```python
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
#import netCDF4
import xarray as xr
import datetime
from dateutil.relativedelta import relativedelta

import cartopy.crs as ccrs
import cartopy.feature as cfeature
import matplotlib.ticker as mticker
import matplotlib.style as mplstyle
from cartopy.mpl.ticker import LongitudeFormatter, LatitudeFormatter

import gsw

year_bgn,month_bgn=2022,1
year_end,month_end=2022,1

dt    =datetime.datetime(year_bgn,month_bgn,15)
dt_end=datetime.datetime(year_end,month_end,15)

while True: # 月でループ

    # read MOAA GPV data
    # データを読み込む

    date = ???
    dtyp = ???
    lat = ???
    lon = ???
    prs = ???
    toi = ???
    soi = ???
    ncr.close()

    # 配列形状を fortran 風に変更
    toi = toi.T
    soi = soi.T

    im = int(lon.shape[0]) # 東西格子数
    jm = int(lat.shape[0]) # 南北格子数
    km = int(prs.shape[0]) # 鉛直格子数

    mask = np.ones([im,jm,km]) # 欠損値処理のためのマスク配列　1 を代入
    mask[toi.values < -100.]=0. # 欠損値の格子には 0 を代入
    mask[soi.values < -100.]=0. # 欠損値の格子には 0 を代入

    mask4u=mask[:,0:jm-1,:]*mask[:,1:jm,:] # 東西地衡流のためのマスク配列
    mask4v=mask*np.roll(mask,-1,axis=0) # 南北地衡流のためのマスク配列

    # ループを避けるために 3 次元配列を用意する
    LON_3D = np.zeros([im,jm,km])
    for i in range(im):
        LON_3D[i,:,:]=lon.values[i]
    LAT_3D = np.zeros([im,jm,km])
    for j in range(jm):
        LAT_3D[:,j,:]=lat.values[j]
    PRS_3D = np.zeros([im,jm,km])
    for k in range(km):
        PRS_3D[:,:,k]=prs.values[k]

    # 絶対塩分、保存温度の計算
    SA = gsw.SA_from_SP(soi.values,PRS_3D,LON_3D,LAT_3D)
    CT = gsw.CT_from_t(SA,toi.values,PRS_3D)

```

```
    # 欠損値を０としておく
    SA = SA*mask
    CT = CT*mask

    DH=np.zeros([im,jm,km]) # 力学高度の配列準備
    GV=np.zeros([im,jm,km]) # 南北地衡流の配列準備
    GU=np.zeros([im,jm-1,km]) # 東西地衡流の配列準備
    PRS_ref=0. # 基準圧力（0dbとしておく）


    # 力学高度の計算
    SAh=SA.reshape(im*jm,km).T # 絶対塩分配列を[鉛直、水平]に変換
    CTh=CT.reshape(im*jm,km).T # 保存温度配列を[鉛直、水平]に変換
    DH = gsw.geo_strf_dyn_height(???) # 絶対塩分、保存温度から力学高度を計算
    DH=DH.T # [水平、鉛直] に変換
    DH=DH.reshape(im,jm,km)*mask # [経度、緯度、鉛直]の３次元配列に変換


    # meridional geostrophic velocity
    LONxz=np.append(lon.values[:],lon.values[0]) # 南北地衡流のための経度配列
    LATxz=np.zeros(im+1) # 南北地衡流のための緯度配列
    for j in range(jm):
        DHxz=np.append(DH[:,j,:],DH[0,j,:]).reshape(im+1,km) # 南北地衡流のための力学高度配列
        LATxz[:]=lat.values[j]
        GVxz = gsw.geostrophic_velocity(???) # 力学高度から地衡流計算
        GVxz = np.array(GVxz[0])
        GVxz = GVxz.T
        GV[:,j,:] = GVxz[:,:]*mask4v[:,j,:]

    # zonal geostrophic velocity
    LONyz=np.zeros(jm) # 東西地衡流のための経度配列
    for i in range(im):
        DHyz=DH[i,:,:].T # 東西地衡流のための力学高度配列
        LONyz[:]=lon.values[i] # 東西地衡流のための経度配列
        GUyz = gsw.geostrophic_velocity(???) # 力学高度から地衡流計算
        GUyz = np.array(GUyz[0])
        GUyz = GUyz.T
        GU[i,:,:] = GUyz[:,:]*mask4u[i,:,:]

    # 最下層の流速を差し引く（南北流速の場合）
    for i in range(im) :
        for j in range(jm) :
            if mask4v[i,j,0] == 0. :
                continue # 海面が欠測なら（全水深欠測なので）飛ばす
            for k in range(km) :
                if mask4v[i,j,k] == 0. :
                    break # 欠測の深度にきたら抜ける
            if k == km-1 and mask4v[i,j,km-1] == 1. :
                kbot=k # 欠測がない場合
            else :
                kbot=k-1 # 欠速がある場合

            GV[i,j,0:kbot]=GV[i,j,0:kbot]-GV[i,j,kbot]
            GV[i,j,kbot:]=0.
    GV=GV*mask4v

    if dt == dt_end :
        break

    dt=dt+relativedelta(months=1)

```