In [ ]:

```python
import h5py

import numpy as np
import scipy
import matplotlib.pyplot as plt
from scipy import signal
import pandas as pd
import scipy.sparse as sparse
import os

indir_name =  "C:/CSES/file/"

outdir_name = "C:/CSES/plot/"

ext = ('.h5')


def getData(data):
    Bw = 51200 * 2 / 1024
    data = pd.DataFrame(data)

    matrix = []
    for i in range(len(data)):
        matrix.append(data.iloc[i])
    matrix = np.array(matrix)
    data_t = np.empty(shape=(matrix.shape[0], matrix.shape[1]))
    for i in range(0, matrix.shape[0]):
        meanX_b = np.mean(matrix[i])
        data_t[i] = matrix[i] - meanX_b

    M_b = data_t.shape[1]
    hamming_b = signal.get_window("hamming", M_b)
    FFT_low = np.array([scipy.fft.fft(data_t[i] * hamming_b) for i in range(0, data_t.shape[0])])
    out = np.abs(FFT_low.T[:1024]) ** 2
    outX_b = 400 + 20 * np.log10(out / Bw)

    return outX_b


def powerSpectrum(pow):
    powerX = 400 + 20 * np.log10(pow)

    return powerX

def tempArray(data):
    temp_df_x = []
    for i in range(len(df)):
        if (df.iloc[i].WORKMODE == 2):
            temp_df_x.append(df[data].iloc[i])
        else:
            temp_df_x.append(np.empty(np.array(df[data].iloc[i]).shape))
            temp_df_x[i][:] = np.NaN
    temp_df_x = np.array(temp_df_x)
    return temp_df_x



def Amplitude(arr):

    mask = ~np.isnan(arr[row])
    dataX = arr[row][mask]
    for i in range(0, arr.shape[0]):
        if i != row:
            mask = ~np.isnan(arr[i])
            arr[i][mask] = np.nan
    return arr

def dataframeBurstBuild(arrX,arrY,arrZ):

    df_burst = pd.DataFrame(list(zip(arrX[row, :], arrY[row, :], arrZ[row, :], GEO_LAT, GEO_LON)),
                            columns=['arrXb',
                                     'arrYb',
                                     'arrZb', 'GEO_LAT',
                                     'GEO_LON'])


    df_burst['DATE2'] = DATE2

    arraysXb.append([df_burst['arrXb'].to_numpy(),
                     OrbitNumber + '_' + DATE[0]])



    arraysYb.append([df_burst['arrYb'].to_numpy(),
                     OrbitNumber + '_' + DATE[0]])



    arraysZb.append([df_burst['arrZb'].to_numpy(),
                     OrbitNumber + '_' + DATE[0]])

    return arraysXb,arraysYb,arraysZb,df_burst

def dataframeBuild(arrX,arrY,arrZ):

    df['arrX'] = arrX.T[row, :].tolist()
    df['arrY'] = arrY.T[row, :].tolist()
    df['arrZ'] = arrZ.T[row, :].tolist()

    arraysX.append([df['arrX'].to_numpy(), OrbitNumber + '_' + DATE[0]])

    arraysY.append([df['arrY'].to_numpy(), OrbitNumber + '_' + DATE[0]])

    arraysZ.append([df['arrZ'].to_numpy(), OrbitNumber + '_' + DATE[0]])
    return arraysX,arraysY,arraysZ,df



def readFile(f):
    UTC_TIME = f["UTC_TIME"][()][:, 0]
    GEO_LAT = f["GEO_LAT"][()][:, 0]
    GEO_LON = f["GEO_LON"][()][:, 0]
    #ALT = f["ALTITUDE"][()][:, 0]
    Workmode = f["WORKMODE"][()][:, 0]
    #MAG_LAT = f["MAG_LAT"][()][:, 0]
    #MAG_LON = f["MAG_LON"][()][:, 0]
    #VERSE_TIME = f["VERSE_TIME"][()][:, 0]
    A131_W = f["A131_W"][()]
    A132_W = f["A132_W"][()]
    A133_W = f["A133_W"][()]
    A131_P = f["A131_P"][()]
    A132_P = f["A132_P"][()]
    A133_P = f["A133_P"][()]
    columns = list(f.keys())
    df = pd.DataFrame([])
    for column in columns:
        try:
```

```python
            data = np.array(f[column])
            if data.shape[1] == 1:
                df[column] = data.flatten()
            elif column.endswith('_P'):
                mat = sparse.coo_matrix(data, shape=data.shape)
                df[column] = mat.toarray().tolist()
            elif column == "A131_W":
                selected_data = np.array(data[0:len(Workmode), :])
                mat = sparse.coo_matrix(selected_data, shape=selected_data.shape)
                df[column] = mat.toarray().tolist()
            elif column == "A132_W":
                selected_data = np.array(data[0:len(Workmode), :])
                mat = sparse.coo_matrix(selected_data, shape=selected_data.shape)
                df[column] = mat.toarray().tolist()
            elif column == "A133_W":
                selected_data = np.array(data[0:len(Workmode), :])
                mat = sparse.coo_matrix(selected_data, shape=selected_data.shape)
                df[column] = mat.toarray().tolist()
            else:
                print(column + ' skipped')
        except Exception as e:
            pass
    S_burst = df[df.WORKMODE == 2]
    df['DATE_TIME'] = pd.to_datetime(df.UTC_TIME, format='%Y%m%d%H%M%S%f')
    DATE = df.DATE_TIME.map(lambda x: x.strftime('%Y-%m-%d'))
    DATE2 = df.DATE_TIME.map(lambda x: x.strftime('%Y-%m'))
    #df['DATE2'] = df.DATE_TIME.map(lambda x: x.strftime('%Y-%m'))

    return GEO_LAT,GEO_LON, A131_W, A132_W, A133_W, A131_P, A132_P, A133_P, DATE, df, S_burst,DATE2
```

In [ ]:

```python
def minmax(array):
    vals_mean = []
    arraysXdf = pd.DataFrame(array)

    for i, date in enumerate(df_complete.DATE2.unique()):

        arraysXdf_sel = arraysXdf[arraysXdf[1].str.contains(date)]

        arraysX_table = pd.DataFrame()
        for j in arraysXdf_sel.index:
            row = pd.DataFrame(arraysXdf_sel[0][j]).transpose()
            arraysX_table = pd.concat([arraysX_table, row], axis=0)

        arraysX_table.reset_index()

        vals_mean = arraysX_table.mean().transpose().to_numpy() + (i + 1) * (-50)

        max_globalX=(np.nanmax(arraysX_table.transpose())).max()
        max_global_meanX=(np.nanmax(vals_mean)).max()
        min_globalX = (np.nanmin(arraysX_table.transpose())).min()
        min_global_meanX = (np.nanmin(vals_mean)).min()
    return max_globalX,max_global_meanX,min_globalX,min_global_meanX
```

In [ ]:

```python
arraysX = []

arraysY = []

arraysZ = []

arraysXb = []

arraysYb = []

arraysZb = []



dir_name = ""
file_name = dir_name + indir_name
ext = ('.h5')

sampleFreq=51200
nRow=1024
maxFreq=sampleFreq/2
freqRow=maxFreq/nRow
freq_array=np.arange(100,2600,100)
for i in freq_array:
    for path, dirc, files in os.walk(file_name):
        for name in files:
            if name.endswith('.h5'):
                    OrbitNumber = name.split("_")[6]
                    with h5py.File(str(file_name) + str(name), "r") as f:

                        GEO_LAT, GEO_LON, A131_W, A132_W, A133_W, A131_P, A132_P, A133_P, DATE, df, S_burst, DATE2 = readFile(f)
                        f.close()

                    row=int(i/freqRow)
                    powerX = powerSpectrum(A131_P)
                    powerY = powerSpectrum(A132_P)
                    powerZ = powerSpectrum(A133_P)

                    temp_df_x=tempArray('A131_W')
                    outX_b = getData(temp_df_x)

                    temp_df_y=tempArray('A132_W')

                    outY_b = getData(temp_df_y)

                    temp_df_z=tempArray('A133_W')

                    outZ_b = getData(temp_df_z)


                    outX_b = Amplitude(outX_b)
                    outY_b = Amplitude(outY_b)
                    outZ_b = Amplitude(outZ_b)

                    arraysX,arraysY,arraysZ,df=dataframeBuild(powerX,powerY,powerZ)

                    arraysXb,arraysYb,arraysZb,df_burst=dataframeBurstBuild(outX_b,outY_b,outZ_b)
```

In [ ]:

```python
max_globalX,max_global_meanX,min_globalX,min_global_meanX=minmax(arraysX)
max_globalY,max_global_meanY,min_globalY,min_global_meanY=minmax(arraysY)
max_globalZ,max_global_meanZ,min_globalZ,min_global_meanZ=minmax(arraysZ)
max_globalXb,max_global_meanXb,min_globalXb,min_global_meanXb=minmax(arraysXb)
max_globalYb,max_global_meanYb,min_globalYb,min_global_meanYb=minmax(arraysYb)
max_globalZb,max_global_meanZb,min_globalZb,min_global_meanZb=minmax(arraysZb)
```