Particles tracking eReefs Daily output

The OceanParcels project develops `Parcels` (*Probably A Really Computationally Efficient Lagrangian Simulator*), a set of Python classes and methods to create customisable particle tracking simulations using output from Ocean Circulation models such as the eReefs one.

Example https://tristansalles.github.io/EnviReef/6-addson/parcels.html

Load the required Python libraries

First of all, load the necessary libraries:

In [23]:
```
import os
import numpy as np
import xarray as xr

import cmocean

import cartopy
import cartopy.crs as ccrs
import cartopy.feature as cfeature
from cartopy.mpl.gridliner import LONGITUDE_FORMATTER, LATITUDE_FORMATTER
cartopy.config['data_dir'] = os.getenv('CARTOPY_DIR', cartopy.config.get('data_dir'))

from parcels import FieldSet, Field, ParticleSet, Variable, JITParticle
from parcels import AdvectionRK4, plotTrajectoriesFile, ErrorCode

import math
from datetime import timedelta
from operator import attrgetter

from matplotlib import pyplot as plt
#%config InlineBackend.figure_format = 'retina'
plt.ion()  # To trigger the interactive inline mode

#plotting shapefiles
import shapefile as shp  # Requires the pyshp package
import matplotlib.pyplot as plt

#The required SWOT Track
sf = shp.Reader("swot_calval_hr_Dec2022-v07_swath.shp")
```

Build multi-file dataset

Use the `open_mfdataset` function from `xArray` to open multiple netCDF files into a single xarray Dataset.

Load the GBR1km dataset from the NCI server, so let's first define the base URL:

In [2]:
```
# For the hydro dataset
base_url = "https://dapds00.nci.org.au/thredds/dodsC/fx3/gbr1_2.0/gbr1_simple_"
```

In [3]:
```
day_st = 14     #Starting day
day_ed = 14     #Ending day
year = 2022   # Year
month = 5

# Based on the server the file naming convention
hydrofiles = [f"{base_url}{year}-{month:02}-{day:02}.nc" for day in range(day_st, day_
```

In [ ]:
```
hydrofiles
```

Loading dataset into xArray

Using `xArray`, we open these files into a `Dataset`:

In [4]:
```
ds_hydro = xr.open_mfdataset(hydrofiles)
ds_hydro
```

Out[4]: xarray.Dataset

► Dimensions: (k: 44, j: 2389, i: 510, **time**: 24)

▼ Coordinates:

zc	(k)	float64	dask.array<chunksize=(44,), meta=np....	
longitude	(j, i)	float64	dask.array<chunksize=(2389, 510), me...	
latitude	(j, i)	float64	dask.array<chunksize=(2389, 510), me...	
time	(time)	datetime64[ns]	2022-05-13T14:00:00 ... 2022-05-...	

► Data variables: (16)

► Indexes: (1)

► Attributes: (14)

Clip the Dataset

To reduce the `Dataset` size we will clip the spatial extent based on longitudinal and latitudinal values.

This is easely done using the `sel` function with the `slice` method.

In [20]:
```
ds_hydro_clip = ds_hydro.drop(['dhw0','dhw1','dhw2','dhw3','RT_expose','swr','temp_exp

min_lon = 146     # lower left longitude
min_lat = -21     # lower left latitude
max_lon = 149     # upper right longitude
max_lat = -18     # upper right latitude
#min_depth = 42    # -0.5
#max_depth = 43    # 0.5
```

```
# Defining the boundaries
lon_bnds = [min_lon, max_lon]
lat_bnds = [min_lat, max_lat]
#depth_bnds = [min_depth, max_depth]

# Performing the reduction and only taking the surface dataset (k=-1)
#ds_hydro_clip = ds_hydro_clip.sel(latitude=slice(*lat_bnds), longitude=slice(*lon_bnd
ds_hydro_clip = ds_hydro_clip.sel(j=slice(1000,1500), k=42) #monthly data
ds_hydro_clip
```

Out[20]: xarray.Dataset

► Dimensions: (j: 500, i: 510, **time**: 24)

▼ Coordinates:

zc	()	float64	dask.array<chunksize=(), meta=np....
longitude	(j, i)	float64	dask.array<chunksize=(500, 510), ...
latitude	(j, i)	float64	dask.array<chunksize=(500, 510), ...
time	(time)	datetime64[ns]	2022-05-13T14:00:00 ... 2022-05-...

▼ Data variables:

botz	(j, i)	float64	dask.array<chunksize=(500, 510), ...
eta	(time, j, i)	float32	dask.array<chunksize=(24, 500, 51...
u	(time, j, i)	float32	dask.array<chunksize=(24, 500, 51...
v	(time, j, i)	float32	dask.array<chunksize=(24, 500, 51...
w	(time, j, i)	float32	dask.array<chunksize=(24, 500, 51...
temp	(time, j, i)	float32	dask.array<chunksize=(24, 500, 51...

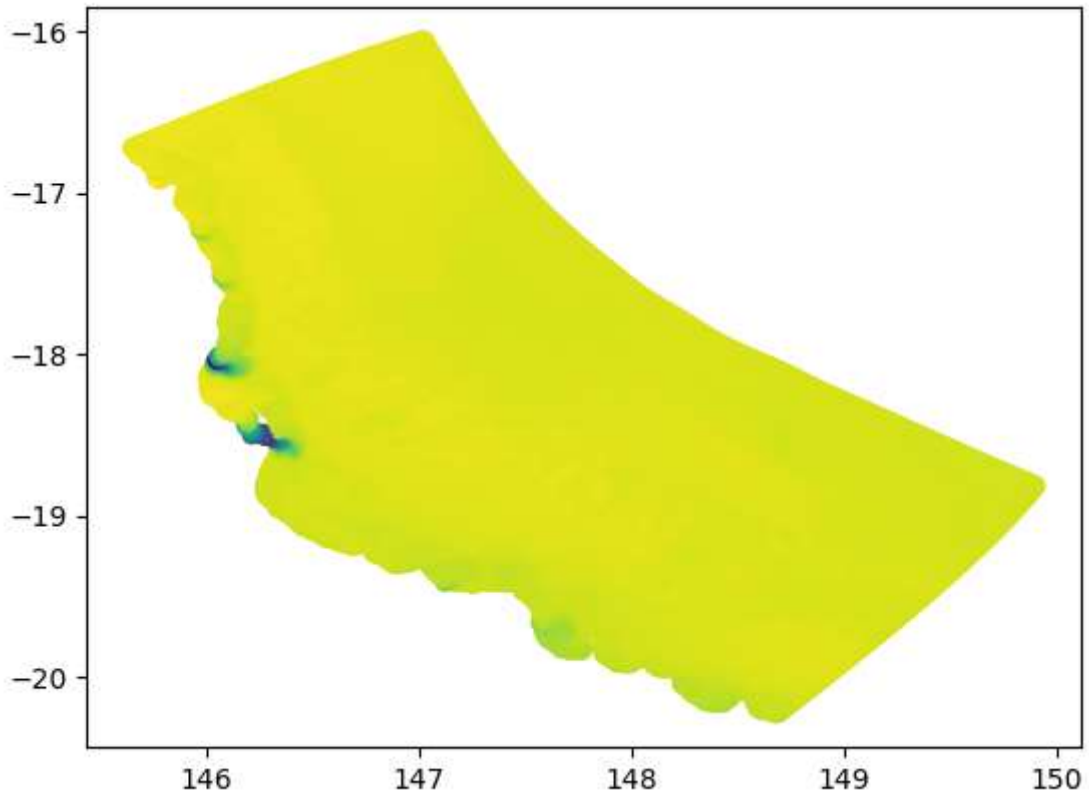
► Indexes: (1)

► Attributes: (14)

In [8]:
```
data_name ='ereefdata_1gbr_May_2022.nc'
try:
    os.remove(data_name)
except OSError:
    pass

ds_hydro_clip.to_netcdf(path=data_name)
```

In [21]:
```
#Testing the domain

plt.scatter(ds_hydro_clip.longitude, ds_hydro_clip.latitude, c = ds_hydro_clip.temp[0,
```

Out[21]: <matplotlib.collections.PathCollection at 0x12e05b48d90>

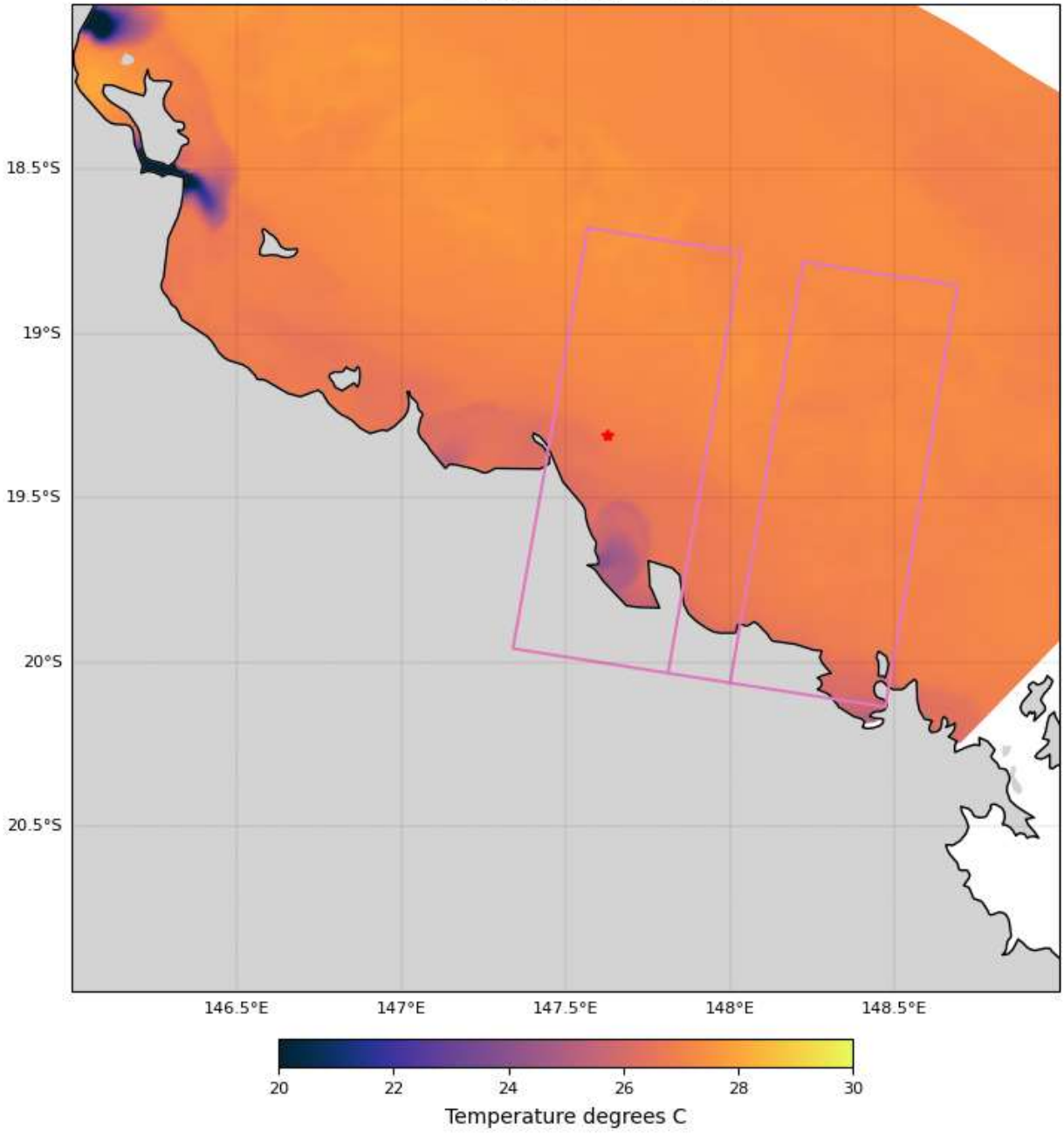


In [37]:

```
#Time
i = 10

# Figure size
size = (9, 10)

# Color from cmocean
color = cmocean.cm.thermal

# Defining the figure
fig = plt.figure(figsize=size, facecolor='w', edgecolor='k')

# Axes with Cartopy projection
ax = plt.axes(projection=ccrs.PlateCarree())
# and extent
ax.set_extent([min_lon, max_lon, min_lat, max_lat], ccrs.PlateCarree())

# Plotting using Matplotlib
# We plot the PH at the surface at the final recorded time interval
#cf = ds_hydro_clip.temp.isel(time=i).plot(transform=ccrs.PlateCarree())
cf = plt.scatter(ds_hydro_clip.longitude, ds_hydro_clip.latitude, c = ds_hydro_clip.te

# Color bar
cbar = fig.colorbar(cf, ax=ax, fraction=0.027, pad=0.045, orientation="horizontal")
cbar.set_label(ds_hydro_clip.temp.long_name+' '+ds_hydro_clip.temp.units, rotation=0,
cbar.ax.tick_params(labelsize=8)


#Plotting the SWOT Tracks
for shape in sf.shapeRecords():
    x = [i[0] for i in shape.shape.points[:]]
    y = [i[1] for i in shape.shape.points[:]]
    plt.plot(x,y)
```

```
# Title
plt.title('Parcels evolution: '+ str(ds_hydro_clip.time.isel(time=i)), fontsize=13)

# Plot lat/lon grid
gl = ax.gridlines(crs=ccrs.PlateCarree(), draw_labels=True, linewidth=0.1, color='k',
gl.top_labels = False
gl.right_labels = False
gl.xformatter = LONGITUDE_FORMATTER
gl.yformatter = LATITUDE_FORMATTER
gl.xlabel_style = {'size': 8}
gl.ylabel_style = {'size': 8}

# Add map features with Cartopy
ax.add_feature(cfeature.NaturalEarthFeature('physical', 'land', '10m', edgecolor='face
ax.coastlines(linewidth=1)

#for k in range(parcels.lon.shape[0]):
    #ax.scatter(parcels.lon.isel(traj=k), parcels.lat.isel(traj=k), s=40, edgecolors=
                #linewidth=0.2, transform=ccrs.PlateCarree()).set_zorder(11)

Yongala_LONLAT = [147.625,-19.308333];
plt.plot(Yongala_LONLAT[0],Yongala_LONLAT[1],'r*')

plt.tight_layout()
plt.show()
fig.clear()
plt.close(fig)
plt.clf()
```

Parcels evolution: <xarray.DataArray 'time' ()>
array('2022-05-14T00:00:00.000000000', dtype='datetime64[ns]')
Coordinates:
zc float64 dask.array<chunksize=(), meta=np.ndarray>
time datetime64[ns] 2022-05-14
Attributes:
long_name: Time
standard_name: time
coordinate_type: time
puv__uom: http://vocab.nerc.ac.uk/collection/P06/current/UTAA/
_ChunkSizes: 512



<Figure size 640x480 with 0 Axes>

Sampling a Field with Particles

One typical use case of particle simulations is to sample a Field (such as temperature, salinity, surface hight) along a particle trajectory. In `Parcels`, this is very easy to do, with a `custom Kernel`.

Let's define the `FieldSet` as above:

In [38]:
```
filenames = {'U': data_name,
             'V': data_name}
```

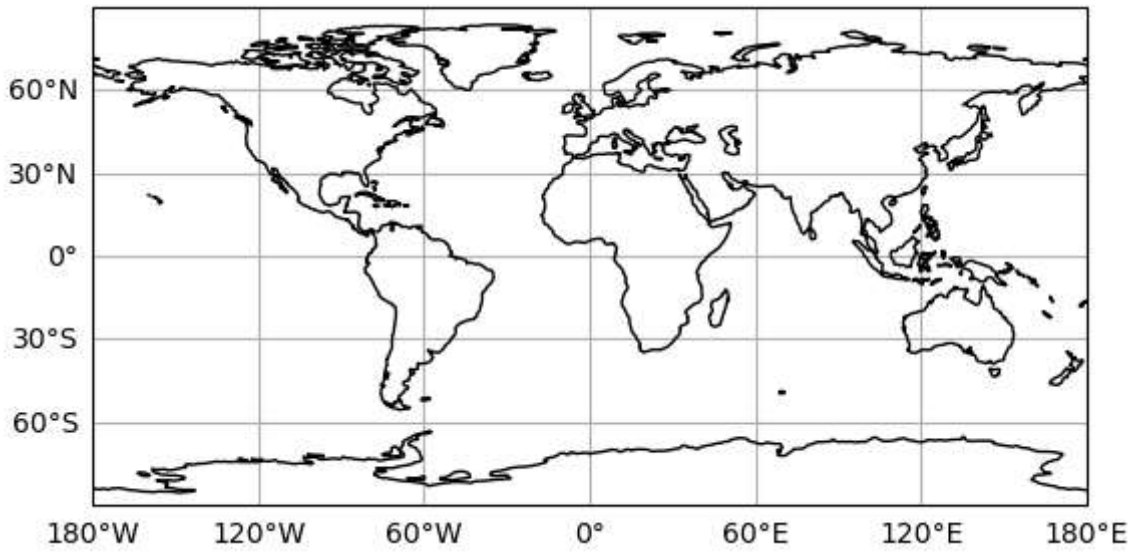
In [53]:
```
filenames = {'U': {'lon': 'mesh_mask.nc4',
                   'lat': 'mesh_mask.nc4',
                   'data': data_name},
             'V': {'lon': 'mesh_mask.nc4',
                   'lat': 'mesh_mask.nc4',
                   'data': data_name}}
variables = {'U': 'u',
             'V': 'v'}
dimensions = {'lon': 'glamf', 'lat': 'gphif', 'time': 'time'}
field_set = FieldSet.from_nemo(filenames, variables, dimensions, allow_time_extrapolat
```

In [54]:
```
field_set = FieldSet.from_nemo(filenames, variables, dimensions, allow_time_extrapolat
```

In [55]:
```
field_set.U.show()
```

```
WARNING: Field.show() does not always correctly determine the domain for curvilinear
grids. Use plotting with caution and perhaps use domain argument as in the NEMO 3D tu
torial
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[55], line 1
----> 1 field_set.U.show()

File ~\anaconda3\lib\site-packages\parcels\field.py:1265, in Field.show(self, animation, show_time, domain, depth_level, projection, land, vmin, vmax, savefile, **kwargs)
   1252 """Method to 'show' a Parcels Field
   1253
   1254 :param animation: Boolean whether result is a single plot, or an animation
   (...)
   1262 :param savefile: Name of a file to save the plot to
   1263 """
   1264 from parcels.plotting import plotfield
-> 1265 plt, _, _, _ = plotfield(self, animation=animation, show_time=show_time, domain=domain, depth_level=depth_level,
   1266                          projection=projection, land=land, vmin=vmin, vmax=vmax, savefile=savefile, **kwargs)
   1267 if plt:
   1268     plt.show()

File ~\anaconda3\lib\site-packages\parcels\plotting.py:143, in plotfield(field, show_time, domain, depth_level, projection, land, vmin, vmax, savefile, **kwargs)
    141 show_time = fld.grid.time[0] if show_time is None else show_time
    142 if fld.grid.defer_load:
--> 143     fld.fieldset.computeTimeChunk(show_time, 1)
    144 (idx, periods) = fld.time_index(show_time)
    145 show_time -= periods * (fld.grid.time_full[-1] - fld.grid.time_full[0])

File ~\anaconda3\lib\site-packages\parcels\fieldset.py:1052, in FieldSet.computeTimeChunk(self, time, dt)
   1050             fb.close()
   1051         fb = None
-> 1052     data = f.computeTimeChunk(data, tind)
   1053 data = f.rescale_and_set_minmax(data)
   1055 if (isinstance(f.data, DeferredArray)):

File ~\anaconda3\lib\site-packages\parcels\field.py:1412, in Field.computeTimeChunk(self, data, tindex)
   1410 elif len(buffer_data.shape) == 3:
   1411     buffer_data = lib.reshape(buffer_data, sum(((buffer_data.shape[0], 1, ), buffer_data.shape[1:]), ()))
-> 1412 data = self.data_concatenate(data, buffer_data, tindex)
   1413 self.filebuffers[tindex] = filebuffer
   1414 return data

File ~\anaconda3\lib\site-packages\parcels\field.py:1371, in Field.data_concatenate(self, data, data_to_concat, tindex)
   1369 lib = np if isinstance(data, np.ndarray) else da
   1370 if tindex == 0:
-> 1371     data = lib.concatenate([data_to_concat, data[tindex+1:, :]], axis=0)
   1372 elif tindex == 1:
   1373     data = lib.concatenate([data[:tindex, :], data_to_concat], axis=0)

File <__array_function__ internals>:180, in concatenate(*args, **kwargs)

ValueError: all the input array dimensions for the concatenation axis must match exactly, but along dimension 2, the array at index 0 has size 2389 and the array at index 1 has size 1021
```



In [47]:
```
def DeleteParticle(particle, fieldset, time):
    particle.delete()
```

In [48]:
```
npart = 10  # number of particles to be released
time = np.arange(0,npart)*timedelta(minutes=40).total_seconds()  # release every parti
```

In [49]:
```
pset = ParticleSet.from_line(fieldset=fieldset,
                             pclass=JITParticle,
                             size=npart,             # releasing X particles
                             start=(147.5, -19.5),   # releasing on a line: the start
                             finish=(148.0, -19.0),
                             time=time)  # release every particle X hour
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[49], line 3
      1 #Line Plotting
----> 3 pset = ParticleSet.from_line(fieldset=fieldset,
      4                              pclass=JITParticle,
      5                              size=npart,             # releasing X particles
      6                              start=(147.5, -19.5),   # releasing on a line: t
he start longitude and latitude
      7                              finish=(148.0, -19.0),
      8                              time=time)

NameError: name 'fieldset' is not defined
```