TracPy: Wrapping the FORTRAN Lagrangian trajectory model TRACMASS

Kristen M. Thyng

Robert D. Hetland
Texas A&M University

July 10, 2014

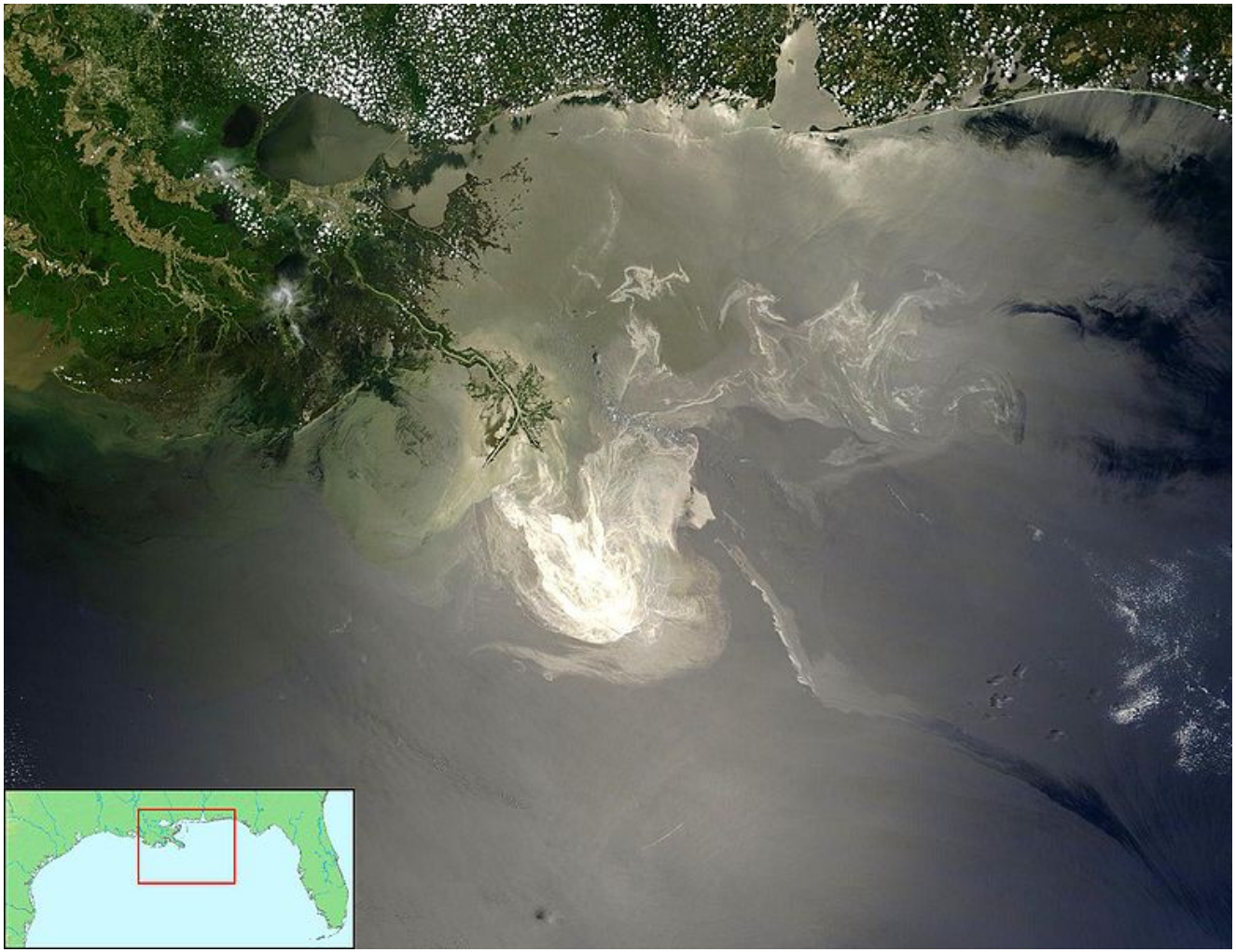
Outline

Drifter Trajectories

Drug Package Origination

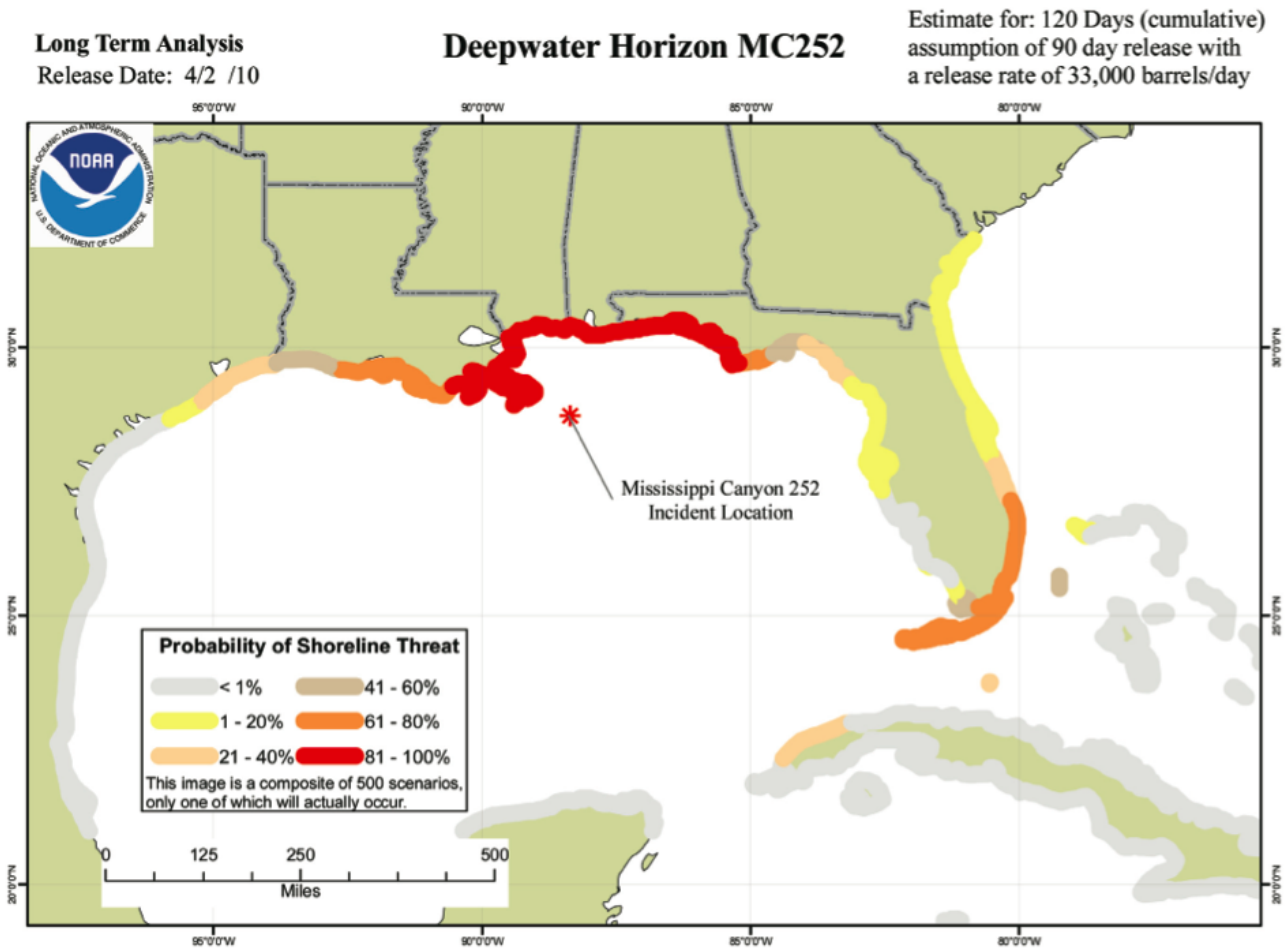
US Coast Guard District 8

Oil spill transport



NASA's Terra Satellite

Oil spill transport

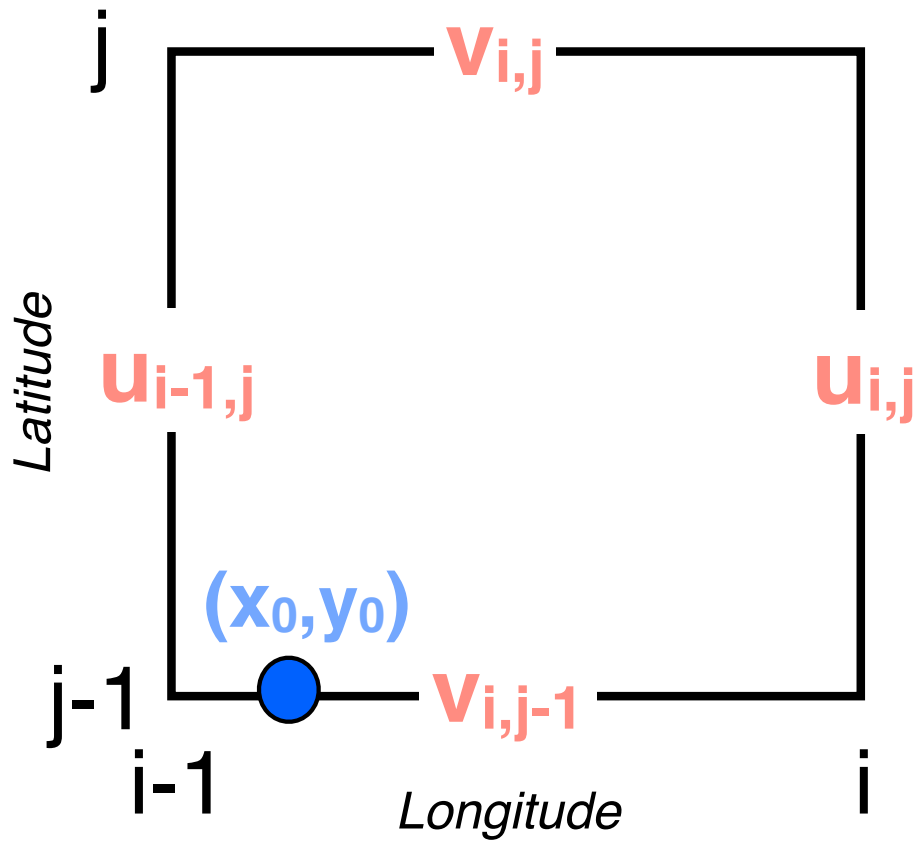


Barker, C. H., 2005, AGU Monograph: *Monitoring and Modeling the Deepwater Horizon Oil Spill: A Record-Breaking Enterprise*

Harmful Algal Bloom Events in Texas Waters

Photo: Texas Parks and Wildlife Department

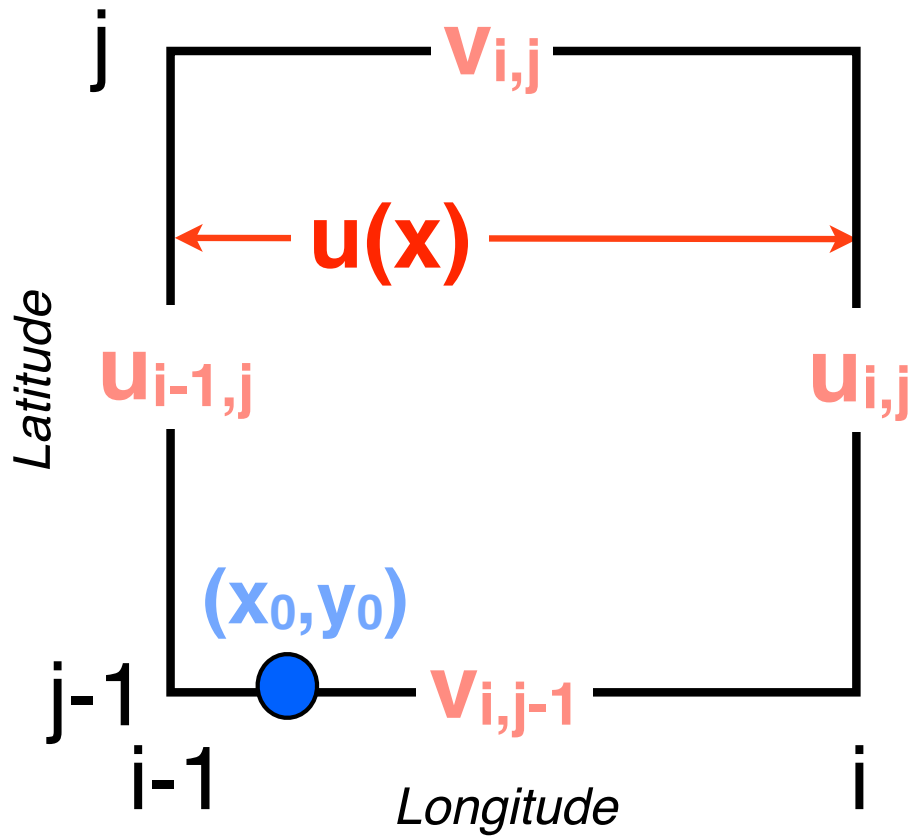
Outline



Horizontal velocities on a staggered Arakawa C grid

After TRACMASS documentation. http://www.tracmass.org, http://doos.misu.su.se/tracmass/

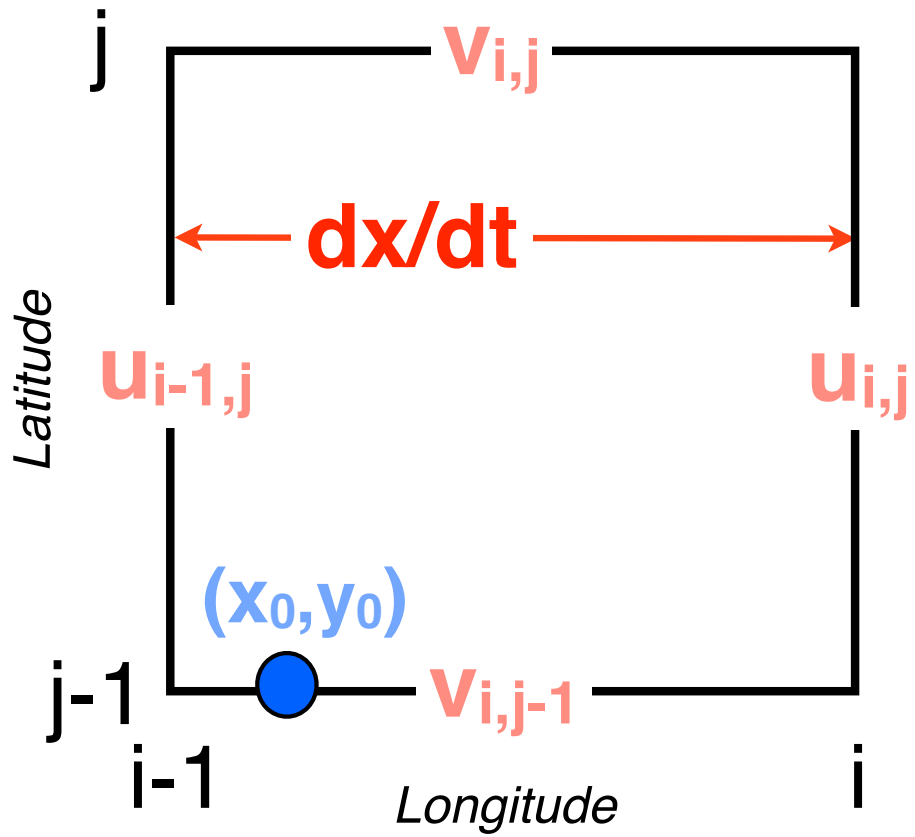
TRACMASS



Linearly interpolate u in x to find $u(x)$ across cell

After TRACMASS documentation. http://www.tracmass.org, http://doos.misu.su.se/tracmass/

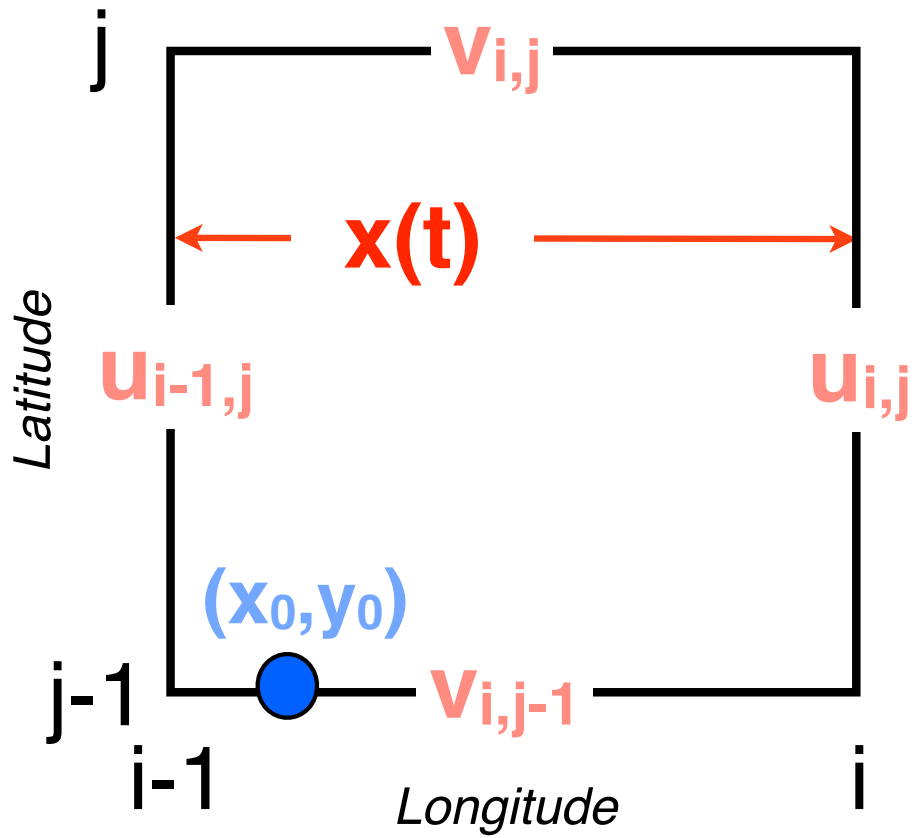
TRACMASS



$$u = \frac{dx}{dt}$$

After TRACMASS documentation. http://www.tracmass.org, http://doos.misu.su.se/tracmass/

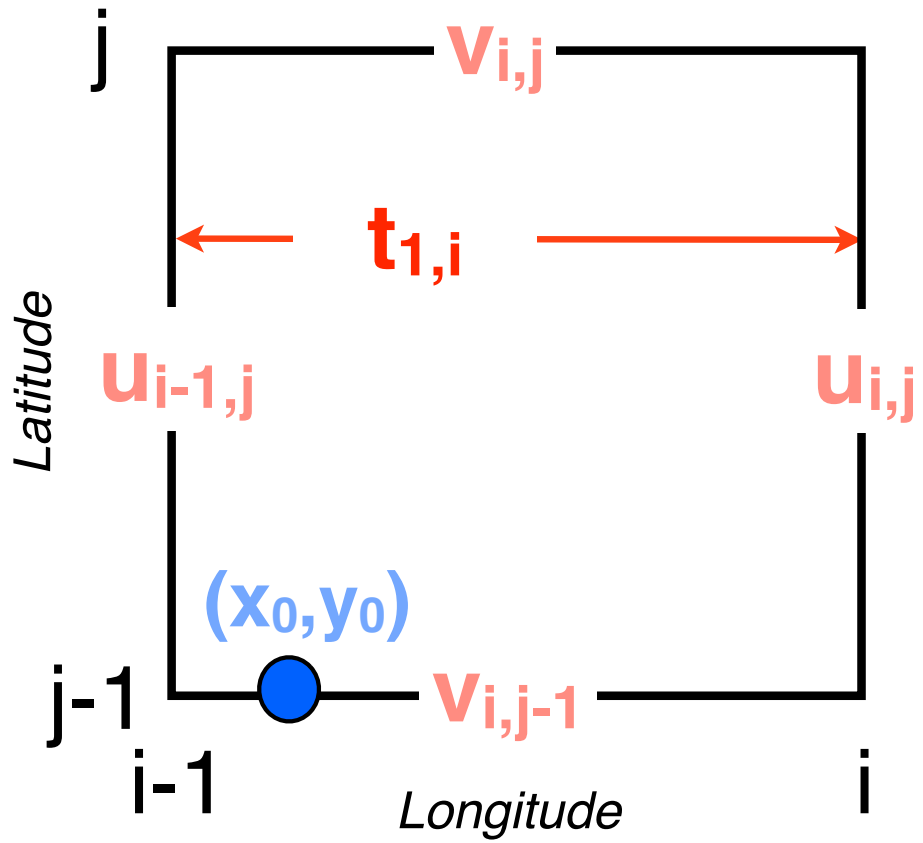
TRACMASS



$\frac{dx}{dt}$ can be analytically solved for $x(t)$

After TRACMASS documentation. http://www.tracmass.org, http://doos.misu.su.se/tracmass/

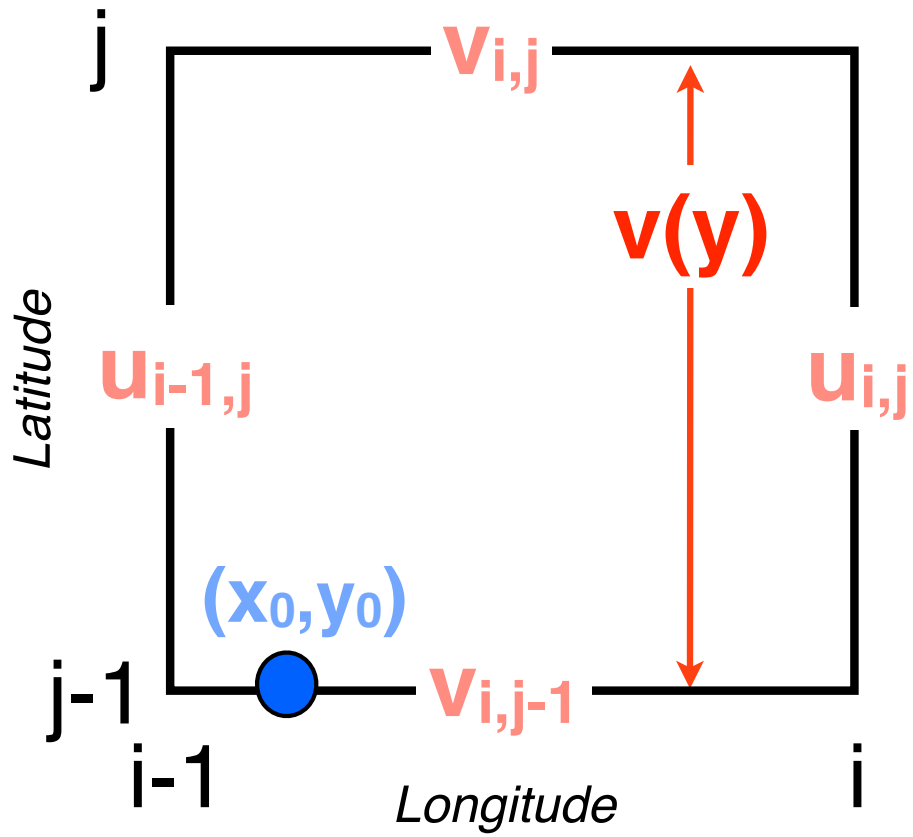
TRACMASS



Solve for the time t when drifter would hit x wall

After TRACMASS documentation. http://www.tracmass.org, http://doos.misu.su.se/tracmass/

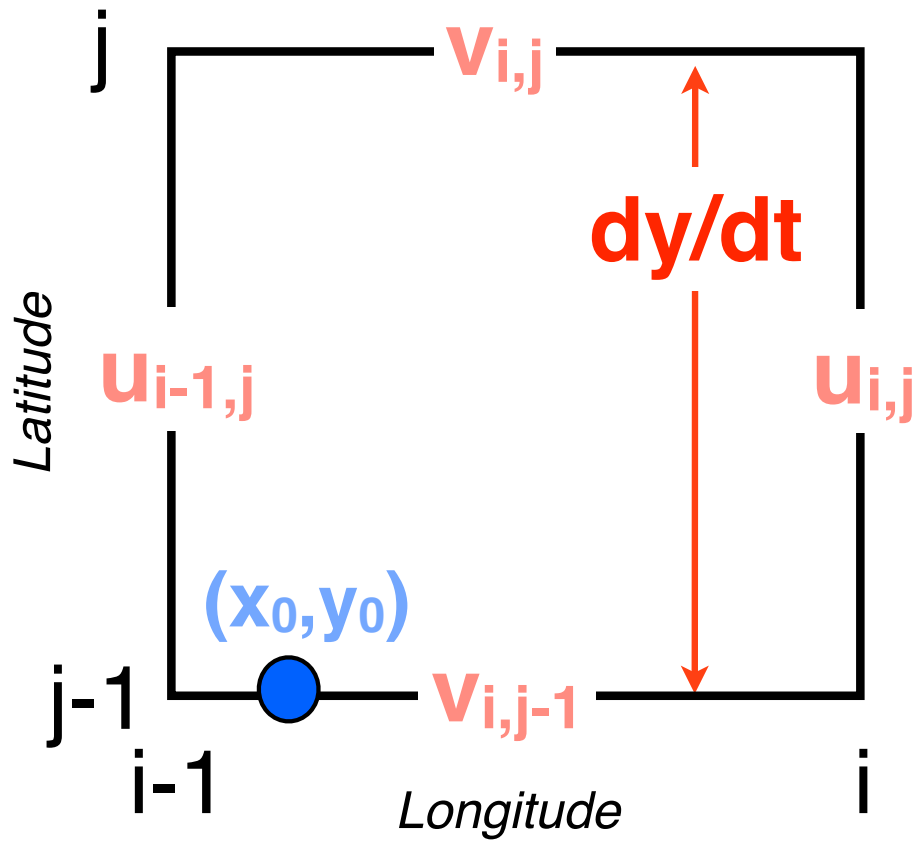
TRACMASS



Same process in y and z directions

After TRACMASS documentation. http://www.tracmass.org, http://doos.misu.su.se/tracmass/

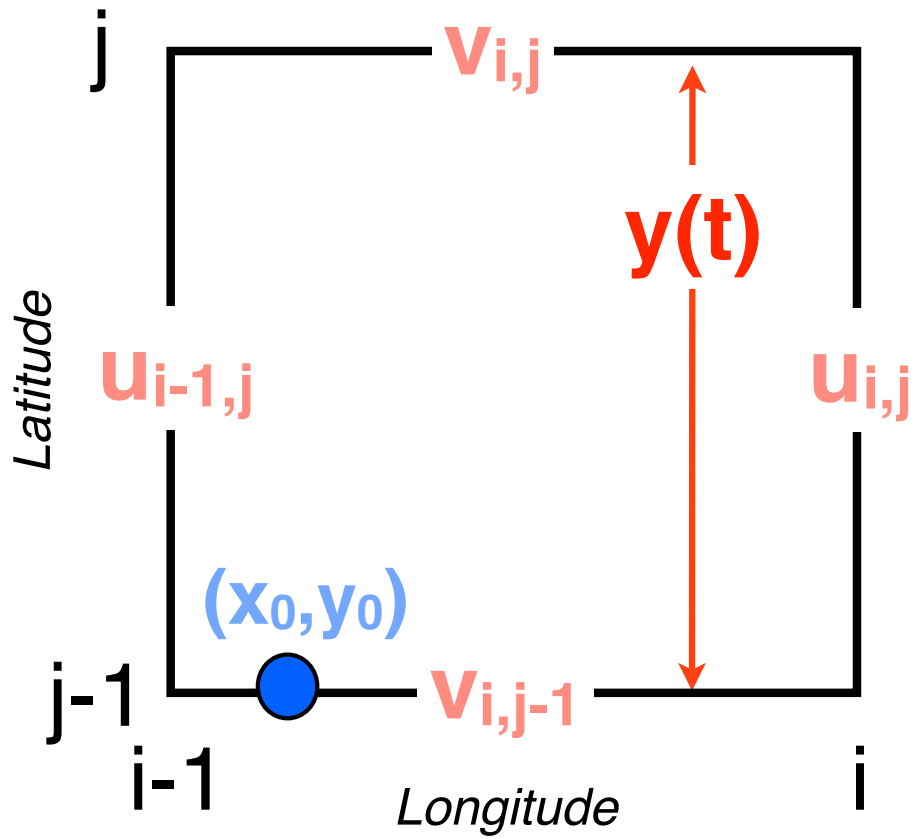
TRACMASS



Same process in y and z directions

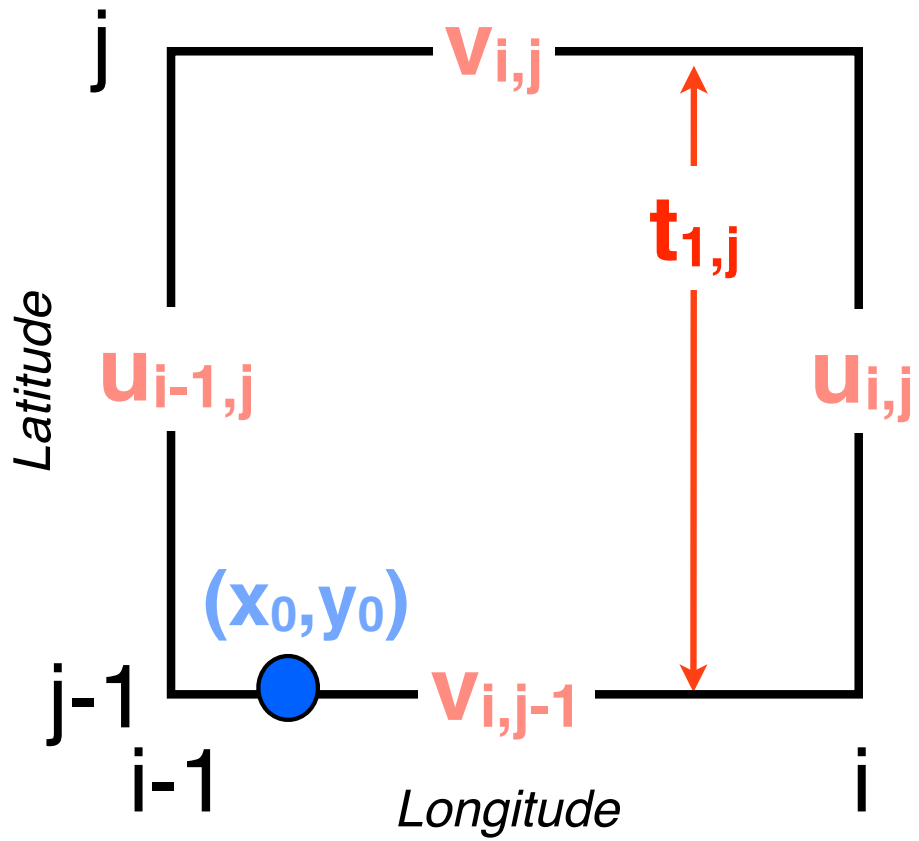
After TRACMASS documentation. http://www.tracmass.org, http://doos.misu.su.se/tracmass/

TRACMASS



Same process in y and z directions

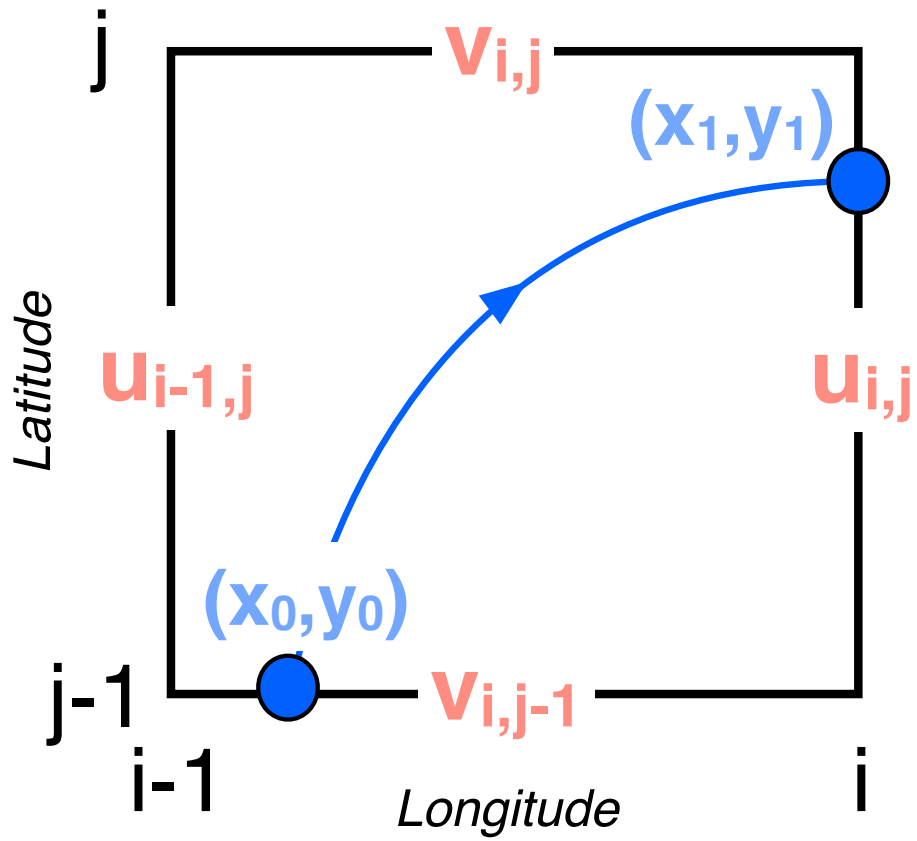
After TRACMASS documentation. http://www.tracmass.org, http://doos.misu.su.se/tracmass/



Same process in y and z directions

After TRACMASS documentation. http://www.tracmass.org, http://doos.misu.su.se/tracmass/

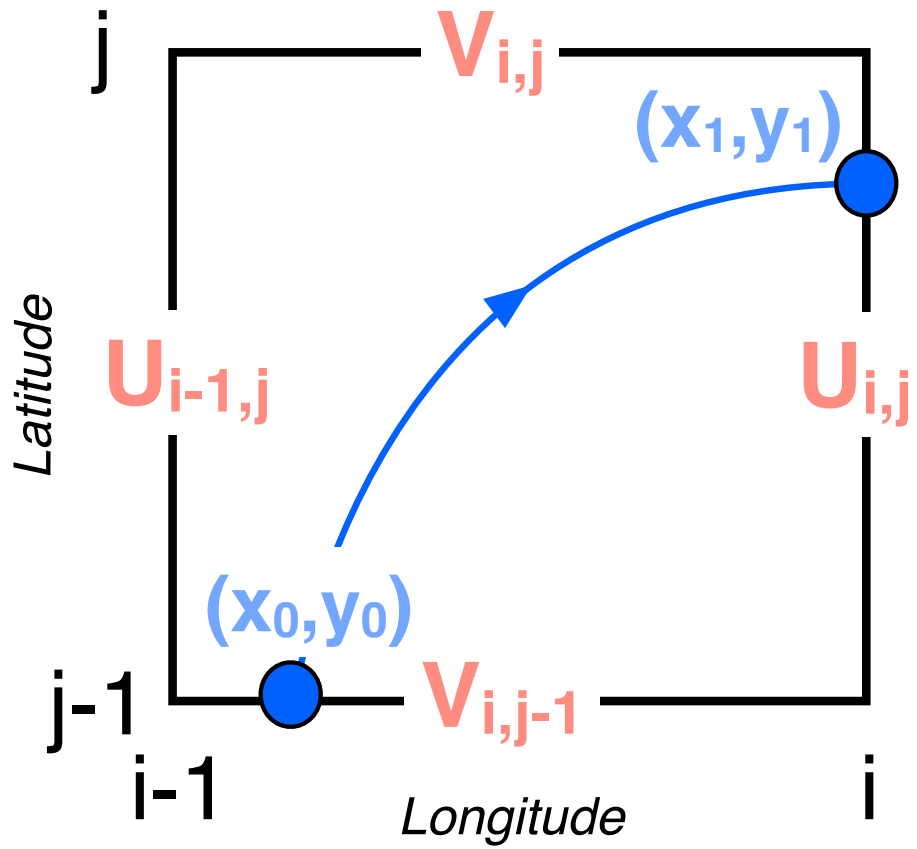
TRACMASS



Minimum overall time is used to calculate position

After TRACMASS documentation. http://www.tracmass.org, http://doos.misu.su.se/tracmass/

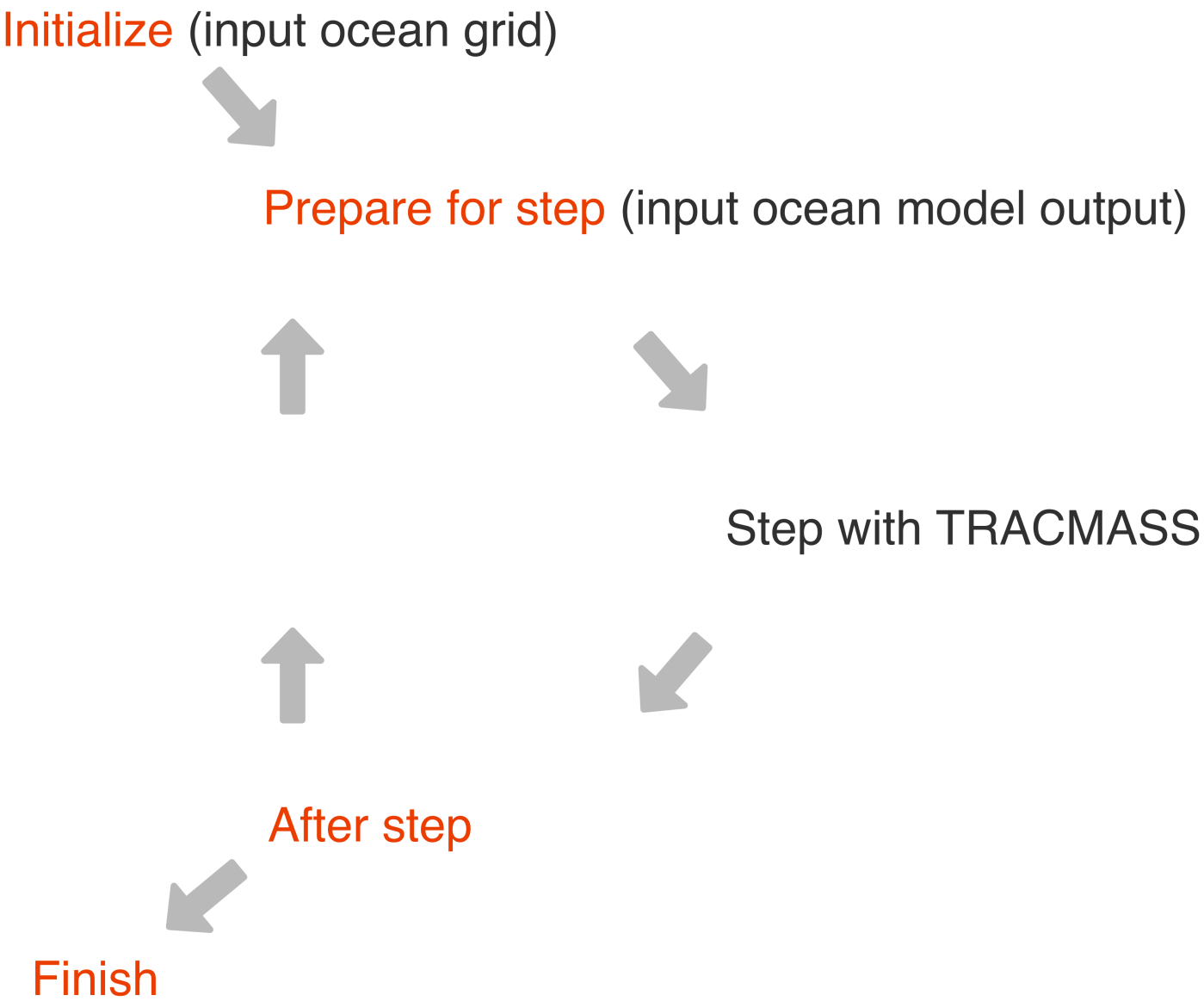
TRACMASS



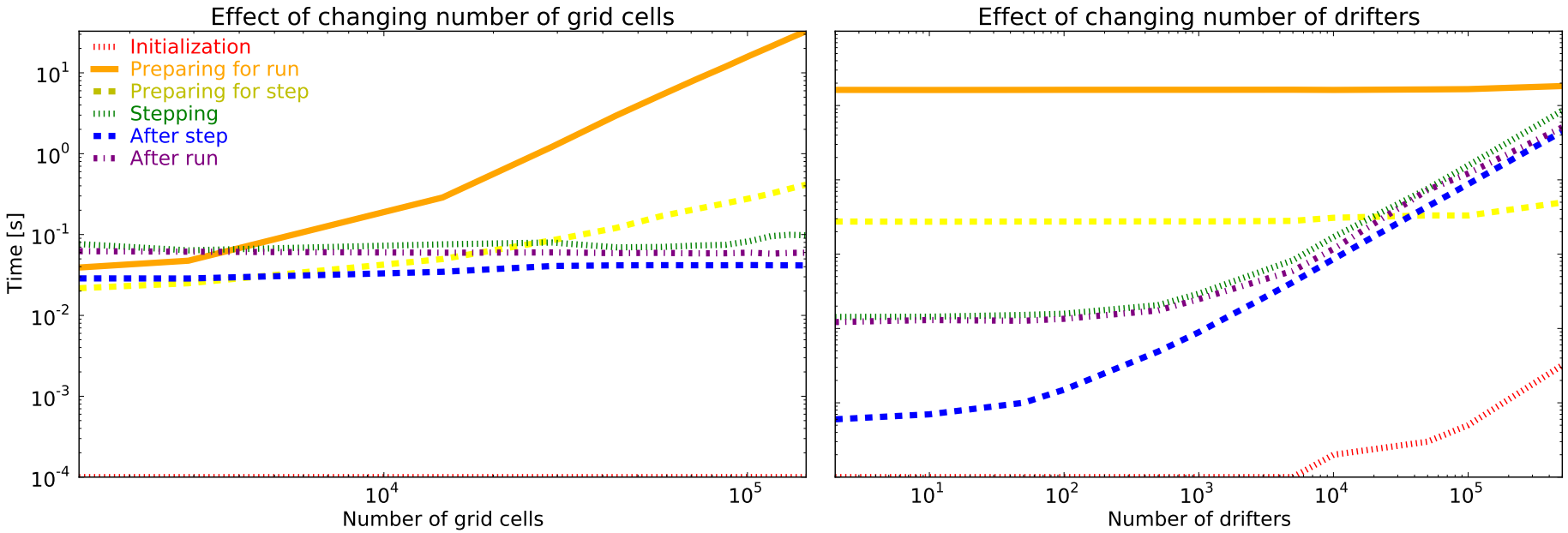
Instead of velocities, use fluxes to allow for differences in grid sizing

After TRACMASS documentation. http://www.tracmass.org, http://doos.misu.su.se/tracmass/

Code Overview



TracPy serial performance

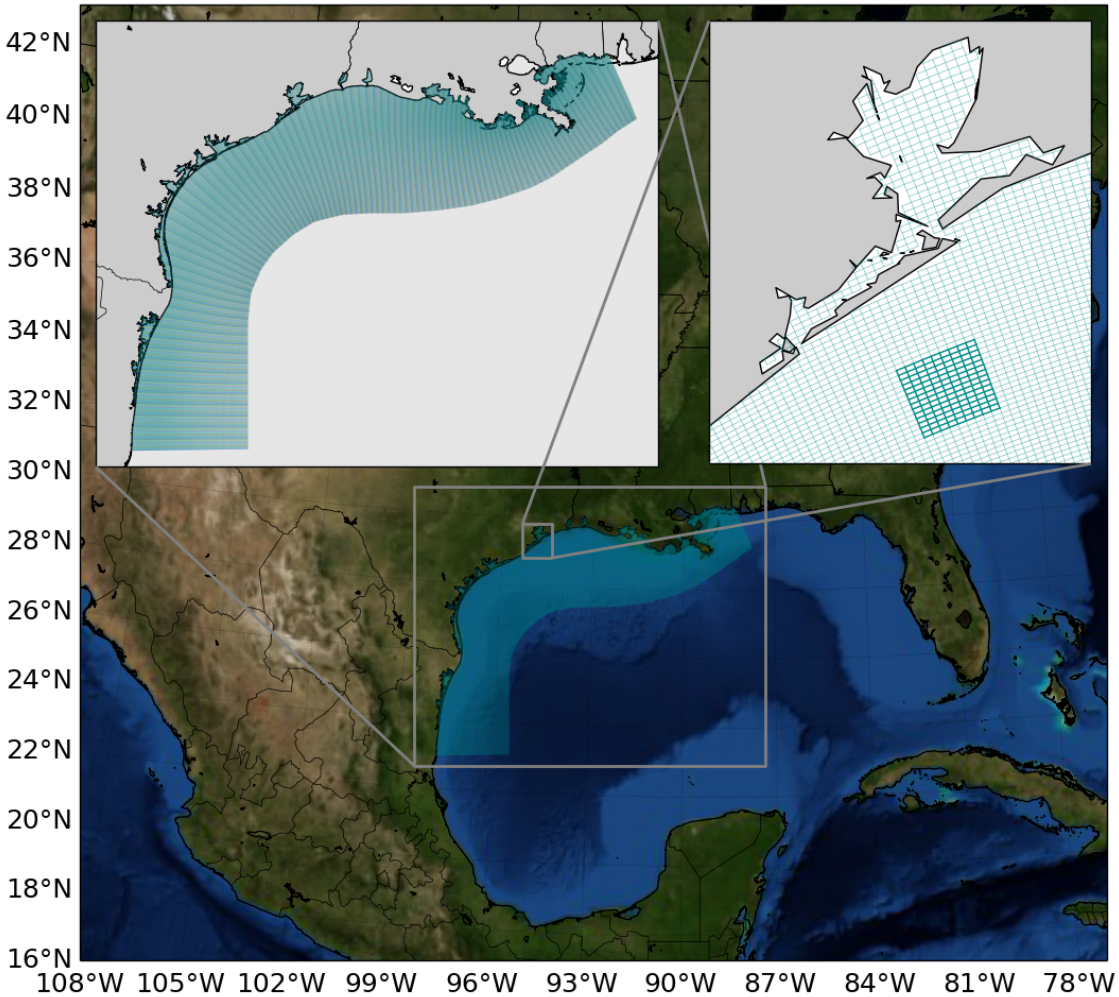


netCDF performance

	netCDF3	netCDF4C	% decrease
Simulation run time (s)	1038	1038	0
File save time (s)	3527	131	96
File size (GB)	3.6	2.1	42

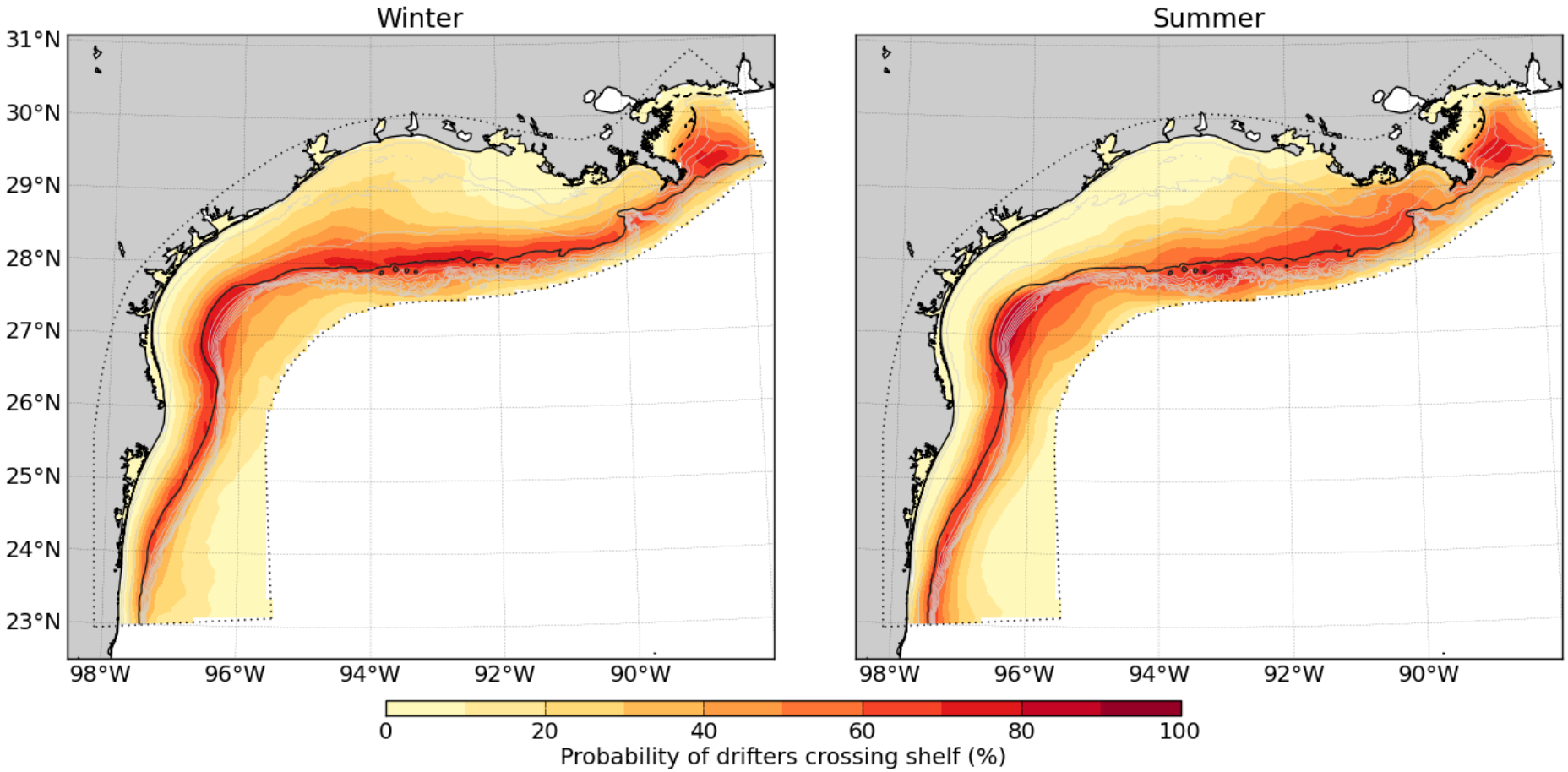
Outline

Numerical Domain

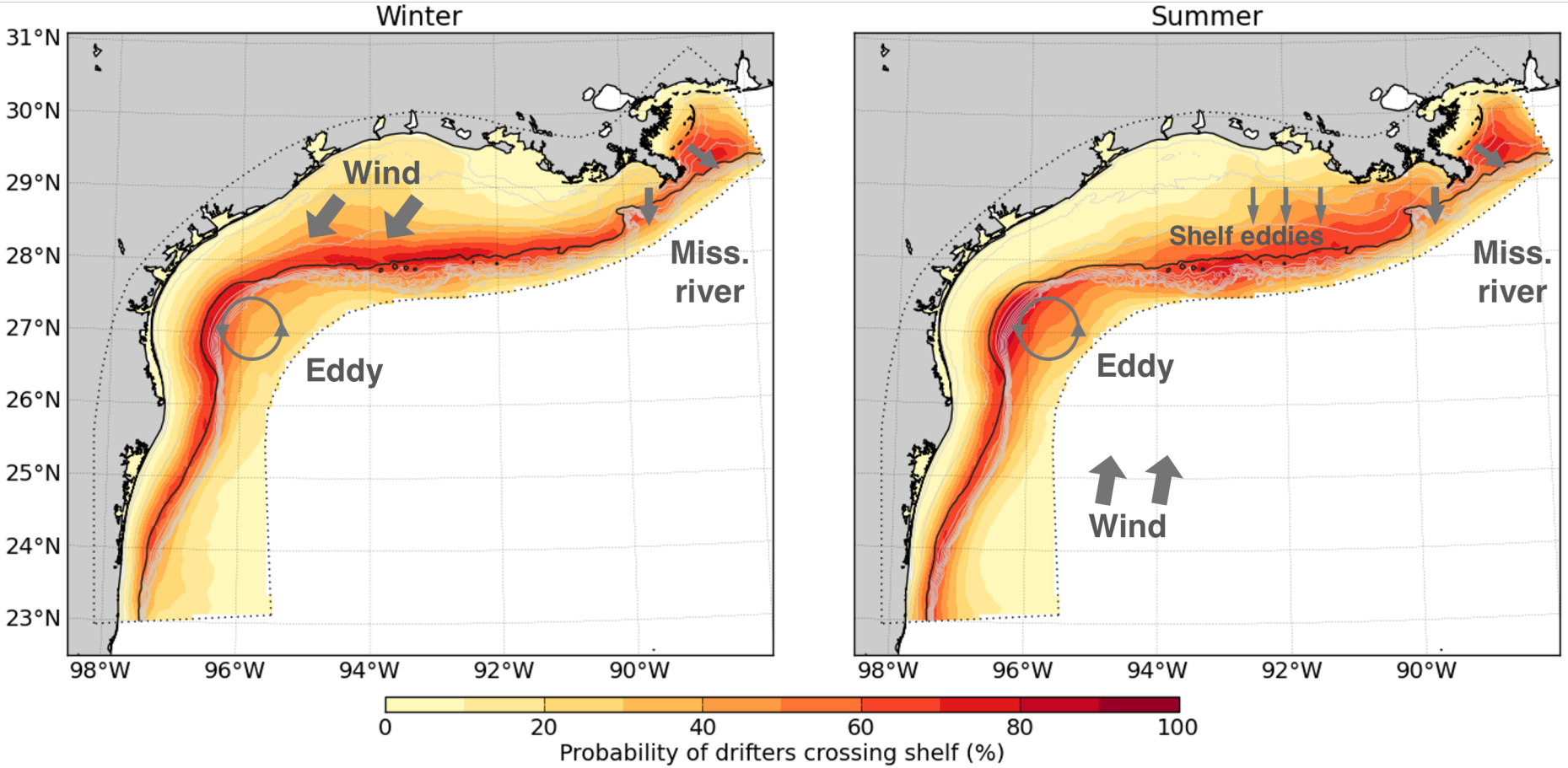


Surface Salinity

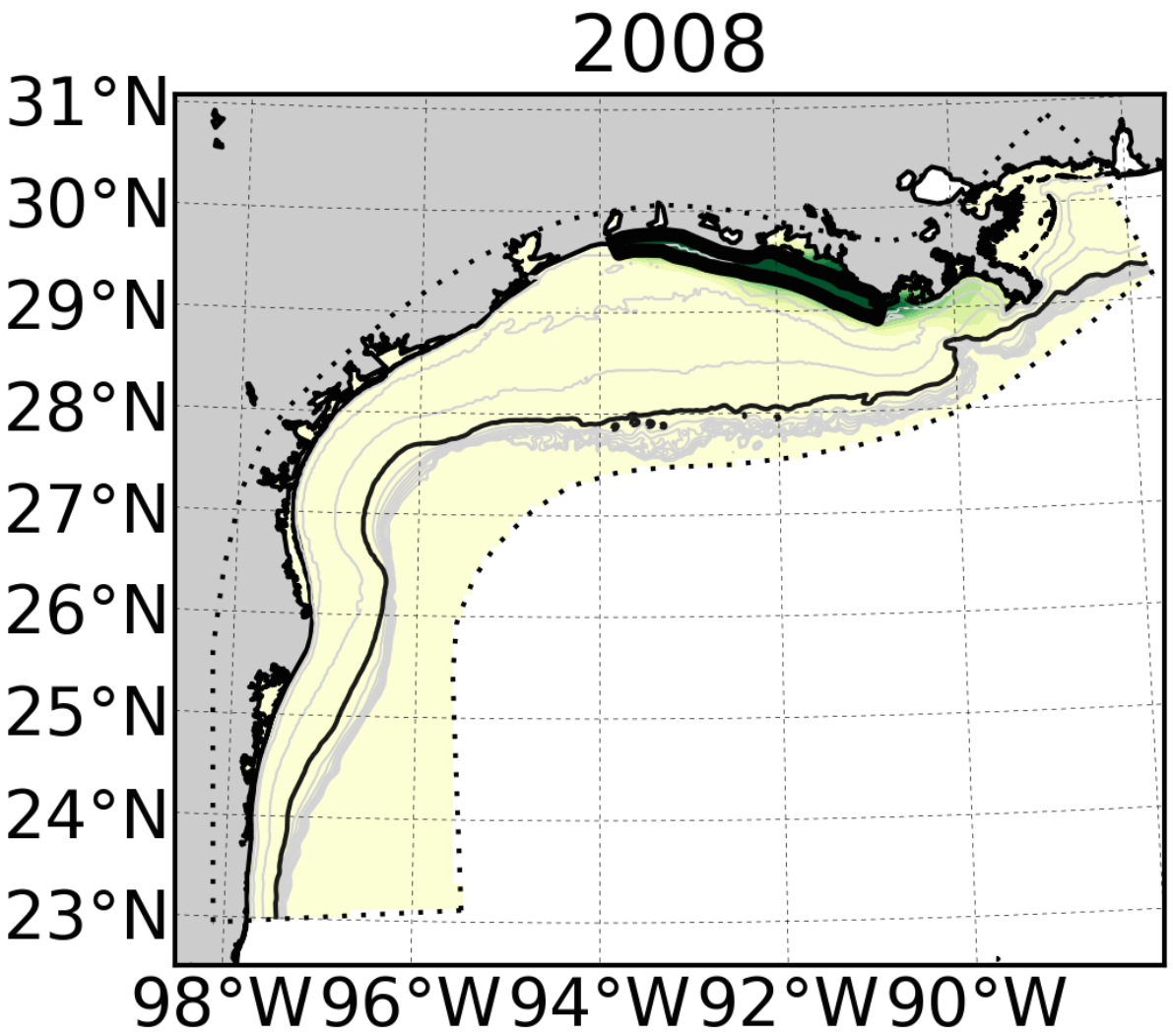
Cross-shelf transport



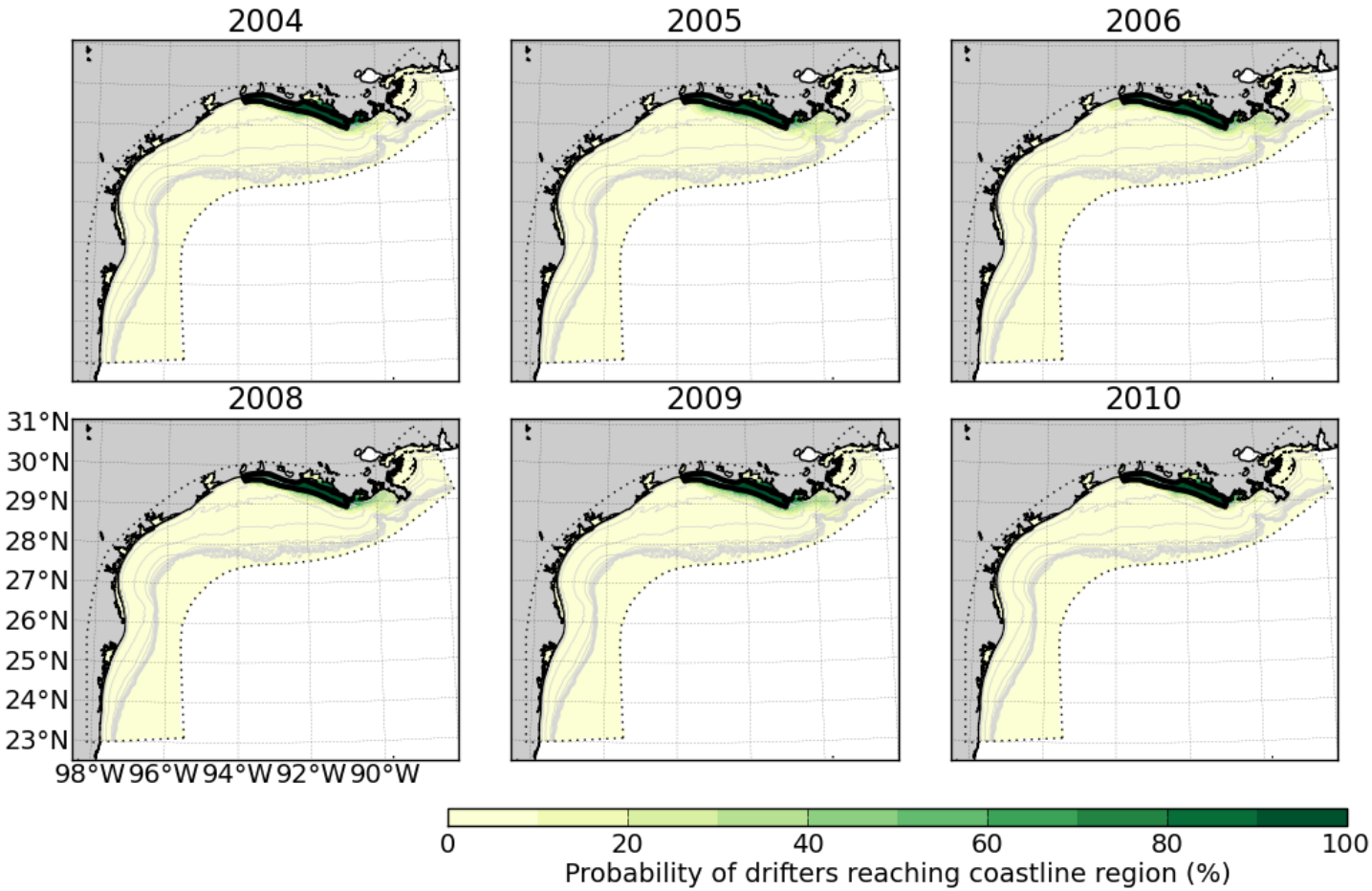
Cross-shelf transport



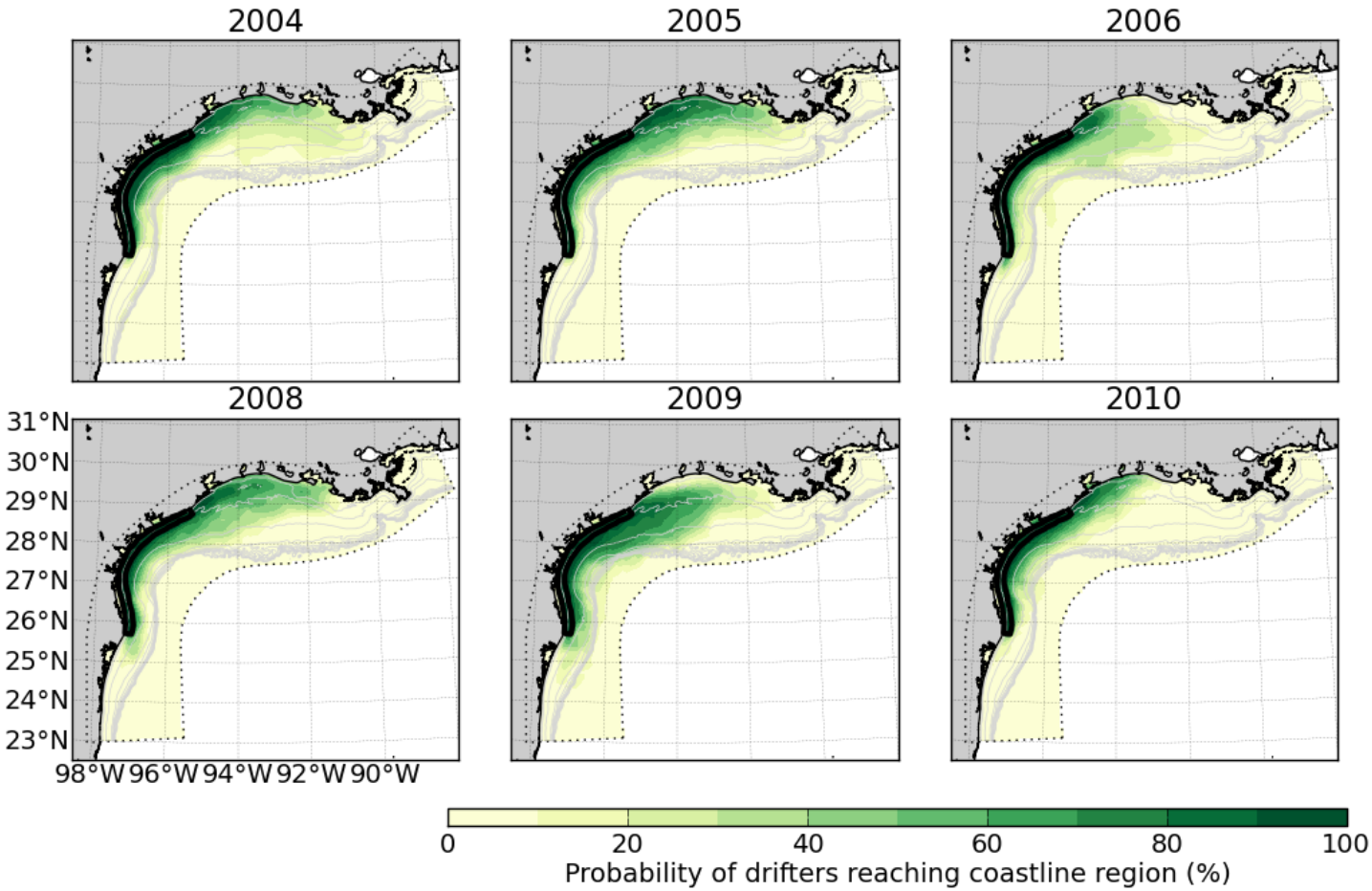
Coastline connectivity: Chenier Plain



Coastline connectivity: Chenier Plain



Coastline connectivity: South Texas



Outline

Summary

- Simulated drifter tracks improve understanding of transport processes
- Many coastal concerns are transport-based
- TracPy wraps trajectory model TRACMASS
- Code has been used for several applications so far
- Improved understanding can help address coastal issues

TracPy Manual

manual

July 6, 2014

```
In [31]: # Turning on inline plots -- just for use in ipython notebooks.
         %pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib
```

1 Initialization of a numerical experiment

Before running a drifter simulation, a number of parameters need to be specified. Previous examples of this are set in `init.py`. Because these examples change over time, we'll go through a specific example here.

```
In [32]: # Normal Python libraries
         import numpy as np
         import netCDF4 as netCDF
         import tracpy
         import tracpy.plotting
         from tracpy.tracpy_class import Tracpy
         matplotlib.rcParams.update({'font.size': 20})
```

1.1 Model output

Model output from a high resolution model of the Texas-Louisiana shelf for the years 2004-2012 is stored on a thredds served at the address in loc. This is freely accessible.

```
In [33]: # Location of TXLA model output file and grid, on a thredds server.
         loc = 'http://barataria.tamu.edu:8080/thredds/dodsC/NcML/txla_nesting6.nc'
```

1.2 Time parameters

Model output is known to occur every four hours. The default test here will start at 00:00 on November 25, 2009 and run for 5 days.

```
In [34]: # Number of days to run the drifters.
         ndays = 3

         # Start date in date time formatting
         date = datetime.datetime(2009, 11, 25, 0)

         # Time between outputs
         tseas = 4*3600 # 4 hours between outputs, in seconds

         # Time units
         time_units = 'seconds since 1970-01-01'
```

```
In [35]: # Sets a smaller limit than between model outputs for when to force interpolation if has
         nsteps = 5
         # Controls the sampling frequency of the drifter tracks.
         N = 4
```

After initialization, drifters can be stepped forward or backward in time. Running backward in time essentially means that we change the sign of the velocity fields and step backward in the model output files (in which case we set `ff=-1`). We'll move forward in time (`ff=1`).

```
In [36]: # Use ff = 1 for forward in time and ff = -1 for backward in time.
         ff = 1
```

1.3 Subgrid parameterization parameters

An integer flag is used to control whether or not to use subgrid parameterization in the particle tracking, and if so, which kind.

Options are:

- `doturb=0` uses no sub grid parameterization and thus the drifters are passively advected according strictly to the output velocity fields
- `doturb=1` adds to the current velocity fluxes parameterized turbulent velocity fluxes of the order of the current velocity fluxes
- `doturb=2` adds to the calculated new drifter location a slightly displaced drifter location that is randomly placed based on a circle around the drifter location
- `doturb=3` adds to the calculated new drifter location a slightly displaced drifter location that is randomly placed based on an ellipse of the bathymetry around the drifter location

The horizontal and vertical diffusivities are set by the user. These values may or may not be used in the experiment depending on whether a subgrid parameterization is used, and, if so, which is used. The horizontal diffusivity value is used by all of the horizontal subgrid parameterizations. The vertical diffusivity is not used in the two-dimensional case. Since this experiment is not using either diffusivity values, they will be set to zero to avoid confusion.

Appropriate values to use for this are currently being investigated using sensitivity studies on the Texas-Louisiana shelf. Some values have been used and compared in studies, and values can be calculated from physical drifters for a specific domain. This is on-going work! In a sensitivity study, a smaller value, like `ah=5`, leads to somewhat diffused results that are still very close to the non-diffusive case. A larger value of `ah=20` led to more diffused results that were still quite similar to the non-diffusive case.

```
In [37]: ah = 0. # m^2/s
         av = 0. # m^2/s

         # turbulence/diffusion flag
         doturb = 0
```

TracPy Manual

```
In [39]: # for 3d flag, do3d=0 makes the run 2d and do3d=1 makes the run 3d
         do3d = 0

         ## Choose method for vertical placement of drifters
         z0 = 's' # I know the size from checking #'s after eliminating those outside domain '
         num_layers = 30
         zpar = num_layers-1 # 29 #-10 #grid['km']-1 # 30 #grid['km']-1

         # #### 3D Sample Options ####
         # # for 3d flag, do3d=0 makes the run 2d and do3d=1 makes the run 3d
         # do3d = 1

         # ## Choose method for vertical placement of drifters
         # z0 = np.zeros(676) # I know the size from checking #'s after eliminating those outs
         # num_layers = 30
         # zpar = 'fromZeta' #num_layers-1 # 29 #-10 #grid['km']-1 # 30 #grid['km']-1
         # ####
```

1.5 Initialize TracPy class

```
In [40]: # Initialize Tracpy class
         tp = Tracpy(loc, name=name, tseas=tseas, ndays=ndays, nsteps=nsteps, usebasemap=True,
                     N=N, ff=ff, ah=ah, av=av, doturb=doturb, do3d=do3d, z0=z0, zpar=zpar, time_un

In [41]: # read in grid
         tp._readgrid()
```

1.6 Drifter initialization

1.6.1 Horizontal

Drifters are seeded by the latitude and longitude. A simple way to do this is to set up a mesh of points within a lat/lon box. In this case, we are looking at drifters starting throughout the TX-LA shelf domain. For the `linspace` function, we can play around with the number of points to control approximately how far apart the drifters begin. For this example, the number of points are about 20 km apart.

After initializing these points, we can run them through a check script to eliminate points outside the domain (without this step, points outside the numerical domain will cause an error).

```
In [42]: # Input starting locations as real space lon,lat locations
         lon0, lat0 = np.meshgrid(np.linspace(-98.5,-87.5,55), \
                                  np.linspace(22.5,31,49)) # whole domain, 20 km

         # Eliminate points that are outside domain or in masked areas
         lon0, lat0 = tracpy.tools.check_points(lon0, lat0, tp.grid)
```

TracPy Manual

2 Run the numerical experiment

```
In [43]: # Note in timing that the grid was already read in
         lonp, latp, zp, t, T0, U, V = tracpy.run.run(tp, date, lon0, lat0)

Using GCM model output index  0
Using GCM model output index  1
Using GCM model output index  2
Using GCM model output index  3
Using GCM model output index  4
Using GCM model output index  5
Using GCM model output index  6
Using GCM model output index  7
Using GCM model output index  8
Using GCM model output index  9
Using GCM model output index  10
Using GCM model output index  11
Using GCM model output index  12
Using GCM model output index  13
Using GCM model output index  14
Using GCM model output index  15
Using GCM model output index  16
Using GCM model output index  17
==========================================

Simulation name:  temp

==========================================
```



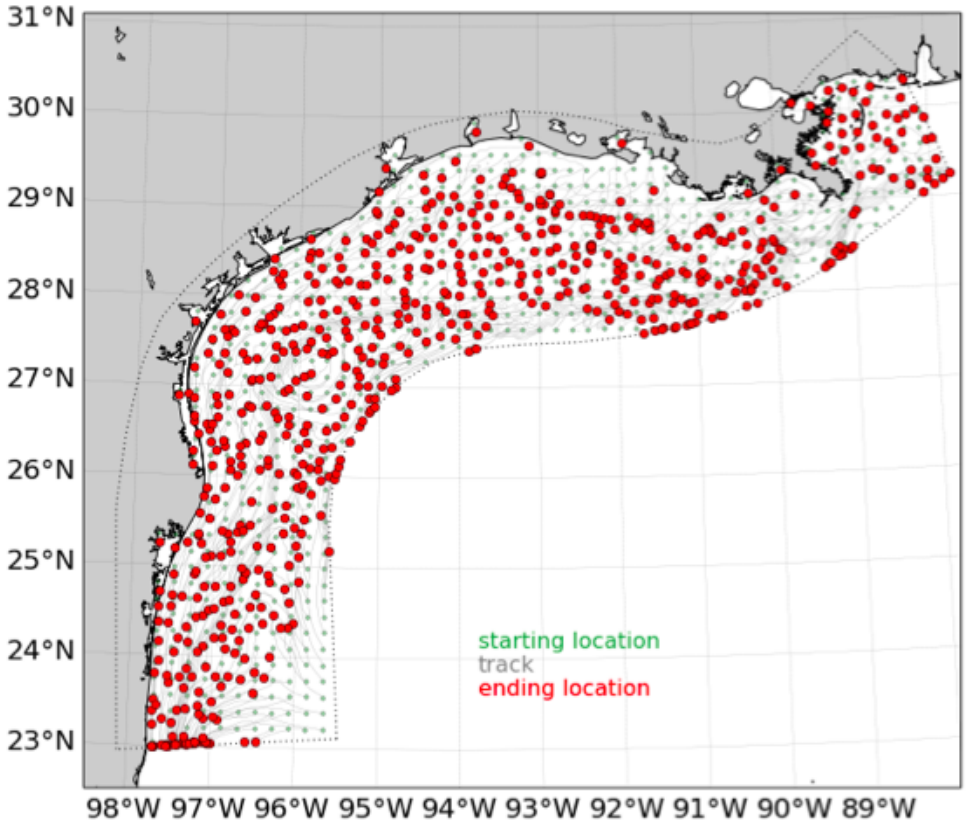
```
Total run time: 19.583090 (seconds)
------------------------------------------
Time spent on:
        1: Preparing for simulation                 1.2274 (6.2676%)
        2: Preparing for model step                 18.1805 (92.8376%)
        3: Stepping, using TRACMASS                 0.0889 (0.4538%)
        4: Processing after model step              0.0089 (0.0454%)
        5: Processing after simulation              0.0775 (0.3955%)
==========================================
```

3 Plotting the results

Plots generated below by the user can be compared with those available in `tracpy/docs/figures`.

3.1 Plot tracks

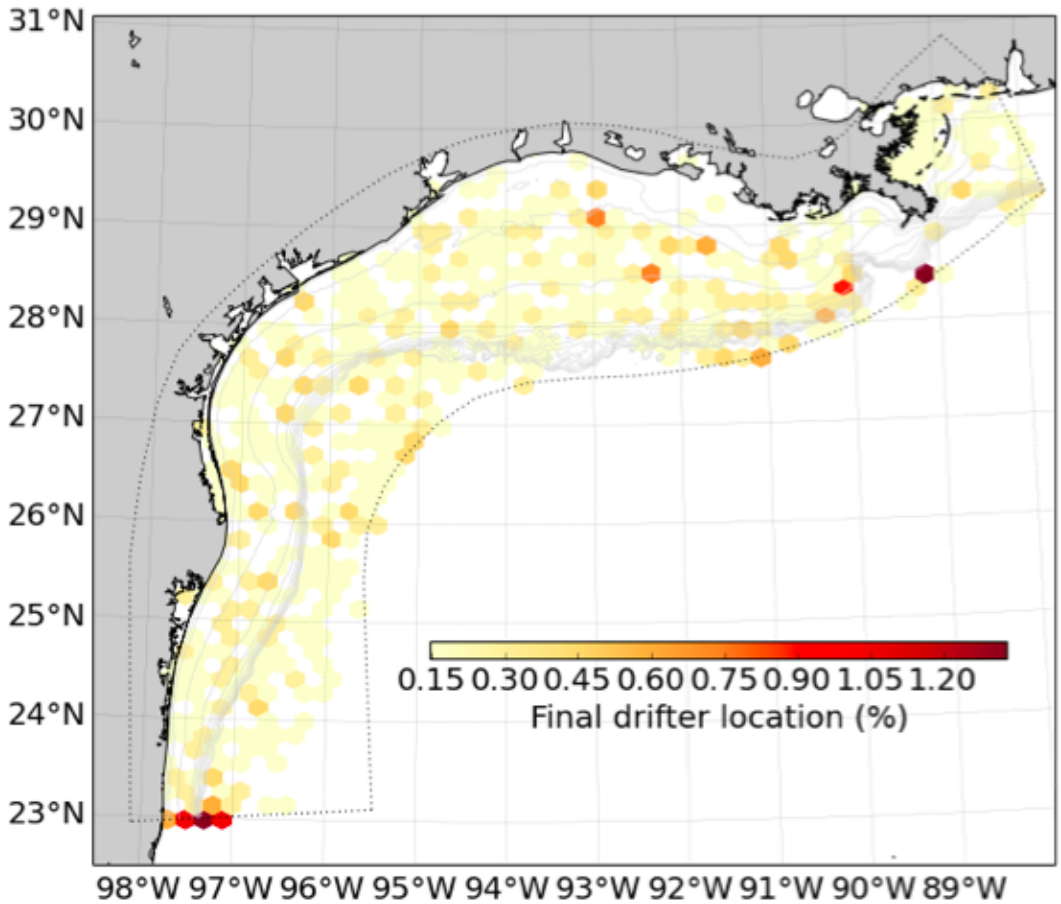
```
In [44]: tracpy.plotting.tracks(lonp, latp, tp.name, tp.grid)
```



3.2 Terminal/origin histograms

A histogram can quickly show where particles end up (for forward time case) or where particles originated (for backward time case).

```
In [45]: tracpy.plotting.hist(lonp, latp, tp.name, grid=tp.grid, which='hexbin', bins=(50,50))
```



Future work

- Integration into NOAA's GNOME oil tracking system
- Improve grid read in speed and memory required
- Append output to track file instead of storing for full run
- Possibly rearrange drifter arrangement in output file to reduce nan storage
- Storage could be updated to full netCDF4 format.
- The modularity of the TracPy class should be improved.

Thank you!

TracPy: https://github.com/kthyng/tracpy