HyosPy User Manual

The University of Texas at Austin

1 Overview

HyosPy (Hydrodynamic and oil spill Python) is a model coupling system developed by Dr. Ben R. Hodges' research group in CWE at University of Texas at Austin. This system is able to generate an ensemble of hydrodynamic and oil spill simulations based on the available forecast/hindcast data, and allows the user to visualize the result on various platforms. The new version of HyosPy integrates a estuarine hydrodynamic model - SUNTANS, a shelf hydrodynamic model - ROMS, an oil spill model - GNOME, a drifter tracking Python wrapper - TracPy and several post-processing tools. HyosPy offers multiple options for running different combination of these numerical models. The dataset formats of the models are converted internally, so that no manual modification is required. The bay model and the shelf model are dynamically linked that provides a downscaling/upscaling blended current field for the oil spill transport models. The result can be visualized as particle trajectories or probability maps on Google Map/Earth and Python basemap.

An user can choose to run all the models or just some of them according to the specific oil spill situation. For example, if a spill occurs in the estuary, in which hourly prediction is important, running the estuary model alone will be an appropriate choice. Running HyosPy requires the user to write their own run script in Python. This script allows the users to customize their simulation by specifying a few parameters, including the choice of hydrodynamic models, oil transport models, number of simulations and visualization methods. A properly written run script will yield accurate simulation result and be time-saving. This user manual provides detailed information about how to write such a script and making your own customized oil spill simulations.

2 Hydrodynamic Model

The Python script can be created by using any of your favorite text editors or Python IDEs. In the example, the script is named as `Hyospy.py`.

First, import the `upper_wrapper` module and initialize the `upper_wrapper` object. As its name suggests, the `upper_wrapper` class is a uppermost level module that controls the whole process. Then we are able to modify the parameters in `upper_wrapper.py` by redefining them in `Hyospy.py`.

```
from upper_wrapper import upper_wrapper

UW = upper_wrapper()
```

Once the object is initialized, we can set the parameters for a customized HyosPy simulation. The start time and end time for the hydrodynamic model are defined as the example below. Note, although these times are corrected to hours, ′00′ is highly recommended for setting the hours in `starttime` and `endtime`. A more detailed time range can always be defined for the oil spill model by setting the start time, `starttime2` and the running period, `period`. The running period for oil spill models has to be less than that of the hydrodynamic model by at least one hour, or GNOME will raise errors.

The time information of the hydrodynamic models is distinct from that of the oil spill model. This allows a user to run a longer simulation for the hydrodynamic model. Below is an example.

```
## Time information for hydrodynamic models
starttime = '2017-05-29-00'
endtime  = '2017-05-31-00'

## Time information for oil spill models
starttime2 = '2017-05-29-20' # starttime for oil spill model
period = 20 # unit: hours
```

After the running time is determined, a user needs to specify the ensemble parameters, including the number of simulations `number` and the time interval `interval`. As time elapses, more observational data will become available. By setting `number=n` where `n`>1, HyosPy will automatically replace the forecasting data with observations and start a new simulation. `interval` is the time interval between multiple simulations and has the unit of seconds. The smallest value for `interval` is 10800, which equals 3 hours. This is because the tidal elevation data at SUNTANS open boundary and the wind data are updated every 3 hours. If a simulation starts when data has not been updated, it simply remains the same as the last simulation.

The choice of these two parameters depends on the simulation time. For instance, if the running period of the oil spill simulation is two days, i.e. `period=48`, and `interval` is set as 10800, there should be at most 16 (48/3) separate simulations that are different with each other. The last one will be using all the observational data. Below is an example.

```
## ensemble parameters
UW.number = 16
UW.interval = 10800 # units: seconds
```

The next step is to make our choice for the hydrodynamic models. The parameter for choosing the hydrodynamic model is called `hydro_model`. HyosPy provides three options: 'SUNTANS', 'ROMS' and 'BLENDED'. Each one refers to a hydrodynamic model.

```
## Hydrodynamic model
UW.hydro_model = 'SUNTANS' # hydrodynamic model options: 'SUNTANS',
    'ROMS', 'BLENDED'
```

```
UW.hydro_run = True # choose whether or not to run a hydrodynamic model
    first
```

In the current version of HyosPy, the SUNTANS model only extends over the Galveston bay. For other areas, we rely on the Texas shelf model, ROMS. A user should choose the model based on the spill situation.

1. `hydro_model = 'SUNTANS'`. If the oil spill occurs in Galveston Bay and is not expected to transport offshore soon, SUNTANS is recommended.

2. `hydro_model = 'ROMS'`. If the oil spill occurs nearshore or offshore, and only the approximated spill trajectory is needed, a user should use ROMS directly which saves the time of running SUNTANS.

3. `hydro_model = 'BLENDED'`. When the spill is likely to transport across bays and the Texas shelf, a user should use the blended product of SUNTANS and ROMS.

However, if the output of the hydrodynamic model is already available, running it again is time consuming and not necessary. The user can choose whether or not he needs to run the hydrodynamic model first by modifying the parameter `hydro_run`. This parameter adds some flexibility to the entire system. However, rerunning the hydrodynamic model is recommended in the forecast simulation.

The output of the ROMS model is available on the Texas A&M server. The user can determine to download the data while running or before running the system by modifying the parameter `ROMS_datasource`.

```
## ROMS datasource
UW.ROMS_datasource = 'offline' # ROMS data source options: 'online',
    'offline'
```

1. By choosing the `'online'` mode, the ROMS data is downloaded during the simulation. For example, if SUNTANS requires ROMS output as its initial or boundary condition, the SUNTANS simulation will only start after the downloading process has ended. If you run an ensemble of simulations, the data is downloaded multiple times.

2. The `'offline'` mode allows downloading the ROMS data ahead of any HyosPy simulations. This ensures the simulation will not terminate as a result of failure of accessing the online data. By choosing `'offline'`, the HyosPy system will assume the ROMS output has already been downloaded and is available for use in the hard drive of the local computer.

The choice of the suitable parameter `ROMS_datasource` depends on a few factors. (1) The online server gets unstable sometimes when people are uploading, updating or reorganizing the data. In such situation, the downloading process can easily terminate. (2) The download speed depends on the network environment. As a

result of the model's domain size, the ROMS dataset is large. So we read the data at every time step in the formant of OPENDap. Downloading the ROMS data may cost several hours under a poor Internet speed. Hence, it is better for the user to have the ROMS data ready on file, and set `ROMS_datasource='offline'`, unless the updating of hindcast/forecast information is required for each ensemble of the simulations. HyosPy provides a library for downloading the ROMS data. All a user need is to create a Python script, call the library and specify the time period. The default saving directory of the ROMS data is in /DATA. An example Python script is given as below:

```
import lib
from ROMS_offline import offline_download

starttime='2017-05-29-00'
endtime='2017-06-02-00'
roms = offline_download(starttime, endtime)
outfile = 'DATA/ROMS_offline_data.nc'
roms.Writefile(outfile)
roms.Go()
```

When `hydro_model` is set as `'SUNTANS'` or `'BLENDED'`, SUNTANS is the primary hydrodynamic model in HyosPy. The user has many options to run the hydrodynamic model, SUNTANS. HyosPy provides a few choices for the boundary and initial conditions. Changes can be made in the parameter `OBC_opt` and `IC_opt`.

```
## SUNTANS IC and BC
UW.OBC_opt = 'ROMSFILE' # Type 3 boundary condition option:
#'file','OTIS', 'ROMS', 'ROMSOTIS','ROMSFILE', 'ROMSOTISFILE'
UW.IC_opt = 'ROMS' # initial condition options: 'constant', 'ROMS'
```

As the name indicates, `OBC_opt` is short for open boundary condition options. The available options are: (1) observational data from NOAA stations, `OBC_opt='file'`; (2) predicted tides from OSU Tidal Inversion Software, `OBC_opt='OTIS'`; (3) ROMS data at the boundary, `OBC_opt='ROMS'`; (4) a combination of ROMS and OTIS tidal model, `OBC_opt='ROMSOTIS'`; (5) a combination of ROMS and observational data, `OBC_opt='ROMSFILE'`; (6) a combination of ROMS, OTIS and observation, `OBC_opt='ROMSOTISFILE'`. Currently, the best prediction is obtained by using all the three data sources, `'ROMSOTISFILE'`. However, this relies on the availability of the ROMS data. The default value of `OBC_opt` is set to be `'FILE'`, where the observation only at the open boundary can yield reasonable prediction in the estuary and ROMS is not required.

The initial condition `IC_opt` is simple. When ROMS output is available on the disk, we use an interpolated current field as SUNTANS initial condition, i.e. `IC_opt='ROMS'`. The default setting of IC is that we simply make a 0 initial condition and set `IC_opt='constant'`. Since a spin-up time is expected for this type of IC, HyosPy internally pushes forward the start time by 5 days and increases the running period. This will only increase the running time by 15 minutes. More de-

tailed explanation of how to modify the SUNTANS initial and boundary condition can be found on the SUNTANSpy page on Github. Other settings, such as the hydrodynamic parameters (in directory SUNTANS/SampleCase/suntans1.dat), the number of processors (in directory SUNTANS/Makefile), require the users to change by themselves. However, unless the HyosPy system is relocated to a new estuary, these parameters should be kept the same as the model has been validated already in a previous study. The information of the SUNTANS model is included in the directory /SUNTANS, while the output is saved in /SUNTANS/rundata. The output is updated every time a new ensemble of simulations starts.

3 Oil Spill Model

Once the parameters for the hydrodynamic models are determined, we need to make options for the oil spill models. There are two oil transport models available in HyosPy, i.e. GNOME and TracPy. GNOME is the General NOAA Operational Modeling Environment that is designed for simulating the fate and trajectories of the oil spill. This model is efficient and can run on many types of grids, e.g. curvilinear, regular and unstructured. It also has many features to model the physical and chemical evolution of a spill. GNOME is the primary oil spill model in HyosPy. Although the parameters for running GNOME has already been set, the user can modify them in `oilspill_wrapper.py`. More details can be found in their user manual.

As a complement of GNOME, the user can choose to run TracPy for comparisons. TracPy is a Python wrapper of TRACMASS, a drifter transport model that is designed for Arakawa-C grid, or ROMS-type grid. TRACMASS is efficient and uses a new algorithm other than the Runge-Kutta method, which is used in most Lagrangian transport models. TracPy will only work when `hydro_model='ROMS'` or `'BLENDED'`. Similar to GNOME, the parameters of TracPy is determined in `oilspill_wrapper.py`.

The user is able to run GNOME and TracPy separately, or both by turning on/off the parameters, `runGNOME` and `runTracPy`. For example, `runGNOME=True` makes HyosPy run GNOME. The system creates a new folder, '/GNOME'. If the ensemble parameter, `number`> 1, subfolders will be created under the name of the sequence number starting from 0. In each subfolder, there exists the input and output of a GNOME simulation. Similarly, turning on `runTracPy` creates the folder '/TracPy' and runs the model in each subfolder.

In HyosPy, we have more advanced options for running GNOME. The grid for the blended model (`hydro_model='BLENDED'`) has the same domain size as the original ROMS model but a higher resolution at the Galveston Bay area. The total number of the grid cells is approximately 10^6. GNOME runs slowly on this grid. It may take over 40 minutes to complete a 2 days' oil spill simulation. Hence, the parameters, `gnome_subset` and `gnome_bbox`, allow the user to subset one portion of the entire domain. For example, by setting `gnome_subset=True` and `gnome_bbox` be any sub-domain in the area of interest, the running time could be reduced signifi-

cantly. In the example below, where `gnome_bbox=[28.17,-95.53,30.0,-93.9]` covers the Galveston Bay and its adjoint coastal region, the running time of GNOME is 10 minutes compared with 40 minutes on the full domain. Comparatively, TracPy is more efficient on curvilinear grids. The choice of the oil spill models and the related parameters depends on the real situation.

```
## Oil spill model
UW.runGNOME = True
UW.runTracPy = False

## GNOME settings
UW.gnome_subset = True    # For blended current product, "subset=True"
    makes GNOME run faster
UW.gnome_bbox = [28.17,-95.53,30.0,-93.9]
```

4 Visualization

HyosPy provides multiple tools for visualizing the result of the oil spill trajectories. GNOME and TracPy have their own visualizing method. For GNOME, in the folder '/images', which is under any GNOME's directory, there are png figures showing particles' locations at every time step and a gif figure that shows the oil movement. For TracPy, there is a sub-folder named '/figures', in which the spill trajectories are included.

Google Map and Google Earth are the primary visualization tools in HyosPy and are naturally embedded in the system. The related files will be generated after each oil spill simulation and are named as

```
GNOME_Google_map.html,
GNOME_GE.kml,
TracPy_Google_map.html,
TracPy_GE.kml.
```

As the names indicate, the files represent the trajectories of GNOME and TracPy, respectively. Since Google Map/Earth are cross-platform softwares, the user can either open the files in the local machine or transfer them to a smart phone, tablet or laptop to look at the results. All they need is a browser and the Google Earth software. Google Earth also allows the users to make animations of the oil transport.

In spite of visualizing the spill trajectory, we can also look at the probability of multiple predictions by turning on the feature of the probability map. The related parameters are `probability_map`, `google_earth` and `mpl`. By setting `probability_map=True`, HyosPy will automatically calculate the probability of the spill trajectory. If `number`= 1, the probability map is from a single prediction. Otherwise, the probability is calculated from an ensemble of predictions, which makes more sense. Because we are able to find the area that is most likely to be contaminated by oil regardless of the uncertainty of boundary informations.

The heavy job of calculating probability is taken by a C++ script in the directory,

'/DATA'. So make sure the C++ compiler is installed in the machine. The user is able to choose whether he wants the probability map to be overlaid on Google Earth by turning on/off the parameter `google_earth`. If `google_earth=True`, a kml file named `GNOME_PM.kml` will be created in the directory '/GNOME'. However, this application of Google Earth usually requires high performance, especially a large memory of the video card. Instead, turning off `google_earth` yields much faster visualizations. The resulting map is rendered on a Python library, basemap. The figure will pop up and the user is able to zoom in and save the figure. `mpl` is the parameter that adjust the speed of generating the probability map. Increasing this number can accelerate the process but will lower the accuracy of the map. Note the value has to be the multiple of 4 and the default value is 8. More details can be found in the library `Probability_map`.

```
## Probability Map
UW.probability_map = True
UW.google_earth = False
UW.mpl = 8 # probability calculation time interval
```

After all the parameters have been specified, we should be able to call the Python class and run the system. The required input variables are the time information of the hydrodynamic model and the oil spill model. The initial spill location is assigned to `init_latlon` in the format of latitude and longitude.

```
## run the system
UW(starttime, endtime, starttime2, period, init_latlon=[29.463089,
    -94.843460])
```

Note, the parameters above are not required in all the scenarios. The examples in the appendix provide more information about modifying the parameters, creating your own run script and successfully running the HyosPy system.

A Examples

A.1 Example 1

In the appendix, we will introduce some examples of using different models in HyosPy. The scripts of examples can be found in the directory /tests.

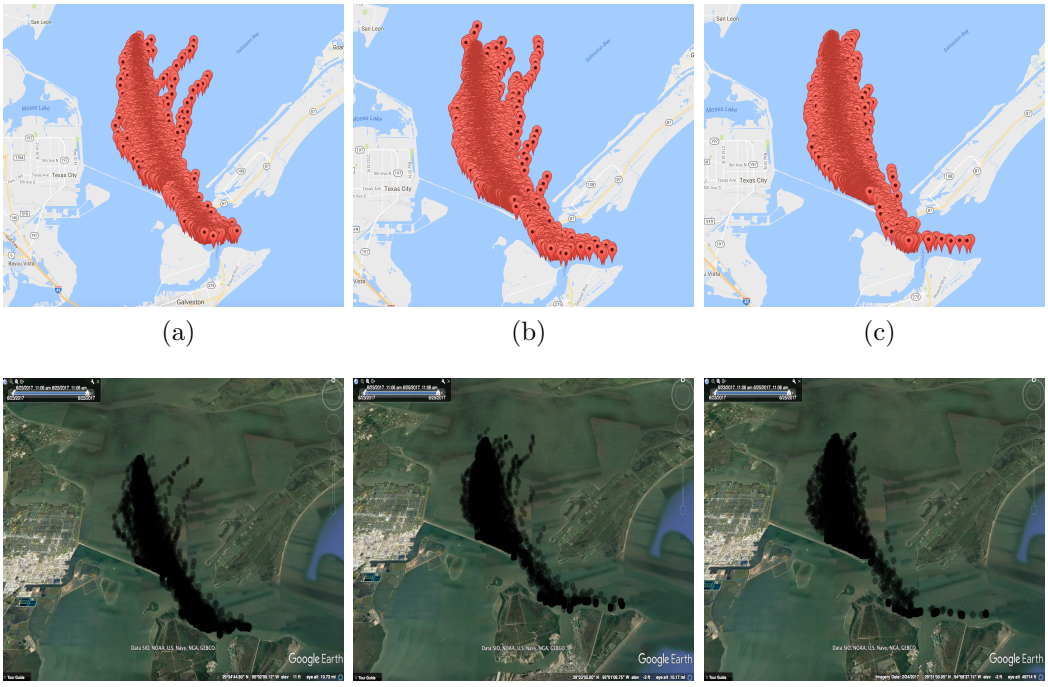
The first example uses SUNTANS as the hydrodynamic model and GNOME as the oil spill model. The related run script is called `hyospy_SUNTANS.py`. The open boundary condition of SUNTANS is sourced from the observational gauge at Freeport, while the initial condition is constant and there is a 5 days' spin-up time. The start times of the two models are different. There are 3 simulations in total and their time interval is 3 hours. We ran a 48 hours' simulation for the hypothetical oil spill that is initiated inside the Galveston Bay. The resulting spill trajectory is visualized in Google Map and Google Earth. A probability map is created based on the three simulations and is overlaid on Google Earth.

Figure 1 shows the result of the 3 simulations. Since the spills occurred during

ebb tides, they were driven by strong outflows and most of the oil beached on the barrier islands. This explains why the variances are small among cases. Still, as the boundary information and wind data was updated, there was slight difference of the spill trajectories. Figure 2 is the probability map overlaid on Google Earth, and figure 3 is the probability map created by Python basemap library. The map shows that the spill beached and accumulated on the Texas City Dike and Pelican Island, while the probability of oil transporting in other directions is equally likely.

```
from upper_wrapper import upper_wrapper

UW = upper_wrapper()
## Hydrodynamic model time info
starttime='2017-06-23-00'
endtime='2017-06-26-00'

## Oil spill model time info
starttime2='2017-06-23-15' # starttime for oil spill model
period=48

## ensemble parameters
UW.number = 3
UW.interval = 10800 # units: seconds

## Hydrodynamic model
UW.hydro_model = 'SUNTANS' # hydrodynamic model options: 'SUNTANS',
    'ROMS', 'BLENDED'
UW.hydro_run = True # choose whether or not to run a hydrodynamic model
    first

## ROMS datasource
UW.ROMS_datasource = 'offline' # ROMS data source options: 'online',
    'offline'

## SUNTANS IC and BC
UW.OBC_opt = 'file' # Type 3 boundary condition option: 'constant',
#'file','OTIS', 'ROMS', 'ROMSOTIS','ROMSFILE', 'ROMSOTISFILE'
UW.IC_opt = 'constant' # initial condition options: 'constant', 'ROMS'

## Oil spill model
UW.runGNOME = True
UW.runTracPy = False

## GNOME settings
UW.gnome_subset = True    # For blended current product, "subset=True"
    makes GNOME run faster
UW.gnome_bbox = [28.17,-95.53,30.0,-93.9]
```

```
## Probability Map
UW.probability_map = True
UW.google_earth = True
UW.mpl = 8 # probability calculation time interval

UW(starttime, endtime, starttime2, period, init_latlon=[29.463089,
    -94.843460]) #SUNTANS domain
```



(a) (b) (c)

(d) (e) (f)

Figure 1: The results of the 3 simulations visualized on Google Map and Google Earth

Figure 2: The probability map on Google Earth

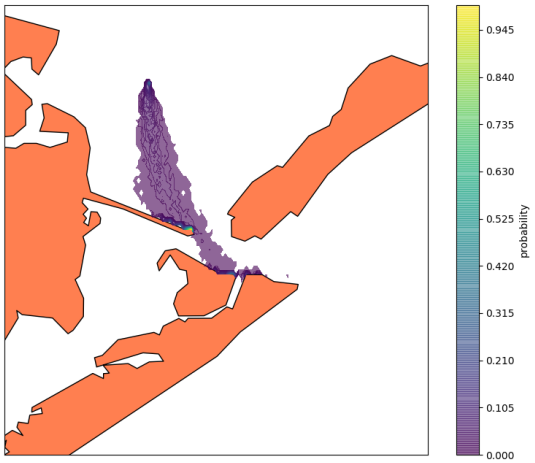


Figure 3: The probability map on Python basemap

A.2 Example 2

Example 2 used ROMS as the hydrodynamic model, GNOME and TracPy as two oil spill transport models. The related run script is called `hyospy_ROMS.py` and is shown as below. The ROMS output is downloaded and saved in /DATA. So we choose the "offline" mode, where `UW.ROMS_datasource = 'offline'` and `UW.hydro_run = False`. This setting ensures the program does not terminate as a result of the failure of ROMS data server. By specifying `UW.parameter=1`, there is only a single simulation. By turning on the flag of `runGNOME` and `TracPy`, we are able to use the two Lagrangian transport models. In the last line of the script, the initial spill location is specified in the coastal region.

The modeled trajectories of GNOME and TracPy are shown in figure 4 and 5. Figure 4(a) and 4(b) are the spill trajectories modeled by GNOME, while the latter one is zoomed. Figure 4(c) is the oil transport predicted by TracPy. By comparing the (b) and (c), we can see that although there are differences between the predictions of the two Lagrangian transport models, the overall trends of oil particles are similar. The differences are due to many factors, such as the algorithm of Lagrangian transport and the diffusion of oil spill. Figure 5 is the probability map of this simulation.

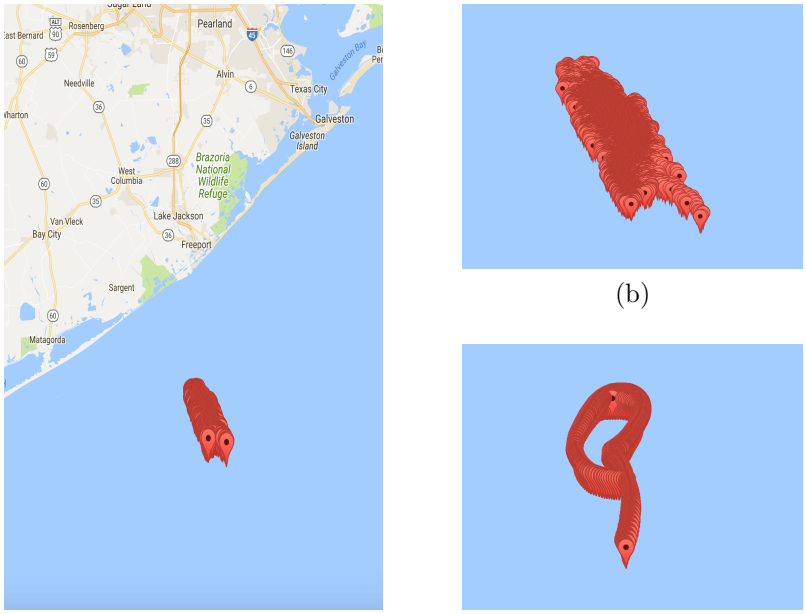
```
from upper_wrapper import upper_wrapper

UW = upper_wrapper()
## Hydrodynamic model time info
starttime='2017-05-29-00'
endtime='2017-05-31-00'

## Oil spill model time info
```

```
starttime2='2017-05-29-05' # starttime for oil spill model
period=40

## ensemble parameters
UW.number = 1
UW.interval = 10800 # units: seconds

## Hydrodynamic model
UW.hydro_model = 'ROMS' # hydrodynamic model options: 'SUNTANS', 'ROMS',
    'BLENDED'
UW.hydro_run = False # choose whether or not to run a hydrodynamic model
    first

## ROMS datasource
UW.ROMS_datasource = 'offline' # ROMS data source options: 'online',
    'offline'

## Oil spill model
UW.runGNOME = True
UW.runTracPy = True

## Probability Map
UW.probability_map = True
UW.google_earth = True
UW.mpl = 8 # probability calculation time interval

UW(starttime, endtime, starttime2, period, init_latlon=[28.353786,
    -95.315109]) #ROMS domain
```

A.3 Example 3

In Example 3, we used the blended product of SUNTANS and ROMS as the hydrodynamic model, GNOME and TracPy as the oil spill model. In this case, the bay model and the shelf model are dynamically linked, which provides a cross-scale current field for oil transport predictions.

The run script for example 3 is called `hyospy_blended.py` and is shown as below. The choice of the hydrodynamic model is specified as `UW.hydro_model='BLENDED'`. Similar to example 2, the ROMS data is downloaded ahead of simulation by choosing `UW.ROMS_datasource='offline'`. Compared with example 1, since the ROMS data is available, we use the output of the shelf model as the boundary and initial condition of the inner bay model, SUNTANS. `UW.OBC_opt='ROMSFILE'` means that the open boundary of SUNTANS is a combination of the observational surface elevation and the ROMS velocity, salinity and temperature. Additionally, the flag for subsetting the blended velocity field is also turned on, `UW.gnome_subset=True`, where `UW.gnome_bbox` specifies the subsetted area. This setting reduces the simulation time of GNOME significantly and depends on the area of interest.



(a) (b) (c)

Figure 4: (a) Oil spill trajectory modeled by GNOME; (b) zoomed view of GNOME trajectory; (c) zoomed view of TracPy trajectory.

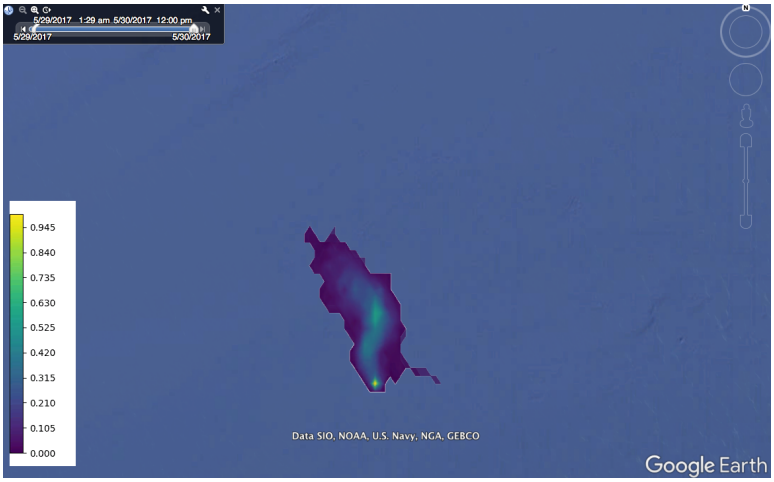
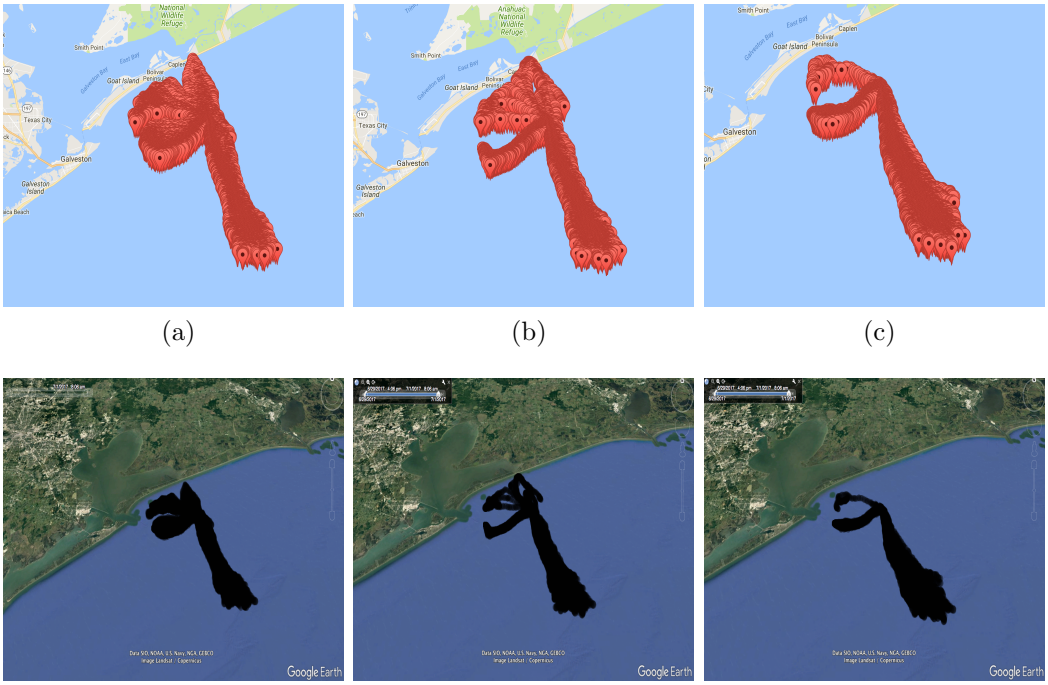


Figure 5: The probability map on Google Earth

The hypothetical spill in test 3 was released near the entrance of Galveston bay. Figure 6 and 7 show the predicted trajectory. As the time intervals between the 3 simulations are 3 hours, there is no evident variation of the overall trend. Most oil were transported towards the offshore direction. However, as the data was updated, the third scenario had less diffusion, while most oil tended to beach on the barrier island in the first scenario when data was predicted. Figure 7 is the probability map of the ensemble. In spite of the extensive diffusion of an oil spill, we were able to find a thin trace that the spill was mostly likely to pass through.

```
from upper_wrapper import upper_wrapper

UW = upper_wrapper()
## Hydrodynamic model time info
starttime='2017-06-29-00'
endtime='2017-07-02-00'

## Oil spill model time info
starttime2='2017-06-29-20' # starttime for oil spill model
period=40

## ensemble parameters
UW.number = 3
UW.interval = 10800 # units: seconds

## Hydrodynamic model
UW.hydro_model = 'BLENDED' # hydrodynamic model options: 'SUNTANS',
    'ROMS', 'BLENDED'
UW.hydro_run = True # choose whether or not to run a hydrodynamic model
    first

## ROMS datasource
UW.ROMS_datasource = 'offline' # ROMS data source options: 'online',
    'offline'

## SUNTANS IC and BC
UW.OBC_opt = 'ROMSFILE' # Type 3 boundary condition option: 'constant',
#'file','OTIS', 'ROMS', 'ROMSOTIS','ROMSFILE', 'ROMSOTISFILE'
UW.IC_opt = 'ROMS' # initial condition options: 'constant', 'ROMS'

## Oil spill model
UW.runGNOME = True
UW.runTracPy = True

## GNOME settings
UW.gnome_subset = True    # For blended current product, "subset=True"
    makes GNOME run faster
UW.gnome_bbox = [28.17,-95.53,30.0,-93.9]
```

```
## Probability Map
UW.probability_map = True
UW.google_earth = True
UW.mpl = 8 # probability calculation time interval

UW(starttime, endtime, starttime2, period, init_latlon=[29.300315,
    -94.611360]) # near Galveston Bay entrance
```



(a) (b) (c)

(d) (e) (f)

Figure 6: The results of the 3 simulations visualized on Google Map and Google Earth in Test 3

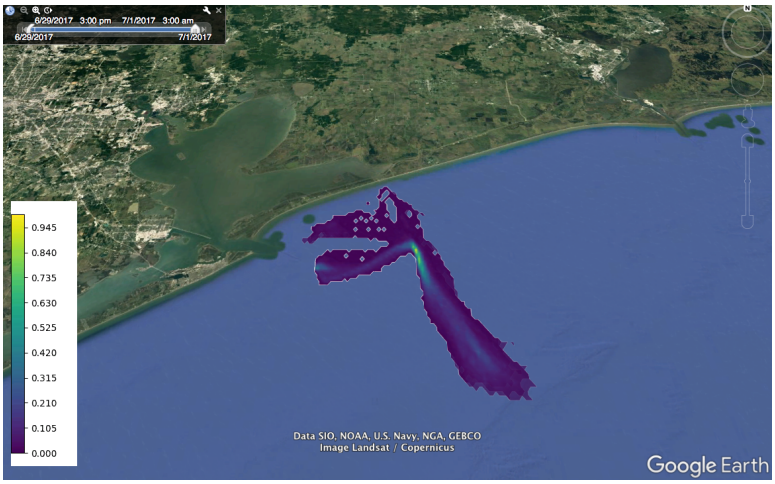


Figure 7: The probability map on Google Earth in Test 3