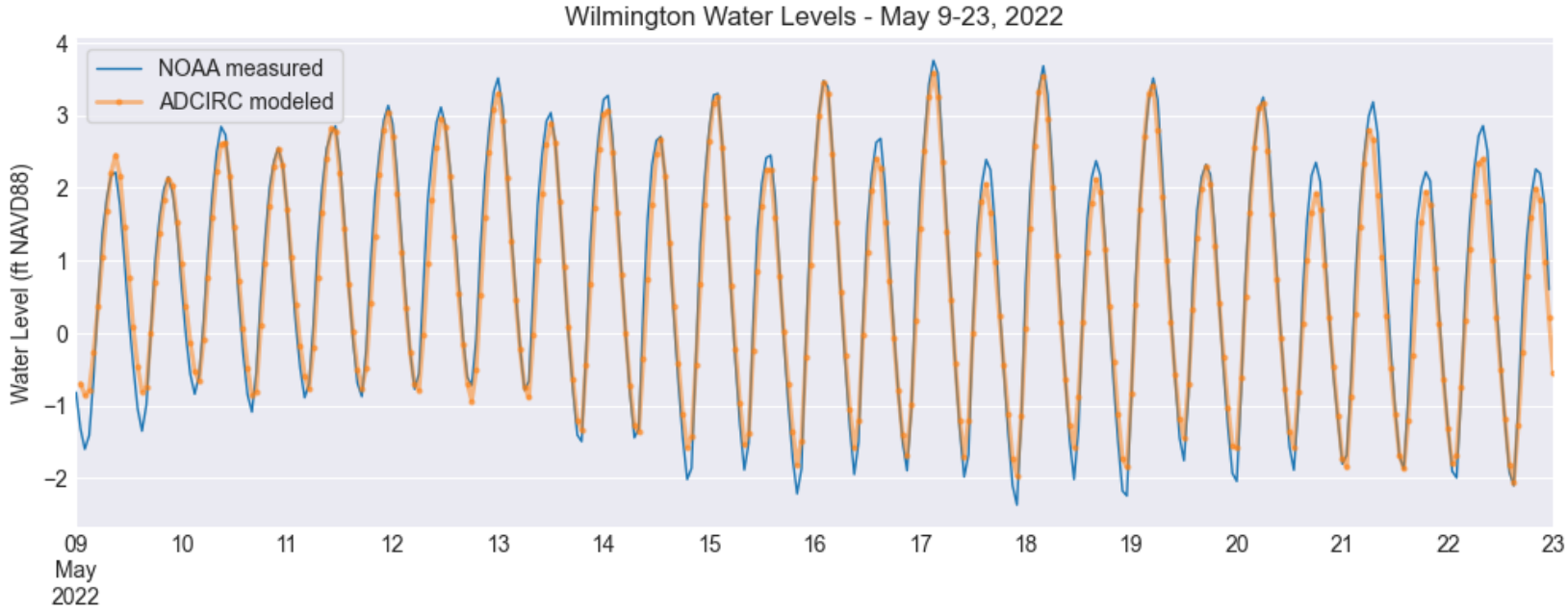
It is interesting to note that the model simulates the high water levels due to tides (May 15-20) more accurately than the high water levels due to winds (May 10-13. See wind data farther on in document). In Carolina Beach, the wind-driven high water levels lag the FIMAN data while the tide-driven high water levels show better agreement in time with the FIMAN tidal signal.

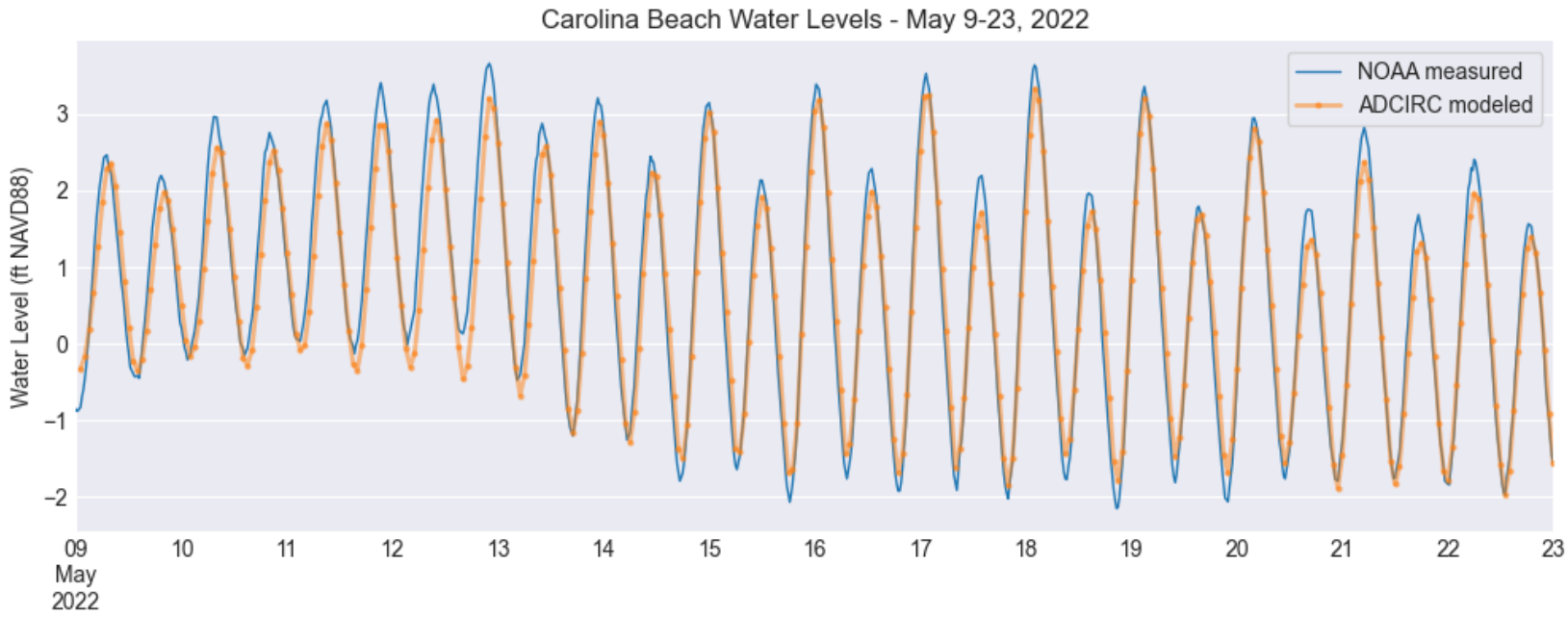
There are a couple ways this comparison might be able to be improved:

1. Use a better data-assimilated hindcast meteorlogical forcing
2. Add more tidal constituents to capture the missing ~0.2 feet of water level on tidal peaks

While we know from review of the NC9rev2.1 mesh that the mesh is not perfect in its representation of Carolina Beach, I suspect we have reached the point of diminishing returns on mesh edits. The disagreement between tide-driven high water levels (May 15-20) is similar between the open ocean gauge at Wrightsville Beach and the gauges at Carolina Beach / Wilmington. This leads to me believe that mesh misrepresentation of channel batymetries is not the biggest factor driving the disagreement between measured and modeled water levels.


Wrightsville Beach Water Levels - May 9-23, 2022
Water Level (ft NAVD88)
NOAA measured
ADCIRC modeled
4
3
2
1
0
−1
−2
−3
09
May
2022
10
11
12
13
14
15
16
17
18
19
20
21
22
23


Wilmington Water Levels - May 9-23, 2022
Water Level (ft NAVD88)
NOAA measured
ADCIRC modeled
4
3
2
1
0
−1
−2
09
May
2022
10
11
12
13
14
15
16
17
18
19
20
21
22
23



Question 3.2 and 3.3 - wind and water level comparisons

Import wave and water level data from NDBC database

Code based on script developed by Michael Gray

In [122…

```
### To see interactive html api request form, visit https://coastwatch.pfeg.noaa.gov/erddap/tabledap/cwwcNDBCMet.html
### set-up ERDDAP server
e = ERDDAP(
    server="https://coastwatch.pfeg.noaa.gov/erddap/",
    protocol="tabledap")

### select file type desired
e.response = "csv"
### dataset id (unique)
e.dataset_id = "cwwcNDBCMet"

### set constraints
```

```
e.constraints = {
    "time>=": '2022-05-09T00:00:00', # YYYY-MM-DDTHH-MM-SS must be string
    "time<=": '2022-05-23T00:00:00', # YYYY-MM-DDTHH-MM-SS must be string
    "latitude>=": 33.4, # degrees N
    "latitude<=": 34.3, # degrees N
    "longitude>=": -78.1, # degrees E (-180,180)
    "longitude<=": -76.8} # degrees E (-180,180)

### choose variables
'''
    Other possible variables:
        gst: Wind Gust Speed
        wvht: Wave Height
        dpd: Wave Period, Dominant
        apd: Wave Period, Average
        mwd: Wave Direction
        atmp: Air Temperature
        wtmp: Sea Surface Temperature
        dewp: Dewpoint Temperature
        vis: Station Visability
        ptdy: Pressure Tendency
        tide: Water Level
        wspu: Wind Speed, Zonal
        wspv: Wind Speed, Meridional
        bar: Atmospheric Pressure
'''
e.variables = [
    "time",
    "station",
    "latitude",
    "longitude",
    "wspd", # wind speed
    "wvht"]  # wave height

### load into pandas dataframe
data = e.to_pandas(index_col = 0)
### I prefer xarray datasets for easy plotting (cells below). Surprisingly, data --> pandas --> xarray is faster than
data = xr.Dataset.from_dataframe(data)
### rename variables for convenience. You can print the dataset to see the default names.
data = data.rename({
    'time (UTC)':'time',
    'latitude (degrees_north)':'lat',
    'longitude (degrees_east)':'lon',
```

```
    'wspd (m s-1)': 'wspd',
    'wvht (m)': 'wvht'})
### convert time for compatibility with xarray internal plotting
data['time'] = data.time.astype('datetime64')
```

In [123…

```
### show Dataset
data
```

Out[123]: xarray.Dataset

► Dimensions: (**time**: 18474)

▼ Coordinates:

time	(time)	datetime64[ns]	2022-05-09 … 2022-05-23

▼ Data variables:

station	(time)	object	'41013' '41013' … 'WLON7' 'WLON7'
lat	(time)	float64	33.44 33.44 33.44 … 34.23 34.23
lon	(time)	float64	-77.74 -77.74 … -77.95 -77.95
wspd	(time)	float64	11.4 11.9 11.8 12.0 … nan nan nan
wvht	(time)	float64	nan nan nan nan … nan nan nan nan

► Indexes: (1)

► Attributes: (0)

In [124…

```
### collect station id's
stations = np.unique(data.station)
print('Total number of stations: '+str(len(stations)))
stations_wvht = np.unique(data.station.loc[~np.isnan(data.wvht)])
print('Total number of stations with wave height data: '+str(len(stations_wvht)))
stations_wspd = np.unique(data.station.loc[~np.isnan(data.wspd)])
print('Total number of stations with wind speed data: '+str(len(stations_wspd)))
```

```
Total number of stations: 11
Total number of stations with wave height data: 4
Total number of stations with wind speed data: 7
```

In [125…

```
### EXAMPLE- Station locations
lats = data.lat # get lat dataarray
```

```
lons = data.lon # get lon dataarray

station_locations = np.zeros((len(stations),2))
for i in range(len(stations)): # loop through stations
    lat_station = lats.loc[data.station == stations[i]].values[0] # select lat of current station (stationary buoys so
    lon_station = lons.loc[data.station == stations[i]].values[0] # select lon of current station (stationary buoys so

    station_locations[i,0] = lat_station # store lat
    station_locations[i,1] = lon_station # store lon

#########################################################################
#### Comment out this section if you don't want cartopy (geo-plots) #####

# fig,ax = plt.subplots(1,1, subplot_kw={'projection': ccrs.PlateCarree()}) # create plot with projection
# gl = ax.gridlines(draw_labels=True, dms=True, x_inline=False, y_inline=False) # show coordinates and grig
# gl.top_labels = False
# gl.right_labels = False
# ax.add_feature(cfeature.LAND)#### Add land ######
# ax.add_feature(cfeature.COASTLINE,lw=0.25)##### Add Coastline #########
# ax.add_feature(cfeature.RIVERS,lw=0.25)##### Add river ######
# ax.add_feature(cfeature.LAKES)###### Add lake #####
# ax.scatter(station_locations[:,1], station_locations[:,0], color = 'k', transform = ccrs.PlateCarree()) # plot lon,
#########################################################################
### Uncomment below line if no cartopy
plt.scatter(station_locations[:,1], station_locations[:,0]) # plot lon, lat
plt.title('Location of buoys in defined region')
plt.show()
```

Question 3.2 - wind speeds

Compare the ADCIRC results to the measured wind speeds at your NDBC buoys

In [126…

```
### Select all windspeed data from stations with wind data
#41013: Frying Pan Shoals
station1_wspd = data.wspd.loc[data.station == stations_wspd[0]]

#41037: Wrightsville Beach Offshore
station2_wspd = data.wspd.loc[data.station == stations_wspd[1]]

#41037: Wrightsville Beach Nearshore
station3_wspd = data.wspd.loc[data.station == stations_wspd[2]]

#41064: Onslow Bay Outer
```

```
station4_wspd = data.wspd.loc[data.station == stations_wspd[3]]

#JMPN7: Johnny Mercer Pier, Wrightsville Beach, NC
station5_wspd = data.wspd.loc[data.station == stations_wspd[4]]

#MBIN7: Masonboro Island South
station6_wspd = data.wspd.loc[data.station == stations_wspd[5]]

#MBNN7: Masonboro Island North
station7_wspd = data.wspd.loc[data.station == stations_wspd[6]]

sta_wind_dict = {
    '41013': 'Frying Pan Shoals',
    '41037': 'Wrightsville Beach Offshore',
    '41038': 'Wrightsville Beach Nearshore',
    '41064': 'Onslow Bay Outer',
    'JMPN7': 'Wrightsville Pier',
    'MBIN7': 'Masonboro Island South',
    'MBNN7': 'Masonboro Island North'
}

stations_wspd
```

Out[126]:
```
array(['41013', '41037', '41038', '41064', 'JMPN7', 'MBIN7', 'MBNN7'],
      dtype=object)
```

Import wind data from text files of time series interpolated from the ADCIRC fort.74.nc output and convert x and y wind speed components to wind speed scalar

In [135…
```
adcWinds = {}

for station in stations_wspd:

    windx = f"C:/Users/ththelen/OneDrive - North Carolina State University/CarolinaBeach/SMS/20221103_NC9rev2.1+0.15m-
    wildpath_x = glob.glob(windx)
    windy = f"C:/Users/ththelen/OneDrive - North Carolina State University/CarolinaBeach/SMS/20221103_NC9rev2.1+0.15m-
    wildpath_y = glob.glob(windy)

    windx_df = pd.read_csv(wildpath_x[0], sep=' ', header=None)
    windy_df = pd.read_csv(wildpath_y[0],sep=' ', header=None)
    windx_df.columns = ['ADCRIC Timestep (hr)', 'windx']
    windy_df.columns = ['ADCRIC Timestep (hr)', 'windy']
```

```
    windx_df['windy'] = windy_df['windy']
    windx_df['windtot'] = (windx_df['windy']**2 + windx_df['windx']**2)**0.5
    windx_df['Time Delta (hrs)'] = pd.to_timedelta(windx_df['ADCRIC Timestep (hr)'], unit='h')
    windx_df['ADCIRC Timestep (UTC)'] = dt_start_utc + windx_df['Time Delta (hrs)']
    #windx_df['ADCIRC Timestep (UTC)'] = windx_df['ADCIRC Timestep (UTC)'].dt.tz_localize('UTC')
    windx_df.index = windx_df['ADCIRC Timestep (UTC)']
    adcWinds[station] = windx_df

#adcWinds
```

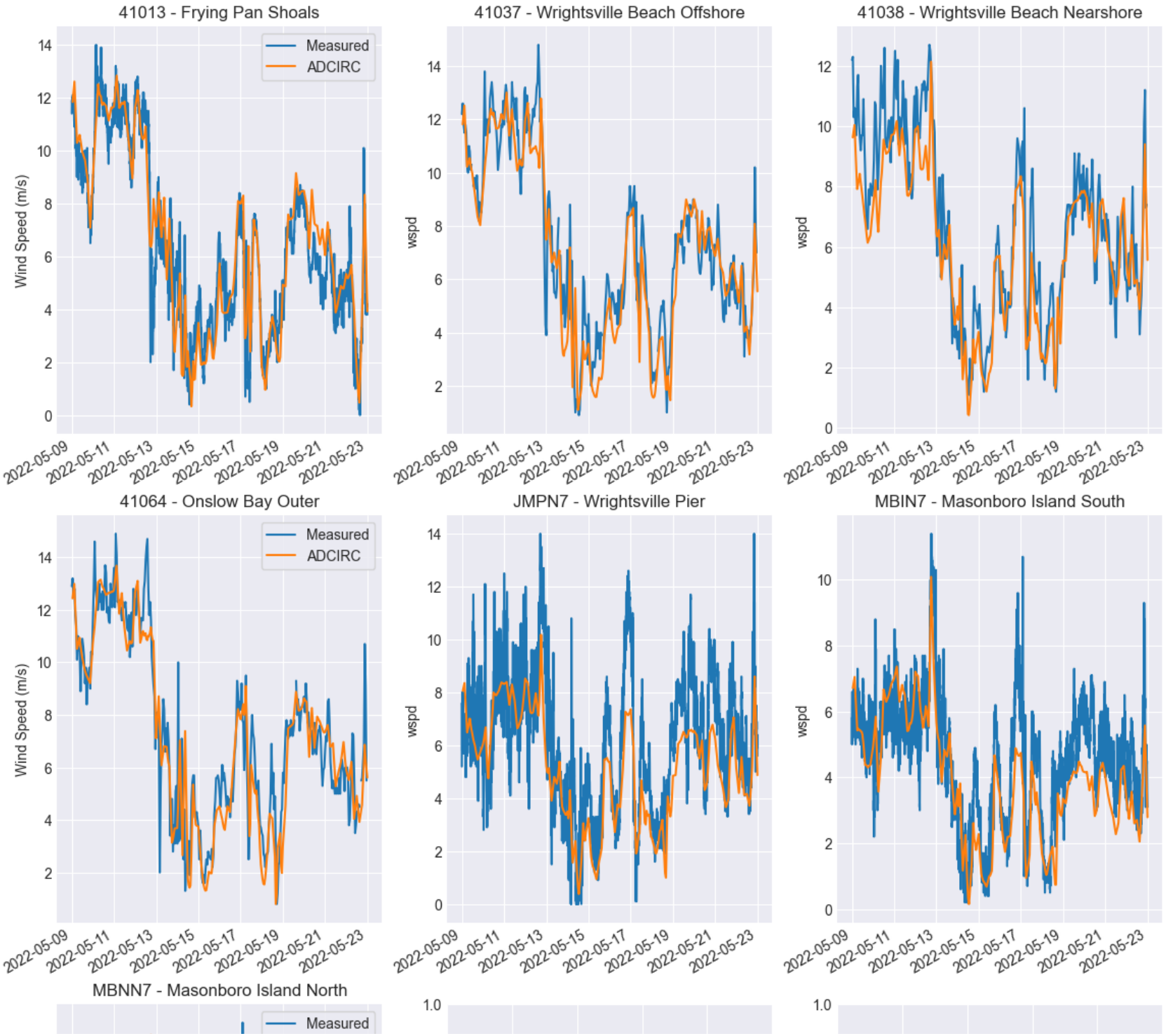
Plot measured vs modeled wind speed data at all seven stations from the NBDC database

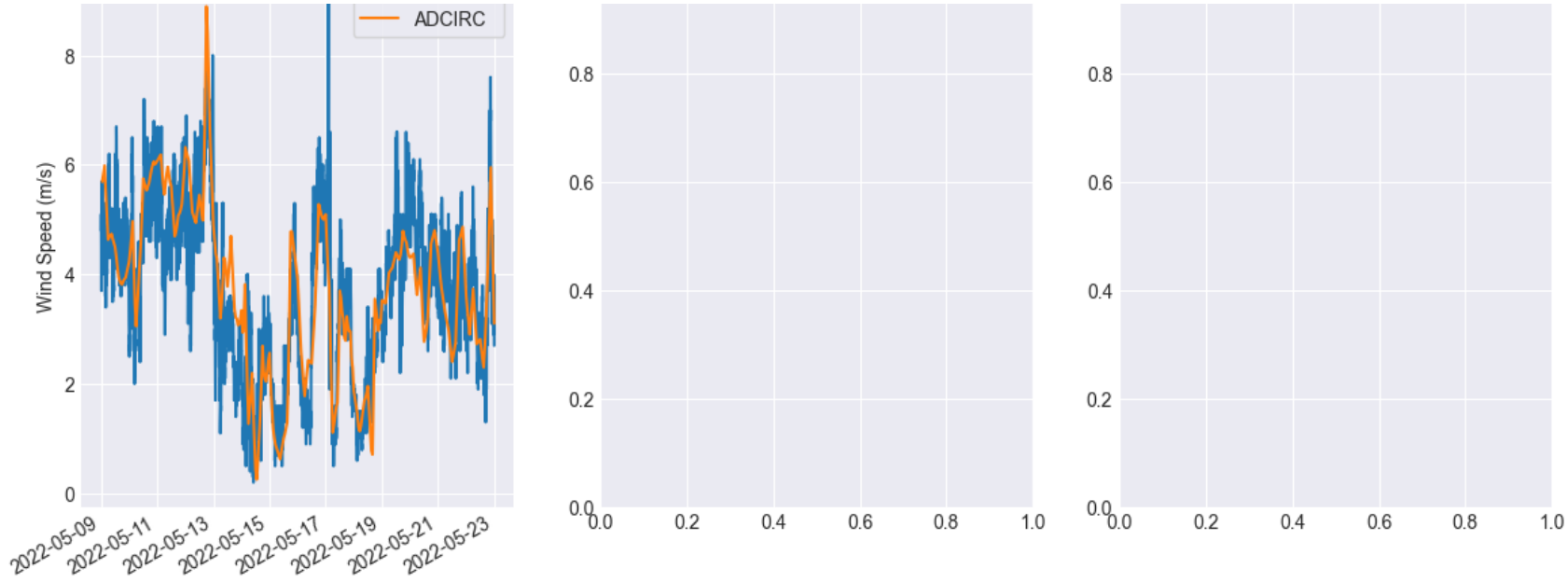
In [128…

```
fig, axs= plt.subplots(nrows = 3, ncols = 3, figsize=(14, 20))

### use xarray's internal plotting for quick visualization
station1_wspd.plot(ax = axs[0,0])
axs[0,0].set_title(f'{stations_wspd[0]} - {sta_wind_dict[stations_wspd[0]]}')
adcWinds[stations_wspd[0]]['windtot'].plot(ax = axs[0,0])
axs[0,0].set_xlabel(' ')
axs[0,0].set_ylabel('Wind Speed (m/s)')
axs[0,0].legend(['Measured', 'ADCIRC'], frameon = True)

station2_wspd.plot(ax = axs[0,1])
axs[0,1].set_title(f'{stations_wspd[1]} - {sta_wind_dict[stations_wspd[1]]}')
adcWinds[stations_wspd[1]]['windtot'].plot(ax = axs[0,1])
axs[0,1].set_xlabel(' ')

station3_wspd.plot(ax = axs[0,2])
axs[0,2].set_title(f'{stations_wspd[2]} - {sta_wind_dict[stations_wspd[2]]}')
adcWinds[stations_wspd[2]]['windtot'].plot(ax = axs[0,2])
axs[0,2].set_xlabel(' ')

station4_wspd.plot(ax = axs[1,0])
axs[1,0].set_title(f'{stations_wspd[3]} - {sta_wind_dict[stations_wspd[3]]}')
adcWinds[stations_wspd[3]]['windtot'].plot(ax = axs[1,0])
axs[1,0].set_xlabel(' ')
axs[1,0].set_ylabel('Wind Speed (m/s)')
axs[1,0].legend(['Measured', 'ADCIRC'], frameon = True)

station5_wspd.plot(ax = axs[1,1])
axs[1,1].set_title(f'{stations_wspd[4]} - {sta_wind_dict[stations_wspd[4]]}')
adcWinds[stations_wspd[4]]['windtot'].plot(ax = axs[1,1])
```

```
axs[1,1].set_xlabel(' ')

station6_wspd.plot(ax = axs[1,2])
axs[1,2].set_title(f'{stations_wspd[5]} - {sta_wind_dict[stations_wspd[5]]}')
adcWinds[stations_wspd[5]]['windtot'].plot(ax = axs[1,2])
axs[1,2].set_xlabel(' ')

station7_wspd.plot(ax = axs[2,0])
axs[2,0].set_title(f'{stations_wspd[6]} - {sta_wind_dict[stations_wspd[6]]}')
adcWinds[stations_wspd[6]]['windtot'].plot(ax = axs[2,0])
axs[2,0].set_xlabel(' ')
axs[2,0].set_ylabel('Wind Speed (m/s)')
axs[2,0].legend(['Measured', 'ADCIRC'], frameon = True)

plt.xlabel(None)
```

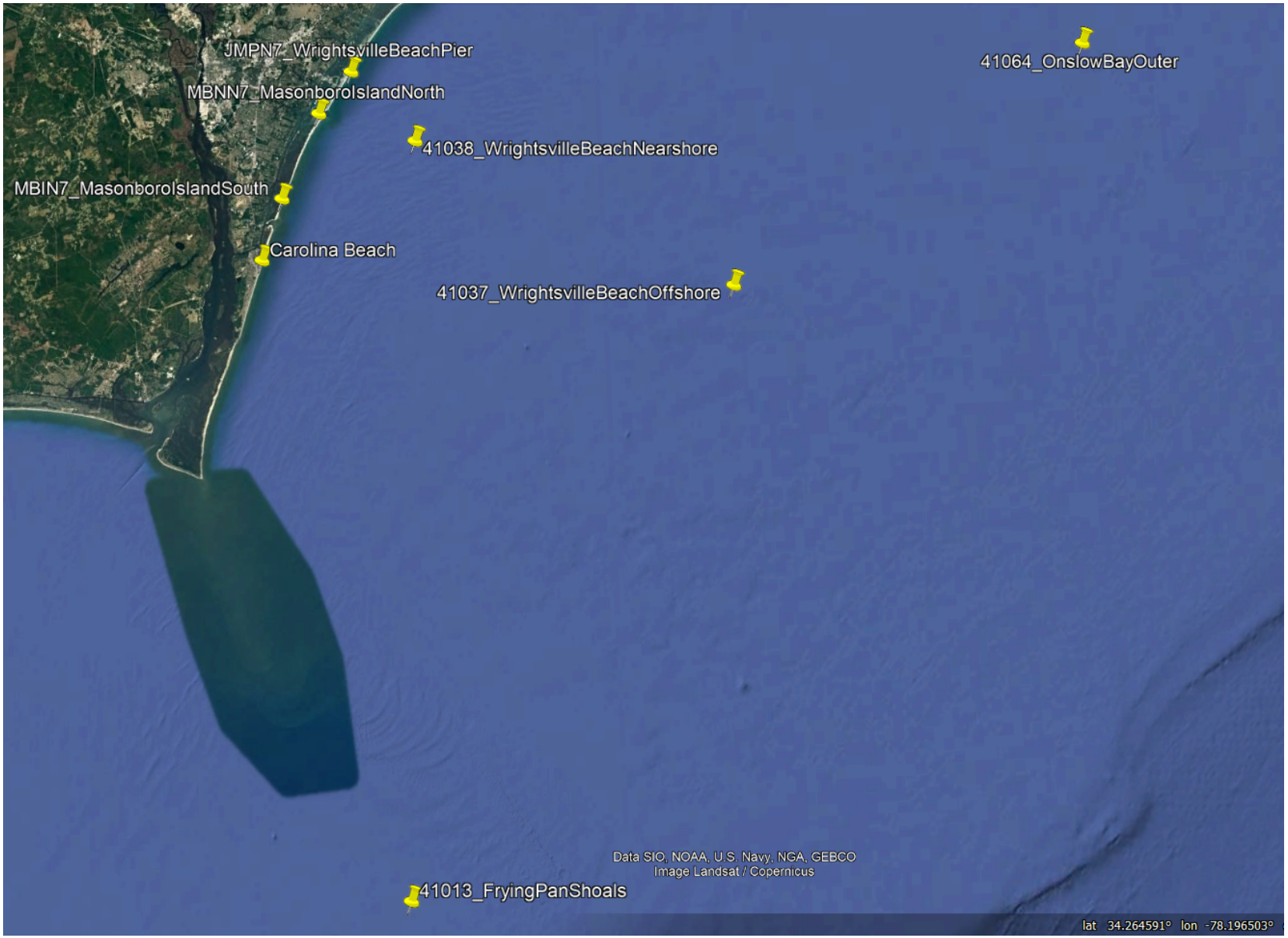
Out[128]: Text(0.5, 0, '')


41013 - Frying Pan Shoals
41037 - Wrightsville Beach Offshore
41038 - Wrightsville Beach Nearshore
41064 - Onslow Bay Outer
JMPN7 - Wrightsville Pier
MBIN7 - Masonboro Island South
MBNN7 - Masonboro Island North
Measured
ADCIRC
Wind Speed (m/s)
wspd
2022-05-09
2022-05-11
2022-05-13
2022-05-15
2022-05-17
2022-05-19
2022-05-21
2022-05-23



How well do wind speed predictions match to the measurements? What are the implications for the accuracy of the wave and surge models?

Model winds speeds generally show good agreement with measured values, although agreement appears to be slightly better at offshore stations compared to nearshore and coastal stations. This indicates that we could expect similarly good agreement for wave and surge model results. Wind measurement locations are shown in the figure below.



Question 3.3 - wave heights

Compare the ADCIRC results to the measured significant waves heights at your NDBC buoys

In [129…

```
# 41013: Frying Pan Shoals
station1_wvht = data.wvht.loc[data.station == stations_wvht[0]]

# 41108: Wilmington Harbor
station2_wvht = data.wvht.loc[data.station == stations_wvht[1]]

# 41110: Masonboro Inlet
station3_wvht = data.wvht.loc[data.station == stations_wvht[2]]

# 41159: Onslow Bay Outer
station4_wvht = data.wvht.loc[data.station == stations_wvht[3]]

sta_wave_dict = {
     '41013': 'Frying Pan Shoals',
     '41108': 'Wilmington Harbor',
     '41110': 'Masonboro Inlet',
     '41159': 'Onslow Bay Outer'
}

station1_wvht
```

Out[129]: xarray.DataArray 'wvht' (**time**: 2016)

array([nan, nan, nan, ..., 0.84, nan, nan])

▼ Coordinates:

time	(time)	datetime64[ns]	2022-05-09 ... 2022-05-23

► Indexes: (1)

► Attributes: (0)

Import wave data from text files of time series interpolated from the ADCIRC swan_HS.63.nc

In [136…

```
adcWaves = {}

for stations in stations_wvht:

    wvpath = f"C:/Users/ththelen/OneDrive - North Carolina State University/CarolinaBeach/SMS/20221103_NC9rev2.1+0.15m
    wildpath_wave = glob.glob(wvpath)
```

```
    wave_df = pd.read_csv(wildpath_wave[0], sep=' ', header=None)
    wave_df.columns = ['ADCRIC Timestep (hr)', 'wvht']
    wave_df['Time Delta (hrs)'] = pd.to_timedelta(wave_df['ADCRIC Timestep (hr)'], unit='h')
    wave_df['ADCIRC Timestep (UTC)'] = dt_start_utc + wave_df['Time Delta (hrs)']
    #windx_df['ADCIRC Timestep (UTC)'] = windx_df['ADCIRC Timestep (UTC)'].dt.tz_localize('UTC')
    wave_df.index = wave_df['ADCIRC Timestep (UTC)']
    adcWaves[stations] = wave_df

#adcWaves
```

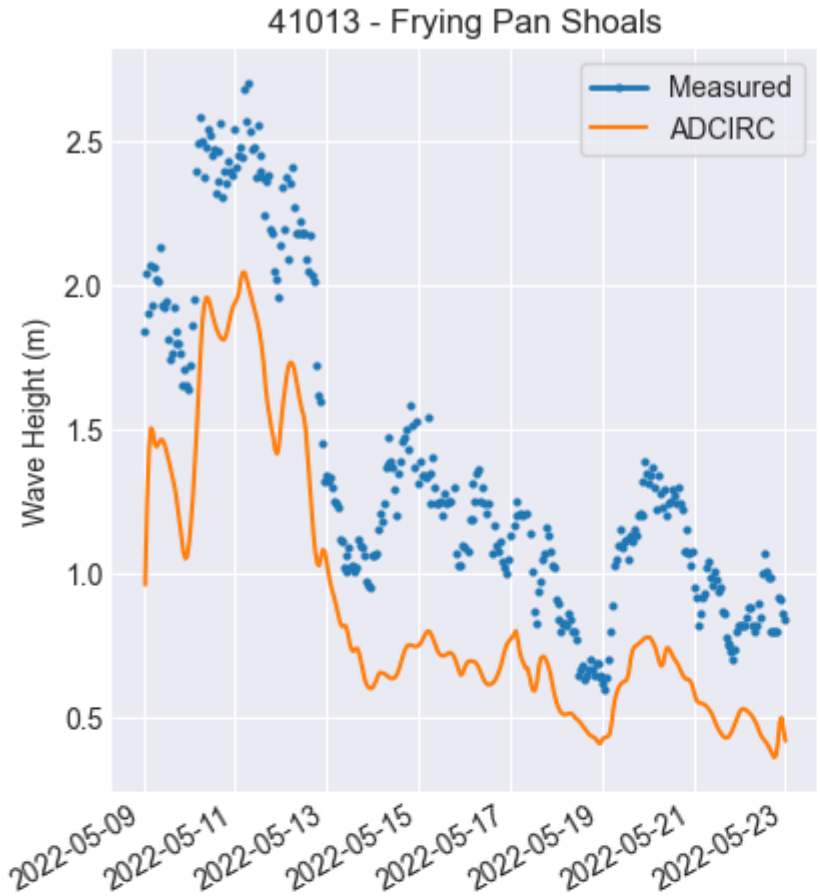
Plot measured vs modeled wave heights at all four stations from the NBDC database. Note that the Frying Pan Shoals buoy has wave data with a strange nan format that prevents display as a line plot. Data is shown as a scatter plot instead.

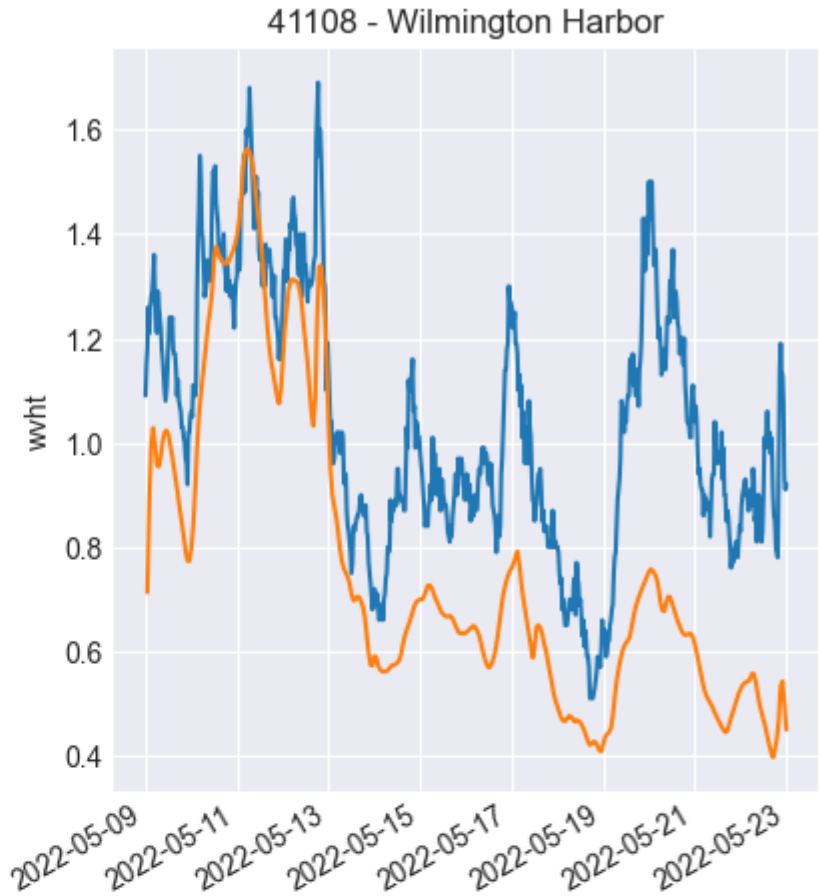
In [131…

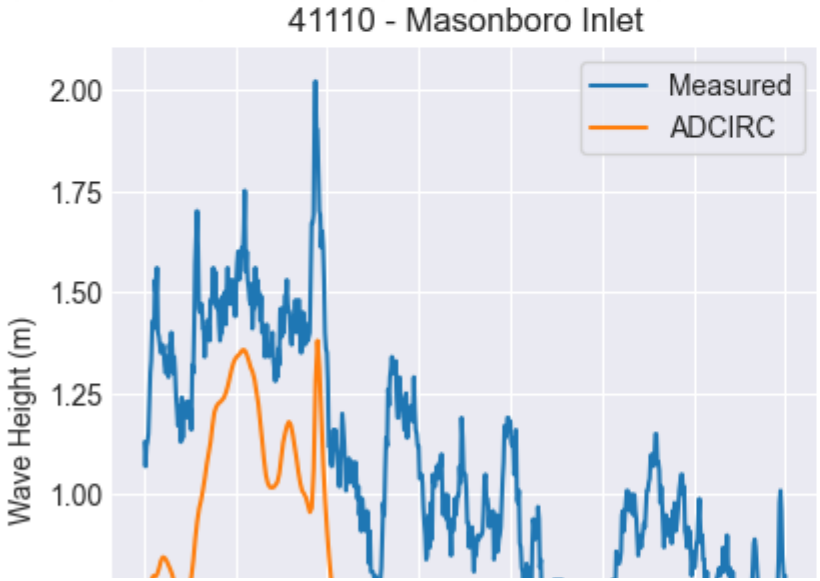
```
fig, axs= plt.subplots(nrows = 2, ncols = 2, figsize=(10, 12))

### use xarray's internal plotting for quick visualization
station1_wvht.plot(ax = axs[0,0], linewidth = 2, ls = '-', marker = '.', markersize = 4)
axs[0,0].set_title(f'{stations_wvht[0]} - {sta_wave_dict[stations_wvht[0]]}')
adcWaves[stations_wvht[0]]['wvht'].plot(ax = axs[0,0])
axs[0,0].set_xlabel(' ')
axs[0,0].set_ylabel('Wave Height (m)')
axs[0,0].legend(['Measured', 'ADCIRC'], frameon = True)

station2_wvht.plot(ax = axs[0,1])
axs[0,1].set_title(f'{stations_wvht[1]} - {sta_wave_dict[stations_wvht[1]]}')
adcWaves[stations_wvht[1]]['wvht'].plot(ax = axs[0,1])
axs[0,1].set_xlabel(' ')

station3_wvht.plot(ax = axs[1,0])
axs[1,0].set_title(f'{stations_wvht[2]} - {sta_wave_dict[stations_wvht[2]]}')
adcWaves[stations_wvht[2]]['wvht'].plot(ax = axs[1,0])
axs[1,0].set_xlabel(' ')
axs[1,0].set_ylabel('Wave Height (m)')
axs[1,0].legend(['Measured', 'ADCIRC'], frameon = True)

station4_wvht.plot(ax = axs[1,1])
axs[1,1].set_title(f'{stations_wvht[3]} - {sta_wave_dict[stations_wvht[3]]}')
adcWaves[stations_wvht[3]]['wvht'].plot(ax = axs[1,1])
axs[1,1].set_xlabel(' ')
```

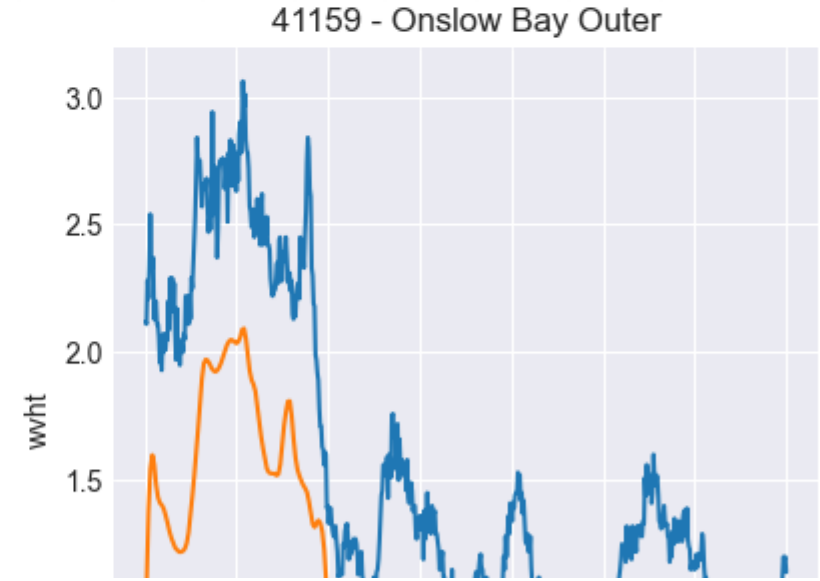
Out[131]:

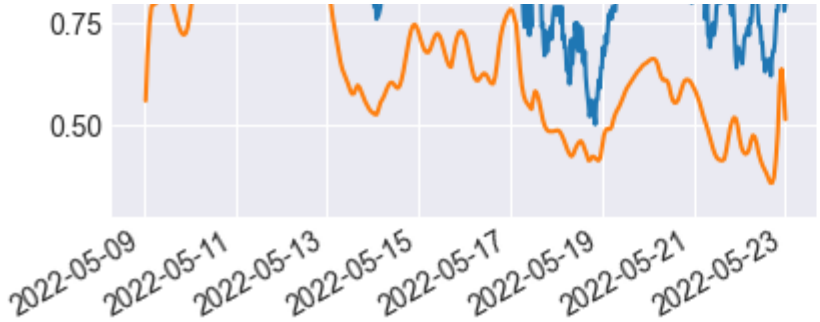
```
Text(0.5, 0, ' ')
```

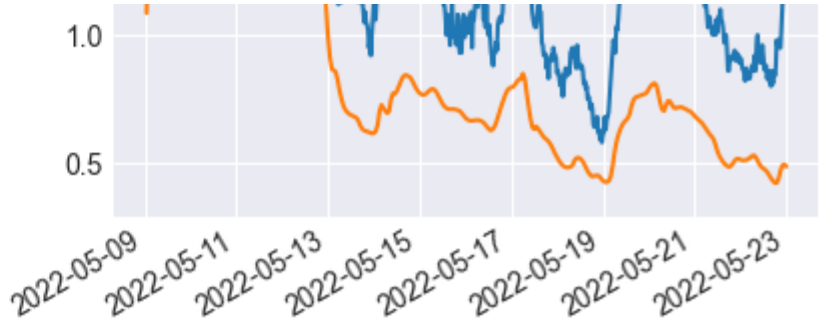

41013 - Frying Pan Shoals
Measured
ADCIRC
Wave Height (m)
2.5
2.0
1.5
1.0
0.5
2022-05-09
2022-05-11
2022-05-13
2022-05-15
2022-05-17
2022-05-19
2022-05-21
2022-05-23


41108 - Wilmington Harbor
wvht
1.6
1.4
1.2
1.0
0.8
0.6
0.4
2022-05-09
2022-05-11
2022-05-13
2022-05-15
2022-05-17
2022-05-19
2022-05-21
2022-05-23


41110 - Masonboro Inlet
Measured
ADCIRC
Wave Height (m)
2.00
1.75
1.50
1.25
1.00

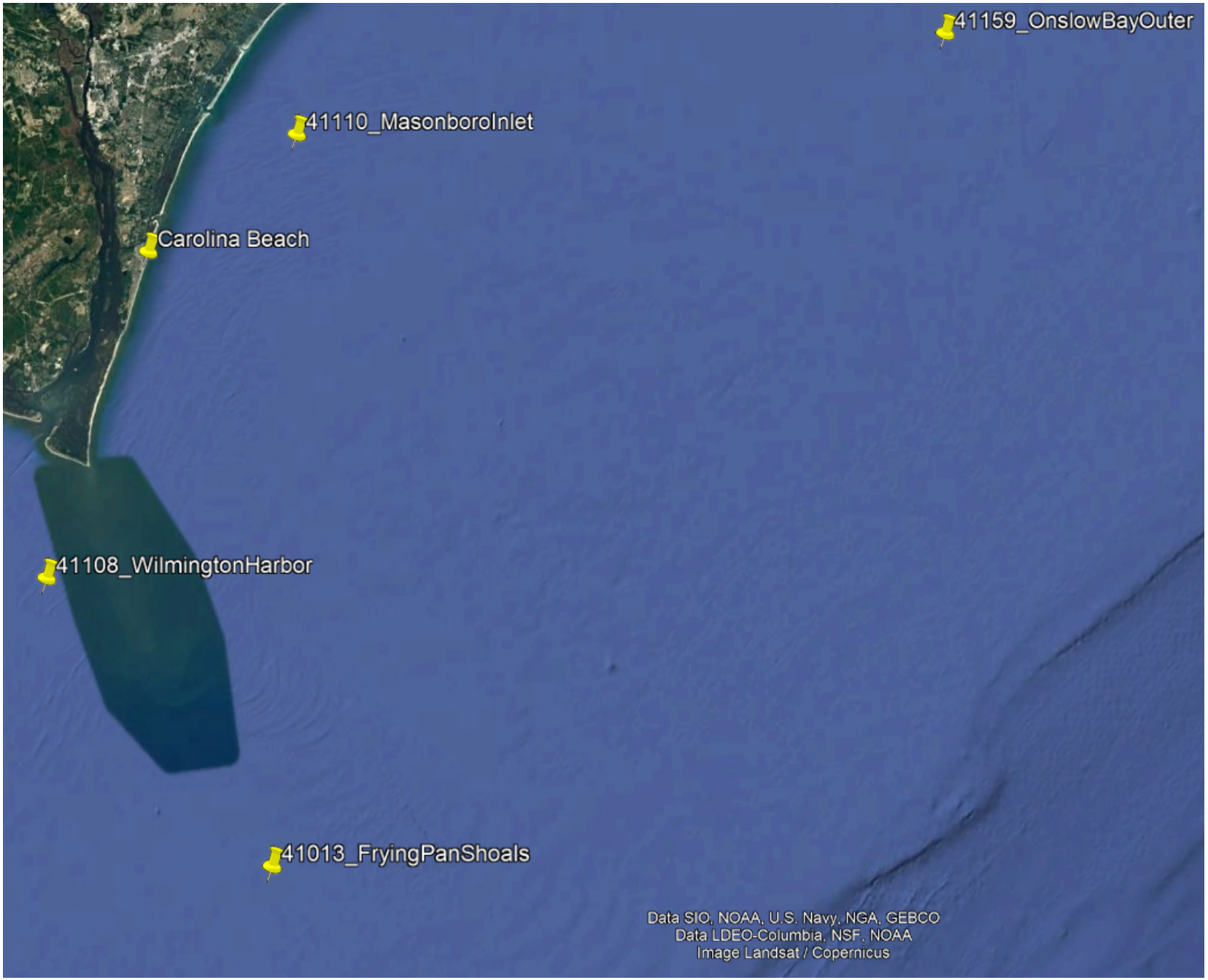

41159 - Onslow Bay Outer
wvht
3.0
2.5
2.0
1.5





Compare the ADCIRC results to the measured wave heights at your NDBC buoys. How well do the predictions match to the measurements? Are the predictions better nearshore or offshore?

Model significant wave heights consistently underpredict measured waves height by 0.5 to 1 m at both nearshore and offshore locations. While I do not know exactly what level of measured vs. modeled agreement is considered acceptable for wave height, this is somewhat suprisingly because wind speed predictions show a better normalized agreement with measured value than wave heights do. The figure below shows the location of wave gauges.


41159_OnslowBayOuter
41110_MasonboroInlet
Carolina Beach
41108_WilmingtonHarbor
41013_FryingPanShoals
Data SIO, NOAA, U.S. Navy, NGA, GEBCO
Data LDEO-Columbia, NSF, NOAA
Image Landsat / Copernicus

Question 4

Based on the results from all of your validation, what is your opinion of your simulated event? If you could improve one aspect of your simulation, then what would it be?

The positive takeaways from this simulation are accurate simulation of tide-driven water levels and wind speeds. Areas for improvement are modeled wave heights and wind-driven high water levels.

Appendix

Finding FIMAN outliers that need to be removed

In [132…
```
FIMAN_df.loc['2022-05-18 17:00:11+00:00':'2022-05-18 17:59:11+00:00']
```

```
C:\Users\ththelen\AppData\Local\Temp\ipykernel_19792\1462677667.py:1: FutureWarning: Value based partial slicing on n
on-monotonic DatetimeIndexes with non-existing keys is deprecated and will raise a KeyError in a future Version.
  FIMAN_df.loc['2022-05-18 17:00:11+00:00':'2022-05-18 17:59:11+00:00']
```

Out[132]:

FIMAN Timestep (UTC)	Measurement Time (US/Eastern)	FIMAN Timestep (UTC)	FIMAN WL (ft NAVD88)	FIMAN WL (m NAVD88)
2022-05-18 17:55:03+00:00	2022-05-18 13:55:03-04:00	2022-05-18 17:55:03+00:00	-0.23	-0.070101
2022-05-18 17:49:03+00:00	2022-05-18 13:49:03-04:00	2022-05-18 17:49:03+00:00	-0.13	-0.039622
2022-05-18 17:43:01+00:00	2022-05-18 13:43:01-04:00	2022-05-18 17:43:01+00:00	-0.02	-0.006096
2022-05-18 17:36:09+00:00	2022-05-18 13:36:09-04:00	2022-05-18 17:36:09+00:00	0.08	0.024383
2022-05-18 17:34:05+00:00	2022-05-18 13:34:05-04:00	2022-05-18 17:34:05+00:00	0.11	0.033526
2022-05-18 17:28:05+00:00	2022-05-18 13:28:05-04:00	2022-05-18 17:28:05+00:00	0.22	0.067053
2022-05-18 17:15:01+00:00	2022-05-18 13:15:01-04:00	2022-05-18 17:15:01+00:00	0.32	0.097531
2022-05-18 17:14:09+00:00	2022-05-18 13:14:09-04:00	2022-05-18 17:14:09+00:00	1.13	0.344407
2022-05-18 17:13:07+00:00	2022-05-18 13:13:07-04:00	2022-05-18 17:13:07+00:00	0.46	0.140201
2022-05-18 17:06:05+00:00	2022-05-18 13:06:05-04:00	2022-05-18 17:06:05+00:00	0.57	0.173728
2022-05-18 17:02:07+00:00	2022-05-18 13:02:07-04:00	2022-05-18 17:02:07+00:00	0.64	0.195062