In [15]:

```
import h5py
import math
from math import nan
import numpy as np
import scipy
import matplotlib.pyplot as plt
from scipy import signal
import pandas as pd
from mpl_toolkits.axes_grid1.axes_divider import make_axes_locatable
import scipy.sparse as sparse
import sys
import os
```

In [14]:

```
indir_name =  "C:/CSES/file/"

outdir_name = 'C:/CSES/Statistic/'

ext = ('.h5')
```

In [ ]:

```
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:
            data = np.array(f[column])
            if data.shape[1] == 1:
                df[column] = data.flatten()
            elif column.endswith('_P'):  # Getting only A131,A132,A133 _P data
                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'))
    TIME = df.DATE_TIME.map(lambda x: x.strftime('%H-%M-%S'))
    date_burst = pd.to_datetime(S_burst.UTC_TIME, format='%Y%m%d%H%M%S%f')
    TIME_BURST = date_burst.map(lambda x: x.strftime('%H-%M-%S'))
    latb=S_burst[S_burst.GEO_LAT >= -46]

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

In [ ]:

```
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 frequency (freq):
    sampleFreq = 51200
    nRow = 1024
    maxFreq = sampleFreq / 2
    freqRow = maxFreq / nRow
    row = int(freq / freqRow)
    return row

def Amplitude2(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 LengthArray(arraysX):
    v = []
    max_len = 0
    for a in arraysX:
        if (len(a[0]) > max_len):
```

```python
            max_len = len(a[0])

    for a in arraysX:
        for _ in range(max_len - len(a[0])):
            a[0] = np.insert(a[0], 0, np.nan, axis=0)

    days = []
    for i in range(len(arraysX)):
        days.append(float(np.array(arraysX).T[1][i][-2:]))

    temp_arraysX = np.array([x for _, x in sorted(zip(days, arraysX), key=lambda pair: pair[0])])


    current_days = []
    days.sort()
    days_slot = np.arange(1, 34, 1)
    keep_day = np.zeros(33)
    if (days[0] >= 5):
        temp = days[0]
        while temp >= 5:
            temp = temp - 5
        current_days.append(0)
        days.insert(0, temp)

    for i in range(0, 7):
        if (days[0] + i * 5 in days):
            current_days.append(1)
        else:
            days.insert(i, days[0] + i * 5)
            current_days.append(0)
    for i in range(len(days_slot)):
        if (i in days):
            keep_day[i] = 1
    counter = 0

    for i in range(len(keep_day)):

        if(keep_day[i] == 1):
            if(current_days[counter]==0):
                keep_day[i] = 0
            counter += 1



    vX = np.array(temp_arraysX)
    vals_npX = []
    names_npX = []
    for i in range(len(vX)):
        vals_npX.append(vX.T[0][i])
        names_npX.append(vX.T[1][i])

    return vals_npX, names_npX, current_days, days, temp_arraysX,vX,keep_day
```

In [ ]:

```python
def ValuesPlot(path, array, ymin, ymax, xmin, xmax, tit,vX,latmin,latmax):
    plt.figure(figsize=(20, 7))
    plt.ylim(ymin, ymax)
    plt.xlim(latmin,latmax)
    plt.title(tit, fontsize=40)
    plt.xlabel('GEO_LAT')
    plt.ylabel('Amplitude [dB]')
    for el in vX:
        x_scale = (np.arange(len(el[0])) / (len(el[0]) - 1)) * (xmax - xmin) + xmin
        plt.plot(x_scale, el[0], label=el[1])
    plt.legend()
    plt.savefig(path, bbox_inches='tight')
    plt.show()


def MeanStd(vals_npX, path, tit, ymin, ymax, xmin, xmax,latmin,latmax):
    vals_mean = np.mean(vals_npX, axis=0)
    vals_std = np.std(vals_npX, axis=0)

    x_scale = (np.arange(len(vals_mean)) / (len(vals_mean) - 1)) * (xmax - xmin) + xmin

    plt.figure(figsize=(20, 7))
    plt.ylim(ymin, ymax)
    plt.xlim(latmin,latmax)
    plt.xlabel('GEO_LAT')
    plt.ylabel('Amplitude [dB]')

    plt.title(tit, fontsize=40)
    plt.plot(x_scale, vals_mean)
    plt.fill_between(x_scale, vals_mean - vals_std, vals_mean + vals_std, color='#4081EA', alpha=.2)

    plt.savefig(path, bbox_inches='tight')

    plt.show()


def MeanValues(vals_npX, tit, path, arrays, ymin, ymax, xmin, xmax,vX,latmin,latmax):
    vals_mean = np.mean(vals_npX, axis=0)
    vals_std = np.std(vals_npX, axis=0)
    plt.figure(figsize=(20, 7))
    plt.ylim(ymin, ymax)
    plt.xlim(latmin,latmax)
    plt.xlabel('GEO_LAT')
    plt.ylabel('Amplitude [dB]')

    plt.title(tit, fontsize=40)
    for el in vX:
        x_scale = (np.arange(len(el[0])) / (len(el[0]) - 1)) * (xmax - xmin) + xmin
        plt.plot(x_scale, el[0], color='#4081EA', alpha=.4)
    x_scale = (np.arange(len(vals_mean)) / (len(vals_mean) - 1)) * (xmax - xmin) + xmin
    plt.plot(x_scale, vals_mean, linewidth=3, color='#4081EA')

    plt.savefig(path, bbox_inches='tight')

    plt.show()

def Colorplot(vX, names_npX, vals_npX, tit, vmin, vmax, lat, path):
    vals_mean = np.mean(vals_npX, axis=0)
    vals_std = np.std(vals_npX, axis=0)
    x_vals = []


    plt.figure(figsize=(20, 10))

    plt.title(tit, fontsize=40)
    plt.ylabel('GEO_LAT', fontsize=20)

    vals_count = 0
    counter = 0
    for i in range(len(keep_day)):
        if (keep_day[i] == 0):

            empty_arr = np.empty(len(vals_mean))
            empty_arr.fill(np.nan)
```

```python
            plt.scatter(x=np.ones(len(vals_mean)) * i , y=np.linspace(lat.min(), lat.max(), len(vals_mean)),
                        c=empty_arr, s=10, marker='p', cmap='jet', vmin=vmin, vmax=vmax)

        else:
            plt.scatter(x=np.ones(len(vals_mean)) * i , y=np.linspace(lat.min(), lat.max(), len(vals_mean)),
                        c=vals_mean, s=10, marker='p', cmap='jet', vmin=vmin, vmax=vmax)

            vals_count += 1
        x_vals.append(i )

    plt.xticks(x_vals) #names_npX

    cbar = plt.colorbar(label="dB", orientation="vertical", pad=0.1)
    cbar.set_label("dB", size=20)
    cbar.set_ticklabels([vmin, vmax])


    plt.savefig(path, bbox_inches='tight')

    plt.show()
```

In [ ]:

```python
valsX, arraysX = [], []

valsY, arraysY = [], []

valsZ, arraysZ = [], []

valsXb, arraysXb = [], []

valsYb, arraysYb = [], []

valsZb, arraysZb = [], []

max_global_meanX = 580
min_global_meanX= 220
max_global_meanY = 580
min_global_meanY = 220
max_global_meanZ= 580
min_global_meanZ = 220
max_global_meanXb = 580
min_global_meanXb= 220
max_global_meanYb = 580
min_global_meanYb = 220
max_global_meanZb = 580
min_global_meanZb= 220


max_globalX= 460
min_globalX= 200
max_globalY= 460
min_globalY= 200
max_globalZ=  460
min_globalZ= 200
max_globalXb= 460
min_globalXb= 200
max_globalYb= 460
min_globalYb= 200
max_globalZb=460
min_globalZb= 200



dir_name = ""
file_name = dir_name + indir_name
ext = ('.h5')
df_complete = pd.DataFrame()
df_burst_complete = pd.DataFrame()

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,latb = readFile(f)

            powerX = powerSpectrum(A131_P)
            powerY = powerSpectrum(A132_P)
            powerZ = powerSpectrum(A133_P)

            temp_df_x = []
            for i in range(len(df)):
                if (df.iloc[i].WORKMODE == 2 ):
                    temp_df_x.append(df['A131_W'].iloc[i])
                else:
                    temp_df_x.append(np.empty(np.array(df['A131_W'].iloc[i]).shape))
                    temp_df_x[i][:] = np.NaN
            temp_df_x = np.array(temp_df_x)

            outX_b = getData(temp_df_x)

            temp_df_y = []
            for i in range(len(df)):
                if (df.iloc[i].WORKMODE == 2 ): #and (df.iloc[i].GEO_LAT>=-46)
                    temp_df_y.append(df['A132_W'].iloc[i])
                else:
                    temp_df_y.append(np.empty(np.array(df['A132_W'].iloc[i]).shape))
                    temp_df_y[i][:] = np.NaN
            temp_df_y = np.array(temp_df_y)

            outY_b = getData(temp_df_y)

            temp_df_z = []
            for i in range(len(df)):
                if (df.iloc[i].WORKMODE == 2 ):
                    temp_df_z.append(df['A133_W'].iloc[i])
                else:
                    temp_df_z.append(np.empty(np.array(df['A133_W'].iloc[i]).shape))
                    temp_df_z[i][:] = np.NaN
            temp_df_z = np.array(temp_df_z)

            outZ_b = getData(temp_df_z)
            freq = 100  # choose frequency 100,500,1700,2000 kHz
            row = frequency(freq)

            outX_b = Amplitude2(outX_b)
            outY_b = Amplitude2(outY_b)
            outZ_b = Amplitude2(outZ_b)

            df_burst = pd.DataFrame(list(zip(outX_b[row, :], outY_b[row, :], outZ_b[row, :], GEO_LAT, GEO_LON)),
                                    columns=[
                                        f'EFDX_Amplitude_burst zone_{freq}Hz_from waveform_[0-{len(S_burst.WORKMODE)}]',
                                        f'EFDY_Amplitude_burst zone_{freq}Hz_from waveform_[0-{len(S_burst.WORKMODE)}]',
                                        f'EFDZ_Amplitude_burst zone_{freq}Hz_from waveform_[0-{len(S_burst.WORKMODE)}]',
                                        'GEO_LAT', 'GEO_LON'])
```

```
            df[f'EFDX_Amplitude {freq}Hz_[0-{powerX.shape[0]}]_from power spectrum whole orbit'] = powerX.T[row,
                                                                                                   :].tolist()
            df[f'EFDY_Amplitude {freq}Hz_[0-{powerY.shape[0]}]_from power spectrum whole orbit'] = powerY.T[row,
                                                                                                   :].tolist()
            df[f'EFDZ_Amplitude {freq}Hz_[0-{powerZ.shape[0]}]_from power spectrum whole orbit'] = powerZ.T[row,
                                                                                                   :].tolist()

            valsX.append([df[f'EFDX_Amplitude {freq}Hz_[0-{powerX.shape[0]}]'], OrbitNumber + '_' + DATE[0]])

            arraysX.append([df[f'EFDX_Amplitude {freq}Hz_[0-{powerX.shape[0]}]'].to_numpy(), OrbitNumber + '_' + DATE[0]])

            valsY.append([df[f'EFDY_Amplitude {freq}Hz_[0-{powerY.shape[0]}]'], OrbitNumber + '_' + DATE[0]])

            arraysY.append([df[f'EFDY_Amplitude {freq}Hz_[0-{powerY.shape[0]}]'].to_numpy(), OrbitNumber + '_' + DATE[0]])

            valsZ.append([df[f'EFDZ_Amplitude {freq}Hz_[0-{powerZ.shape[0]}]'], OrbitNumber + '_' + DATE[0]])

            arraysZ.append([df[f'EFDZ_Amplitude {freq}Hz_[0-{powerZ.shape[0]}]'].to_numpy(), OrbitNumber + '_' + DATE[0]])

            valsXb.append(
                [df_burst[f'Amplitude_burst zone_{freq}Hz_EFDX_[0-{len(S_burst.WORKMODE)}]'], OrbitNumber + '_' + DATE[0]])

            arraysXb.append([df_burst[f'Amplitude_burst zone_{freq}Hz_EFDX_[0-{len(S_burst.WORKMODE)}]'].to_numpy(),
                             OrbitNumber + '_' + DATE[0]])

            valsYb.append(
                [df_burst[f'Amplitude_burst zone_{freq}Hz_EFDY_[0-{len(S_burst.WORKMODE)}]'], OrbitNumber + '_' + DATE[0]])

            arraysYb.append([df_burst[f'Amplitude_burst zone_{freq}Hz_EFDY_[0-{len(S_burst.WORKMODE)}]'].to_numpy(),
                             OrbitNumber + '_' + DATE[0]])

            valsZb.append(
                [df_burst[f'Amplitude_burst zone_{freq}Hz_EFDZ_[0-{len(S_burst.WORKMODE)}]'], OrbitNumber + '_' + DATE[0]])

            arraysZb.append([df_burst[f'Amplitude_burst zone_{freq}Hz_EFDZ_[0-{len(S_burst.WORKMODE)}]'].to_numpy(),

                          OrbitNumber + '_' + DATE[0]])


vals_npX, names_npX, current_days, days, temp_arraysX,vX,keep_day = LengthArray(arraysX)
vals_npY, names_npY, current_days, days, temp_arraysY,vY,keep_day  = LengthArray(arraysY)
vals_npZ, names_npZ, current_days, days, temp_arraysZ,vZ,keep_day  = LengthArray(arraysZ)
vals_npXb, names_npXb, current_days, days, temp_arraysXb,vXb,keep_day  = LengthArray(arraysXb)
vals_npYb, names_npYb, current_days, days, temp_arraysYb,vYb,keep_day  = LengthArray(arraysYb)
vals_npZb, names_npZb, current_days, days, temp_arraysZb,vZb,keep_day  = LengthArray(arraysZb)
```

In [ ]:

```
MeanStd(vals_npX, outdir_name + f'Amplitude_{freq}Hz_EFDX_mean and standard deviation_{DATE2[0]}.jpg',

                    f'Amplitude_{freq}Hz_EFDX_mean and standard deviation_{DATE2[0]}', min_global_meanX, max_global_meanX, GEO_LAT.min(),
                    GEO_LAT.max(),GEO_LAT.min(),GEO_LAT.max())

MeanStd(vals_npY, outdir_name + f'Amplitude_{freq}Hz_EFDY_mean and standard deviation_{DATE2[0]}.jpg',

                    f'Amplitude_{freq}Hz_EFDY_mean and standard deviation_{DATE2[0]}',  min_global_meanY, max_global_meanY, GEO_LAT.min(),
                    GEO_LAT.max(),GEO_LAT.min(),GEO_LAT.max())

MeanStd(vals_npZ, outdir_name + f'Amplitude_{freq}Hz_EFDZ_mean and standard deviation_{DATE2[0]}.jpg',

                   f'Amplitude_{freq}Hz_EFDZ_mean and standard deviation_{DATE2[0]}',  min_global_meanZ, max_global_meanZ, GEO_LAT.min(),
                   GEO_LAT.max(),GEO_LAT.min(),GEO_LAT.max())

MeanStd(vals_npXb, outdir_name + f'Amplitude_burst zone_{freq}Hz_EFDX_mean and standard deviation_{DATE2[0]}.jpg',

                    f'Amplitude_burst zone_{freq}Hz_EFDX_mean and standard deviation_{DATE2[0]}', min_global_meanXb, max_global_meanXb,
                    GEO_LAT.min(), GEO_LAT.max(),latb.GEO_LAT.min(),latb.GEO_LAT.max())
MeanStd(vals_npYb, outdir_name + f'Amplitude_burst zone_{freq}Hz_EFDY_mean and standard deviation_{DATE2[0]}.jpg',

                    f'Amplitude_burst zone_{freq}Hz_EFDY_mean and standard deviation_{DATE2[0]}',  min_global_meanYb, max_global_meanYb,
                     GEO_LAT.min(), GEO_LAT.max(),latb.GEO_LAT.min(),latb.GEO_LAT.max())

MeanStd(vals_npZb, outdir_name + f'Amplitude_burst zone_{freq}Hz_EFDZ_mean and standard deviation_{DATE2[0]}.jpg',

                    f'Amplitude_burst zone_{freq}Hz_EFDZ_mean and standard deviation_{DATE2[0]}', min_global_meanZb, max_global_meanZb,
                     GEO_LAT.min(), GEO_LAT.max(),latb.GEO_LAT.min(),latb.GEO_LAT.max())

MeanValues(vals_npX, f'Amplitude_{freq}Hz_EFDX_mean and orbits_{DATE2[0]}',

                       outdir_name + f'Amplitude_{freq}Hz_EFDX_mean and orbits_{DATE2[0]}.jpg', arraysX, min_global_meanX, max_global_meanX, GEO_LAT.min(), GEO_LAT.max(),vX,GEO_LAT.min(

MeanValues(vals_npY, f'Amplitude_{freq}Hz_EFDY_mean and orbits_{DATE2[0]}',

                        outdir_name + f'Amplitude_{freq}Hz_EFDY_mean and orbits_{DATE2[0]}.jpg', arraysY,  min_global_meanY, max_global_meanY, GEO_LAT.min(), GEO_LAT.max(),vY,GEO_LAT.mi

MeanValues(vals_npZ, f'Amplitude_{freq}Hz_EFDZ_mean and orbits_{DATE2[0]}',

                        outdir_name + f'Amplitude_{freq}Hz_EFDZ_mean and orbits_{DATE2[0]}.jpg', arraysZ,  min_global_meanZ, max_global_meanZ, GEO_LAT.min(), GEO_LAT.max(),vZ,GEO_LAT.mi

MeanValues(vals_npXb, f'Amplitude_burst zone_{freq}Hz_EFDX_mean and orbits_{DATE2[0]}',

                       outdir_name + f'Amplitude_burst zone_{freq}Hz_EFDX_mean and orbits_{DATE2[0]}.jpg', arraysXb, min_global_meanXb, max_global_meanXb, GEO_LAT.min(), GEO_LAT.max(),v

MeanValues(vals_npYb, f'Amplitude_burst zone_{freq}Hz_EFDY_mean and orbits_{DATE2[0]}',

                        outdir_name + f'Amplitude_burst zone_{freq}Hz_EFDY_mean and orbits_{DATE2[0]}.jpg', arraysYb,  min_global_meanYb, max_global_meanYb, GEO_LAT.min(), GEO_LAT.max()

MeanValues(vals_npZb, f'Amplitude_burst zone_{freq}Hz_EFDZ_mean and orbits_{DATE2[0]}',

                        outdir_name + f'Amplitude_burst zone_{freq}Hz_EFDZ_mean and orbits_{DATE2[0]}.jpg', arraysZb, min_global_meanZb, max_global_meanZb, GEO_LAT.min(), GEO_LAT.max(),

ValuesPlot(outdir_name + f'Amplitude_{freq}Hz_EFDX_{DATE2[0]}.jpg', valsX, min_globalX, max_globalX, GEO_LAT.min(),
                       GEO_LAT.max(),f'Amplitude_{freq}Hz_EFDX_{DATE2[0]}',vX,GEO_LAT.min(),GEO_LAT.max())

ValuesPlot(outdir_name + f'Amplitude_{freq}Hz_EFDY_{DATE2[0]}.jpg', valsY, min_globalY, max_globalY, GEO_LAT.min(),
                       GEO_LAT.max(),f'Amplitude_{freq}Hz_EFDY_{DATE2[0]}',vY, GEO_LAT.min(),GEO_LAT.max())

ValuesPlot(outdir_name + f'Amplitude_{freq}Hz_EFDZ_{DATE2[0]}.jpg', valsZ, min_globalZ, max_globalZ, GEO_LAT.min(),
                        GEO_LAT.max(),f'Amplitude_{freq}Hz_EFDZ_{DATE2[0]}',vZ, GEO_LAT.min(),GEO_LAT.max())

ValuesPlot(outdir_name + f'Amplitude_burst zone_{freq}Hz_EFDX_{DATE2[0]}.jpg', valsXb, min_globalXb, max_globalXb,
                       GEO_LAT.min(), GEO_LAT.max(),f'Amplitude_burst zone_{freq}0Hz_EFDX_{DATE2[0]}',vXb,latb.GEO_LAT.min(),latb.GEO_LAT.max())

ValuesPlot(outdir_name + f'Amplitude_burst zone_{freq}Hz_EFDY_{DATE2[0]}.jpg', valsYb, min_globalYb, max_globalYb,
                        GEO_LAT.min(), GEO_LAT.max(),f'Amplitude_burst zone_{freq}Hz_EFDY_{DATE2[0]}',vYb, latb.GEO_LAT.min(),latb.GEO_LAT.max())

ValuesPlot(outdir_name + f'Amplitude_burst zone_{freq}Hz_EFDZ_{DATE2[0]}.jpg', valsZb, min_globalZb, max_globalZb,  GEO_LAT.min(), GEO_LAT.max(),f'Amplitude_burst zone_2kHz_EFDZ_{DATE2[

Colorplot(vX, names_npX, vals_npX, f'Heatmap_Amplitude_{freq}Hz_EFDX_mean_{DATE2[0]}', min_global_meanX,
                    max_global_meanX, GEO_LAT,

                    outdir_name + f'Heatmap_Amplitude_{freq}Hz_EFDX_mean_{DATE2[0]}.jpg')

Colorplot(vY, names_npY, vals_npY, f'Heatmap_Amplitude_{freq}Hz_EFDY_mean_{DATE2[0]}', min_global_meanY,
                     max_global_meanY, GEO_LAT,

                     outdir_name + f'Heatmap_Amplitude_{freq}Hz_EFDY_mean_{DATE2[0]}.jpg')
```

```
Colorplot(vZ, names_npZ, vals_npZ, f'Heatmap_Amplitude_{freq}Hz_EFDZ_mean_{DATE2[0]}', min_global_meanZ,
                     max_global_meanZ, GEO_LAT,

                    outdir_name + f'Heatmap_Amplitude_{freq}Hz_EFDZ_mean_{DATE2[0]}.jpg')

Colorplot(vXb, names_npXb, vals_npXb, f'Heatmap_Amplitude_burst zone_{freq}Hz_EFDX_mean_{DATE2[0]}', min_global_meanXb, max_global_meanXb, GEO_LAT,
                    outdir_name + f'Heatmap_Amplitude_burst zone_{freq}Hz_EFDX_mean_{DATE2[0]}.jpg' )

Colorplot(vYb, names_npYb, vals_npYb, f'Heatmap_Amplitude_burst zone_{freq}Hz_EFDY_mean_{DATE2[0]}',min_global_meanYb, max_global_meanYb, GEO_LAT,
                     outdir_name + f'Heatmap_Amplitude_burst zone_{freq}Hz_EFDY_mean_{DATE2[0]}.jpg')

Colorplot(vZb, names_npZb, vals_npZb, f'Heatmap_Amplitude_burst zone_{freq}Hz_EFDZ_mean_{DATE2[0]}', min_global_meanZb, max_global_meanZb, GEO_LAT,
                   outdir_name + f'Heatmap_Amplitude_burst zone_{freq}Hz_EFDZ_mean_{DATE2[0]}.jpg')
```