Implementation

June 5, 2020

This section presents the methods and tools needed for visualizing and calculating the flow field of a selected area of the globe. A presentation of how to classify route from AIS-data will also be done. A rundown of the program will be done together with the dependencies needed to run the program. The section is divided into a route creation using AIS-data, a map creation section and

First, the applied method was inspired and done by a self-study reading through the literature. The motivation and idea was brought to light with an issue of the analytic potential solutions not being able to represent the map accurately. In addition, the analytic solution does not take into account the geometry of land of arbitrary shapes which in this case is crucial. This approach resulted in a guessing game where it was attempted to represent the flow field by placing different potential in the different in order to fit the map profile. A new approach was needed, a numerical one.

The source panel method seemed as an appropriate and more applicable approach. The idea here is to produce flow-fields given information about the map as input and calculate and thus solve for the velocity components needed to visualize a flow-field. There was not found any research regarding using real maps in the literature. The closest research was from Thor, I. Fossen et al(refrefref) where an artifical map was produced.

The reason for this was that the solution relied on objects places in the area and thereafter finding the flow-field. Also, the source panel method approximates objects with straight lines (panels). This was also an important property because if it was possible to separate and set a constraint on the boundaries of land and sea then the boundary conditions and panels could then represent a coastlines on the selected area. The method would now need the coastlines of the selected map. This was one of reasons to choose Basemap as a map creator as their is a built-in attribute of the Basemap class instance returning the coastlines as line segments. For more details see below.

In order to study the data and the environment around, a visualisation tool is necessary. This grants insight into how the vessel traverse and how much data is necessary for extracting wanted routes in area of interest. This will be done with a map present. In addition, the flow fields need to be plotted in order to understand it's motion. This section presents a framework in how to create a flow field using the source panel method applied for a real life scenario. From the works of Thor Fossen ref!!! the method showed promising results for vessels. Thereafter, a map can be created showing the most interesting areas and in addition saving the computer from unnecessary and excessive calculations. This is done with a package from the matplotlib toolkit called Basemap (explained in more detail below).

With the map created with boundaries the next step will be to plot up some of the AIS-data. Reading the data into a python program is done with pandas.

Dependencies

The main program requires some packages: Pandas and Basemap. Furthermore, the main program calls additional functions which also need to be present in the folder in order to run properly. These functions calculate the source strengths (sourcepanel.py) and the potential at each grid cell (sourcepoint.py).

Creating routes with Pandas

Pandas is a library were it's sole purpose is used for data analysis. In more detail, a Dataframe type object (df) is created which let's the user manipulate a dataset by indexing, slicing, subsetting, merging, accessing rows and columns to only name a few operations.

The first step is to take a look at the data. As mentioned above (AIS-data Extracting routes) the timestamp column can be used to sample routes from the data set. The data set has dimension 71694 X 12. Each column represent a variable for specific timestamp (see figure of AIS-data example plot above, ref image!!). Working with all that data becomes complex and computational heavy. To get a clearer view, a subset of the data is drawn out. Here we it is based on the vessel type. We create a new df consisting only of one single vessel type:

```python
df_subset = df[df['ShipType'].isin(['tugboats'])].copy()
```

Listing 1: All tugboat type vessels are stored in a new variable which can further be analyzed

The subset now consists of only one vessel type (in this case tugboats) from the main data set. The objective now is to extract routes from the data subset. From the timestamp column it can be seen that on the same day the data contains a fixed interval of logged AIS-data varying from seconds to minutes. One could argue that this is the same vessel being tracked. A sanity check is to make sure the IMO number is the same for the suggested route.

A variable holding all unique dates in the subset is needed for separating the data even more. This is based on the assumption that every data point holding the same date constitute one unique route. The number of data points vary from date to date. We need to keep track of the number of data points in each route. At last, each set of routes made, we want to find vessels travelling to some end point. Each defined route does not necessarily share the same end point. There are various reasons for this some of which include: vessels turn off their AIS-sender, there could be a malfunction, vessels are not necessarily bound to travel to the same destination. We choose to separate each route into three categorise: vessels travelling to, through and from the destination. This classification is of interest as when using the flow field we want every vessel to converge to the same point.

A distance measure between two points is needed. The Euclidean metric is a useful tool for this operation. We create a new column in the df and calculate the absolute distance between the wanted end point and every registered positional coordinate. This is calculated by:

$$d_i = \sqrt{(x_i - x_{final})^2 + (y_i - y_{final})^2} \tag{1}$$

x_i and y_i are the ith longitude and latitude values, respectively. x_{final} and y_{final} are the end points longitude and latitude values, respectively.

For each route we check if the distance is decreasing or increasing and make the assumption: If the last absolute distance value is larger than the first, we conclude that the vessel is travelling from the destination.

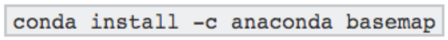
Vessels travelling through is measured by counting the number of times the distance increase occurs. If the count is larger then some number, we conclude the route is travelling through. And vice versa for route travelling to. By making this classification, we can easily discard routes not of interest.

With the use of Pandas we have been able to extract subsets of data which in this case makes it less chaotic when visualizing the data. We have been able categorize and produce routes from the data. In addition, a basic classification determining the vessels' direction and end points is made.

Basemap

Creating the map in python is done with the matplotlib basemap toolkit. Together with matplotlib Basemap can represent geospatial data onto different map projections. By sorting the AIS-data in terms of some vessel characteristics the array containing vessel information can be used to read the longitude and latitude values needed for setting up the boundaries of the map. The necessary requirements for Basemap are: Matplotlib 1.0.0 (or later, download) , Python 2.6 (or later, including Python 3) (download) , Matplotlib 2.2 LTS requires Python 2.7 or later , Matplotlib 3.0 requires Python 3.5 or later , NumPy 1.2.1 (or later) , Array support for Python (download) , PROJ4 Cartographic Projections Library. REF matplotlib siden!

If using anaconda Basemap can be installed by the command in terminal: When creating a map with Basemap an class object instance is made. The object as attributes and methods which are treated equally as any other class object in



```
conda install -c anaconda basemap
```

python. The arguments passed into the Basemap object the boundary points, projection type, resolution. Plotting the map is done using matplotlibs standard plotting commands.

Creating map

Given the AIS-data we now can limit our map to the maximum and minimum values of our the data. The range of longitude and latitude degrees are (-180, 180) and (-90,90) respectively. 0 degrees of longitude is the Prime Meridian which goes through Greenwich, England. negative values represent the western hemisphere and positive values represent the eastern hemisphere. For degrees of latitude, the 0 degree is placed at equator. Therefore, negative values represent the southern hemisphere and positive values the northern hemisphere.

The maximum and minimum values can found by:

```python
minlon = max(-180, min(df['Longitude']))
minlat = max(-90, min(df['Latitude']))
maxlon = min(180, max(df['Longitude']))
maxlat = min(90, max(df['Latitude']))
```

Listing 2: Boundary points for the map can be found from simple max and min function in the standard python library

The projection chosen for this map is cylindrical

Functions

Find land(x, y), sourcepoint, sourcepanel,

Flows

Class property (source, sink, doublet, uniform), geometric and strength solver. Solving system of equations.

Trouble with division of zero.

Creating flow environment