

```

1: # code to run particle tracking for Fig 1
2:
3: from oceantracker.main import OceanTracker # load
4: ot = OceanTracker() # create an instance to build parameter dictionary= ot.params
5:
6: # add settings
7: ot.settings(output_file_base='OTpaper_exmple_A', root_output_dir='output', time_step= 600.)
8:
9: # add reader to acces the hyrdo-model
10: ot.add_class('reader', input_dir=r'oceantracker\demos\demo_hindcast',
11:             file_mask= 'demoHindcastSchism*.nc') # file mask to search for
12: # add a point release
13: ot.add_class('release_groups', name = 'my_point_release', class_name='PointRelease',
14:             points= [[1595000, 5482600, -2], [1594000, 5484200, -2]], # (x,y,z) of release points
15:             release_interval= 600, pulse_size= 5000)
16: # add polygon release at random depths between two z values
17: ot.add_class('release_groups', name = 'my_polygon_release', class_name='PolygonRelease',
18:             points=[[1597682., 5486972], [1598604, 5487275], [1598886, 5486464],
19:                    [1597917., 5484000], [1597300, 5484000], [1597682, 5486972]],
20:             release_interval= 600, pulse_size= 50, z_min= -2., z_max = 0.5)
21: # add grid releasing at random depths between two z values
22: ot.add_class('release_groups', name = 'my_grid_release', class_name='GridRelease',
23:             grid_center=[1592000, 5489200], grid_span=[500, 1000], grid_size=[3, 4],
24:             release_interval= 1800, pulse_size= 2, z_min= -2, z_max = -0.5)
25: # add a decaying particle property, # with exponential decay based on age
26: ot.add_class('particle_properties', name = 'a_pollutant', class_name='AgeDecay',
27:             initial_value= 1000, decay_time_scale = 7200.) # exponential decay time scale 2hours
28: # add a gridded particle statistic to use as heat map
29: ot.add_class('particle_statistics', name = 'my_heatmap', class_name= 'GriddedStats2D_timeBased',
30:             grid_size=[120, 121], release_group_centered_grids = True, update_interval = 600,
31:             particle_property_list = ['a_pollutant'], status_min = 'moving', z_min = -10.)
32: ot.add_class('resuspension', critical_friction_velocity=0.01) #set value for particle resuspension
33:
34: # run OT and return file name useful in plotting
35: case_info_file= ot.run()

```