""" read a set of HYCOM outputs from HMON/HWRF/HAFS, and

estimate footprint values (at the 34-kt radius) for the ocean skill metrics -

var=OHC

Z26

MLD\_temp\_crit

MLT\_temp\_crit

MLD\_dens\_crit

MLT\_dens\_crit

T100

PEA

and

returns a set of plots of a var.

-------------

prerequisite:

-------------

run archv3nc to convert HYCOM [ab] to .nc

--------

history:

--------

edited by hsk 5/11/2021 """

################################################################################

#%% Frist, user inputs for a storm of interest, tcID, and cycle

EXPT='HMON'

storm='nine'

tcid='09l'

cycle='2020072918'

var='OHC'

i2pickl = True

# Load Python modules and libraries

import xarray as xr

import numpy as np

from datetime import datetime, timedelta

import seawater as sw

from utils4HWRF import readTrack6hrly, Rwinds6hr

from geoutils import haversine

import os, sys, glob

import pandas as pd

import matplotlib.pylab as plt

from utils import coast180, hsk\_Spectral\_r

from pathlib import Path

#########################################################

##%%% graphdir

graphdir=os.getcwd()+'/output/'+EXPT+'/'+cycle+'/'

if not os.path.isdir(graphdir):

p=Path(graphdir)

p.mkdir(parents=True)

# Find atcf coincide with a cycle of interest

# Adeck

rundir='/work/noaa/hwrf/save/hskim/Isaias/'+EXPT+'/'+cycle+'/'+tcid.upper()

aid = rundir+'/'+storm.lower()+tcid.lower()+'.'+cycle+'.trak.'+EXPT.lower()+'.atcfunix'

adn,aln,alt,apmn,avmx=readTrack6hrly(aid)

aln=-1\*aln+360

cx,cy=coast180()

cx=cx+360

#%% 6-hourly 1/12-degree POM netcdf files

infiles = sorted(glob.glob(rundir+'/\*rtofs\_hat10\*.nc'))

ncf = xr.open\_mfdataset(infiles)

ncfx,ncfy = np.meshgrid(ncf.Longitude.data,ncf.Latitude.data)

adum=np.ones(ncfx.shape)\*np.nan

ttm = ncf.MT.data

#---------- building dataframe for each footprint

#names=['YMDH','lon','lat','SST','SSS','OHC','Z26','MLDTc','MLTTc','MLDdc','MLTdc','PEA']

#c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,c13=[],[],[],[],[],[],[],[],[],[],[],[],[]

ac1,ac2,ac3,ac4,ac5,ac6,ac7=[],[],[],[],[],[],[]

names=['YMDH','lon','lat','SST','SSS','OHC']

DF=pd.DataFrame()

knt=0

plt.close('all')

for N,D in enumerate(adn):

dfc=pd.DataFrame()

dR=haversine(ncfx,ncfy,aln[N],alt[N])/1000.

ymdh=datetime.strftime(D,'%Y%m%d%H')

# find a set of radii for the 34-, 50- and 64-kt wind, and apply the 34-kt wind radius as for a footprint

brads=Rwinds6hr(aid,N\*6)

if brads[0] != 0:

dR[dR>brads[0]]=np.nan

# Find the lat and lon index in the HYCOM grid

ii=np.where(~np.isnan(dR))

i0=np.unique(ii[0])

i1=np.unique(ii[1])

ni0=len(i0)

ni1=len(i1)

# Footprint (x,y)

lns,lts=np.meshgrid(ncfx[0,i1],ncfy[i0,0])

one=dR/dR

oneq=np.nan\*np.ones([ni0,ni1])

for r,R in enumerate(i0):

for c,C in enumerate(i1):

oneq[r,c]=one[R,C]

#------ each forecast lead time @6 hourly

################################################################################

#%% estimate var (OHC.Z26,MLD\_temp\_cr,MLT\_temp\_cr,MLD\_dens\_cr,MLT\_dens\_cr,T100,PEA)

# at each grid cell on a footprint

# mixed depth by a temperature criteria

dtemp = 0.2

ref\_depth = 10 # meters

# mixed depth by a density criteria

drho = 0.125

ref\_depth = 10 # meters

Ta=np.array(np.squeeze(ncf.temperature[N,0,i0,i1].data))

Sa=np.array(np.squeeze(ncf.salinity[N,0,i0,i1].data))

OHC=np.array(np.squeeze(ncf.ocean\_heat\_content[N,i0,i1].data))

OHC=OHC\*oneq

ac1.extend(np.tile(D,[ni0\*ni1,1]).flatten())

ac2.extend(lns.flatten())

ac3.extend(lts.flatten())

ac4.extend(Ta.flatten())

ac5.extend(Sa.flatten())

ac6.extend(OHC.flatten())

if np.mod(knt,9)==0:

plt.figure(figsize=(13,10))

pn=plt.gcf().number

plt.subplot(3,3,1)

plt.contourf(lns,lts,OHC\*oneq,levels=np.arange(0,160,10),extend='max',cmap=hsk\_Spectral\_r()); plt.colorbar()

plt.title('(1) '+ymdh)

plt.plot(aln,alt,'-ko',markersize=3); plt.plot(cx,cy,'gray',alpha=0.6)

plt.axis([lns.min(),lns.max(),lts.min(),lts.max()])

#plt.text(lns.mean()+(lns.max()-lns.min())/4,lts.min()+0.08,"%8.2f"%(np.nansum(OHC)/ni0/ni1),fontsize=10)

plt.text(lns.mean()+(lns.max()-lns.min())/4,lts.min()+0.08,"%8.2f"%(np.nanmean(OHC)),fontsize=10)

else:

plt.figure(pn)

h=np.mod(knt,9)+1

print("knt & h="+str(knt)+', '+str(h))

plt.subplot(3,3,h)

plt.contourf(lns,lts,OHC\*oneq,levels=np.arange(0,160,10),extend='max',cmap=hsk\_Spectral\_r()); plt.colorbar()

plt.title('('+str(h)+') '+ymdh)

plt.plot(aln,alt,'-ko',markersize=3); plt.plot(cx,cy,'gray',alpha=0.6)

plt.axis([lns.min(),lns.max(),lts.min(),lts.max()])

#plt.text(lns.mean()+(lns.max()-lns.min())/4,lts.min()+0.08,"%8.2f"%(np.nansum(OHC)/ni0/ni1),fontsize=10)

plt.text(lns.mean()+(lns.max()-lns.min())/4,lts.min()+0.08,"%8.2f"%(np.nanmean(OHC)),fontsize=10)

plt.suptitle(EXPT+': \n'+storm.upper()+tcid.upper()+' OHC [kJcm$^{-2}$]',fontsize=13)

knt+=1

print('OHC estimates from '+EXPT+'a footprint done for forecast hour ',str(N\*6))

# dataFrame

dict={'YMDH':ac1, 'lon [E]':ac2, 'lat [N]':ac3, 'SST [C]':ac4, 'SSS [psu]':ac5, 'OHC [kJ/cm^2]':ac6}

dfc = pd.DataFrame.from\_dict(dict, orient='columns')

DF=DF.append(dfc)

for v in np.arange(1,pn,1):

plt.figure(num=v)

plt.savefig(graphdir+'/hmon\_'+storm+tcid+'\_'+cycle+'\_OHC\_fig'+str(v)+'.png',bbox\_inches='tight')

# --- option to save

if i2pickl:

# DataFrame

DF.to\_pickle(graphdir+'/'+storm.upper()+tcid.upper()+'\_OHC.pkl')

print('save data to '+graphdir+'/'+storm.upper()+tcid.upper()+'\_OHC.pkl')

######################################################################################