In [1]:

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

In [2]:

```
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[0:1010, :])
                mat = sparse.coo_matrix(selected_data, shape=selected_data.shape)
                df[column] = mat.toarray().tolist()
            elif column == "A132_W":
                selected_data = np.array(data[0:1010, :])
                mat = sparse.coo_matrix(selected_data, shape=selected_data.shape)
                df[column] = mat.toarray().tolist()
            elif column == "A133_W":
                selected_data = np.array(data[0:1010, :])
                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'))
    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'))

    return GEO_LAT, A131_W, A132_W, A133_W, A131_P, A132_P, A133_P, DATE
```

In [3]:

```
def plot(minVal, maxVal, plotValue, vmin, vmax, title, path, labelX='LATITUDE [degree]', labelY='FREQUENCY [kHz]', barLabel='dB'):
    fig, axs = plt.subplots(sharex=True, figsize=(150, 10))
    ext = [minVal, maxVal, 0, 25.8]
    im = plt.imshow(np.fliplr(np.rot90(plotValue, 2)), interpolation='None', cmap='jet', aspect='auto', extent=ext, vmin=vmin,
                    vmax=vmax)
    axs.set_xlabel(labelX, fontsize=20)
    axs.set_ylabel(labelY, fontsize=20)
    axs.set_title(title, fontsize=40)
    plt.ylim(0, 25.8)

    divider = make_axes_locatable(axs)
    cax = divider.append_axes('right', size="1%")
    cbar = plt.colorbar(im, cax=cax, orientation='vertical')
    cbar.set_label(barLabel, size=20)
    axs.xaxis.set_ticks_position("bottom")
    axs.xaxis.set_label_position("bottom")
    axs.xaxis.grid(False)
    axs.set_xlim(minVal, maxVal)
    axs.xaxis.set_major_locator(ticker.FixedLocator(np.arange(minVal, maxVal, 5)))
    axs.tick_params(axis='x', labelsize=15)
    axs.tick_params(axis='y', labelsize=15)
    plt.savefig(path, bbox_inches='tight')
    plt.show()
```

In [4]:

```
def plot_S(minVal, maxVal, plotValue, vmin, vmax, title, path, labelX='LATITUDE [degree]', labelY='FREQUENCY [kHz]', barLabel='dB'):
    fig, axs = plt.subplots(sharex=True, figsize=(150, 10))
    ext = [minVal, maxVal, 0, 25.8]
    im = plt.imshow(np.rot90(plotValue, 2), interpolation='None', cmap='jet', aspect='auto', extent=ext, vmin=vmin,
                    vmax=vmax)
    axs.set_xlabel(labelX, fontsize=20)
    axs.set_ylabel(labelY, fontsize=20)
    axs.set_title(title, fontsize=40)
    plt.ylim(0, 25.8)

    divider = make_axes_locatable(axs)
    cax = divider.append_axes('right', size="1%")
    cbar = plt.colorbar(im, cax=cax, orientation='vertical')
    cbar.set_label(barLabel, size=20)
    axs.xaxis.set_ticks_position("bottom")
    axs.xaxis.set_label_position("bottom")
    axs.xaxis.grid(False)
    axs.set_xlim(minVal, maxVal)
    axs.xaxis.set_major_locator(ticker.FixedLocator(np.arange(minVal, maxVal, 5)))
    axs.tick_params(axis='x', labelsize=15)
    axs.tick_params(axis='y', labelsize=15)
    plt.savefig(path, bbox_inches='tight')
    plt.show()
```

In [5]:

```
def getData(data):
    minMaxBurstX_OrbitNumber = (nan, nan)

    N1 = np.full((1024, 152 * 50), np.nan)
    N2 = np.full((1024, 386 * 50), np.nan)
    M1 = np.full((1024, 152 * 25), np.nan)
```

```
    M2 = np.full((1024, 386 * 25), np.nan)
    P1 = np.full((152, 1024), np.nan)
    P2 = np.full((386, 1024), np.nan)
    Bw = 51200 * 2 / 1024
    dataX = np.empty(shape=(data.shape[0], data.shape[1]))
    for i in range(0, data.shape[0]):
        meanX_b = np.mean(data[i])
        dataX[i] = data[i] - meanX_b

    M_b = dataX.shape[1]
    hamming_b = signal.get_window("hamming", M_b)
    FFT_low = np.array([scipy.fft.fft(dataX[i] * hamming_b) for i in range(0, dataX.shape[0])])
    out = np.abs(FFT_low.T[:1024]) ** 2
    outX_b = 400 + 20 * np.log10(out / Bw)
    brx1 = np.hstack((M1, outX_b))
    brX = np.hstack((brx1, M2))

    zero = np.zeros_like(dataX)
    outX_b2 = np.array([[i, j] for i, j in zip(dataX, zero)]).reshape(2 * dataX.shape[0], dataX.shape[1])
    M_b = outX_b2.shape[1]
    hamming_b = signal.get_window("hamming", M_b)
    FFT = np.array([scipy.fft.fft(outX_b2[i] * hamming_b) for i in range(0, outX_b2.shape[0])])
    inter = np.abs(FFT.T[:1024]) ** 2
    inter2X = 400 + 20 * np.log10(inter / Bw, where=0 < inter, out=np.nan * inter)
    arx1 = np.hstack((N1, inter2X))
    arX = np.hstack((arx1, N2))

    minMaxBurstX_OrbitNumber  = (
                    ([minMaxBurstX_OrbitNumber[0], round(np.nanmin(outX_b), 2)]),
                    ([minMaxBurstX_OrbitNumber[1], round(np.nanmax(outX_b), 2)]))


    return brX, arX, minMaxBurstX_OrbitNumber, outX_b

def powerSpectrum(pow):
    minMaxPowX_OrbitNumber = (nan, nan)
    powerX=400+20*np.log10(pow)
    minMaxPowX_OrbitNumber = (
                    ([minMaxPowX_OrbitNumber[0], round(np.nanmin(powerX), 2)]),
                    ([minMaxPowX_OrbitNumber[1], round(np.nanmax(powerX), 2)]))
    return powerX, minMaxPowX_OrbitNumber
```

In [6]:

```
def runCode():



    minMaxXPower = (nan, nan)
    minMaxYPower = (nan, nan)
    minMaxZPower = (nan, nan)

    minMaxBurstX = (nan, nan)
    minMaxBurstY = (nan, nan)
    minMaxBurstZ = (nan, nan)


    dir_name = ""
    file_name = dir_name + "C:/CSES/file/"
    ext = ('.h5')
    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, A131_W, A132_W, A133_W, A131_P, A132_P, A133_P, DATE = readFile(f)







                    brX, arX, minMaxBurstX_OrbitNumber,outX_b = getData(A131_W)
                    brY, arY, minMaxBurstY_OrbitNumber,outY_b= getData(A132_W)
                    brZ, arZ, minMaxBurstZ_OrbitNumber,outZ_b= getData(A133_W)

                    powerX,minMaxPowX_OrbitNumber=powerSpectrum(A131_P)
                    powerY,minMaxPowY_OrbitNumber=powerSpectrum(A132_P)
                    powerZ,minMaxPowZ_OrbitNumber=powerSpectrum(A133_P)

                    minMaxXPower = (
                        np.nanmin([minMaxXPower[0], round(np.nanmin(powerX), 2)]),
                        np.nanmax([minMaxXPower[1], round(np.nanmax(powerX), 2)]))
                    minMaxYPower = (
                        np.nanmin([minMaxYPower[0], round(np.nanmin(powerY), 2)]),
                        np.nanmax([minMaxYPower[1], round(np.nanmax(powerY), 2)]))
                    minMaxZPower = (
                        np.nanmin([minMaxZPower[0], round(np.nanmin(powerZ), 2)]),
                        np.nanmax([minMaxZPower[1], round(np.nanmax(powerZ), 2)]))

                    minMaxBurstX = (
                       np.nanmin ([minMaxBurstX[0], round(np.nanmin(outX_b), 2)]),
                        np.nanmax([minMaxBurstX[1], round(np.nanmax(outX_b), 2)]))
                    minMaxBurstY = (
                       np.nanmin([minMaxBurstY[0], round(np.nanmin(outY_b), 2)]),
                       np.nanmax ([minMaxBurstY[1], round(np.nanmax(outY_b), 2)]))
                    minMaxBurstZ = (
                       np.nanmin ([minMaxBurstZ[0], round(np.nanmin(outZ_b), 2)]),
                        np.nanmax([minMaxBurstZ[1], round(np.nanmax(outZ_b), 2)]))

                    plot(minVal=GEO_LAT.min(),
                         maxVal=GEO_LAT.max(), plotValue=brX, vmin=minMaxBurstX[0], vmax=minMaxBurstX[1], title=OrbitNumber + "_" + DATE[
                            0] + "_EFDX_VLF_burst zone_FFT from waveform [mV/m]_ time bin 2.048_" + f"vmin={minMaxPowX_OrbitNumber[0]}" + f"_vmax={minMaxPowX_OrbitNumber[1]}" + f"_vmin
                         path="C:/CSES/PLOT_CSES/efdX_burst_orbit_" + OrbitNumber + "_" + DATE[0] + ".png")

                    plot(minVal=GEO_LAT.min(),
                         maxVal=GEO_LAT.max(), plotValue=brY, vmin=minMaxBurstY[0], vmax=minMaxBurstY[1], title=OrbitNumber + "_" + DATE[
                            0] + "_EFDY_VLF_burst zone_FFT from waveform [mV/m]_ time bin 2.048_" + f"vmin={minMaxPowY_OrbitNumber[0]}" + f"_vmax={minMaxPowY_OrbitNumber[1]}" + f"_vmin
                         path="C:/CSES/PLOT_CSES/efdY_burst_orbit_" + OrbitNumber + "_" + DATE[0] + ".png")

                    plot(minVal=GEO_LAT.min(),
                         maxVal=GEO_LAT.max(), plotValue=brZ, vmin=minMaxBurstZ[0], vmax=minMaxBurstZ[1], title=OrbitNumber + "_" + DATE[
                            0] + "_EFDZ_VLF_burst zone_FFT from waveform [mV/m]_ time bin 2.048_" + f"vmin={minMaxPowZ_OrbitNumber[0]}" + f"_vmax={minMaxPowZ_OrbitNumber[1]}" + f"_vmin
                         path="C:/CSES/PLOT_CSES/efdZ_burst_orbit_" + OrbitNumber + "_" + DATE[0] + ".png")

                    plot(minVal=GEO_LAT.min(),
                         maxVal=GEO_LAT.max(), plotValue=arX, vmin=minMaxBurstX[0], vmax=minMaxBurstX[1], title=OrbitNumber + "_" + DATE[
                            0] + "_EFDX_VLF_burst zone_FFT from waveform [mV/m]_ time bin 0.04096_" + f"vmin={minMaxPowX_OrbitNumber[0]}" + f"_vmax={minMaxPowX_OrbitNumber[1]}" + f"_vmi
                         path="C:/CSES/PLOT_CSES/efdX_inter_orbit_" + OrbitNumber + "_" + DATE[0] + ".png")

                    plot(minVal=GEO_LAT.min(),
                         maxVal=GEO_LAT.max(), plotValue=arY, vmin=minMaxBurstY[0], vmax=minMaxBurstY[1], title=OrbitNumber + "_" + DATE[
                            0] + "_EFDY_VLF_burst zone_FFT from waveform [mV/m]_ time bin 0.04096_" + f"vmin={minMaxPowY_OrbitNumber[0]}" + f"_vmax={minMaxPowY_OrbitNumber[1]}" + f"_vmi
                         path="C:/CSES/PLOT_CSES/efdY_inter_orbit_" + OrbitNumber + "_" + DATE[0] + ".png")

                    plot(minVal=GEO_LAT.min(),
                         maxVal=GEO_LAT.max(), plotValue=arZ, vmin=minMaxBurstZ[0], vmax=minMaxBurstZ[1], title=OrbitNumber + "_" + DATE[
                            0] + "_EFDZ_VLF_burst zone_FFT from waveform [mV/m]_ time bin 0.04096_" + f"vmin={minMaxPowZ_OrbitNumber[0]}" + f"_vmax={minMaxPowZ_OrbitNumber[1]}" + f"_vmi
```

```
        path="C:/CSES/PLOT_CSES/efdZ_inter_orbit_" + OrbitNumber + "_" + DATE[0] + ".png")

    plotS(minVal=GEO_LAT.min(),
          maxVal=GEO_LAT.max(), plotValue=powerX, vmin=minMaxXPower[0], vmax=minMaxXPower[1], title=OrbitNumber + "_" + DATE[
             0] + "_EFDX_VLF_whole orbit survey mode_ power spectrum[mV/Hz^0.5]_ time bin 2.048_" + f"vmin={minMaxPowX_OrbitNumber[0]}" + f"_vmax={minMaxPowX_OrbitNumber[
          path="C:/CSES/PLOT_CSES/efdX_survey_orbit_" + OrbitNumber + "_" + DATE[0] + ".png")

    plotS(minVal=GEO_LAT.min(),
          maxVal=GEO_LAT.max(), plotValue=powerY, vmin=minMaxYPower[0], vmax=minMaxYPower[1], title=OrbitNumber + "_" + DATE[
             0] + "_EFDY_VLF_whole orbit survey mode_ power spectrum[mV/Hz^0.5]_ time bin 2.048_" + f"vmin={minMaxPowY_OrbitNumber[0]}" + f"_vmax={minMaxPowY_OrbitNumber[
          path="C:/CSES/PLOT_CSES/efdY_survey_orbit_" + OrbitNumber + "_" + DATE[0] + ".png")

    plotS(minVal=GEO_LAT.min(),
          maxVal=GEO_LAT.max(), plotValue=powerZ, vmin=minMaxZPower[0], vmax=minMaxZPower[1], title=OrbitNumber + "_" + DATE[
             0] + "_EFDZ_VLF_whole orbit survey mode_ power spectrum[mV/Hz^0.5]_ time bin 2.048_" + f"vmin={minMaxPowZ_OrbitNumber[0]}" + f"_vmax={minMaxPowZ_OrbitNumber[
          path="C:/CSES/PLOT_CSES/efdZ_survey_orbit_" + OrbitNumber + "_" + DATE[0] + ".png")
```

In [ ]:

```
runCode()
```