研究室インターン（小嶋研）

宇宙空間で観測されるプラズマ波動の周波数スペクトル構造の抽出

レポート

工学研究科　電波工学専攻　大村研究室

1030-32-5617　LIU YIN

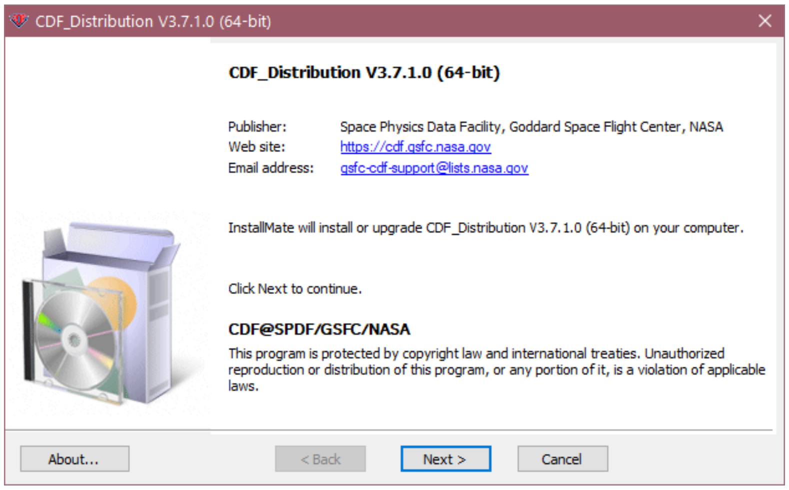
1. Data Preparation

1.1 CDF Software

Data from the ERG satellite is all in a common data format (CDF), in order to open them, a CDF software should be installed for the first step. Here, a newest version of CDF software is installed from the website below.

https://spdf.gsfc.nasa.gov/pub/software/cdf/dist/cdf37_1/

A screenshot from installation panel is shown below.



CDF_Distribution V3.7.1.0 (64-bit)

Publisher:	Space Physics Data Facility, Goddard Space Flight Center, NASA
Web site:	https://cdf.gsfc.nasa.gov
Email address:	gsfc-cdf-support@lists.nasa.gov

InstallMate will install or upgrade CDF_Distribution V3.7.1.0 (64-bit) on your computer.

Click Next to continue.

CDF@SPDF/GSFC/NASA

This program is protected by copyright law and international treaties. Unauthorized reproduction or distribution of this program, or any portion of it, is a violation of applicable laws.

Figure 1. CDF software installation

1.2 SpacePy Package

In order to manage the CDF data in Python environment, thanks to *S. Morley and D. Welling, etc.*, the Python-based library of tools for the space sciences, SpacePy, is applied here. The homepage with open resource is

https://github.com/spacepy/spacepy

To install the SpacePy package, simply using "pip install –upgrade spacepy" in command line will work.

SpacePy Library has very comprehensive functions in analyzing data from space science, which are far powerful than just drawing energy spectrums of wave fields. However, the library still has some shortcomings, like we are not able to implement a FFT on the original high-resolution waveform data.

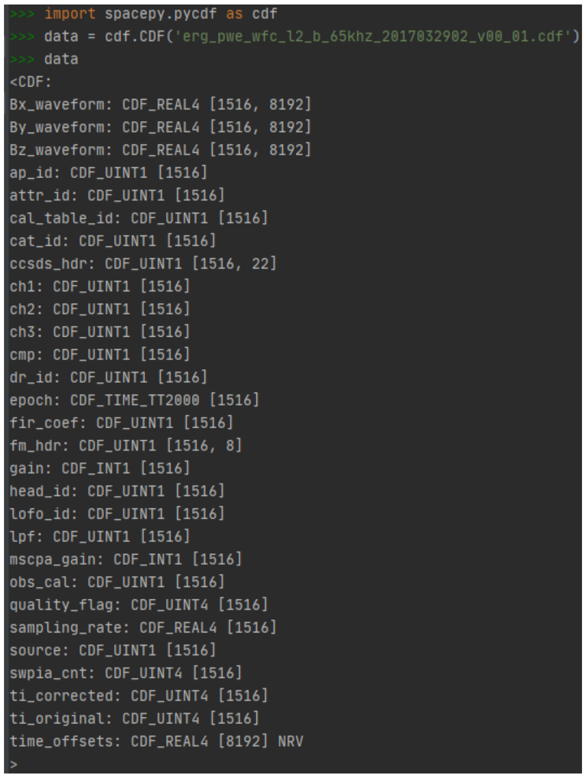
Therefore, only the *pycdf* submodule of SpacePy package is employed to read CDF data to a dictionary type of Python here. (A preinstallation of CDF software is necessary in order to use pycdf module smoothly.)

2. Main Program

2.1 Data Segmentation

In this program, only data form Wave Form Capture (WFC) and Magnetic field experiment (MGF) are used, the former is the main role and discussed hereafter.

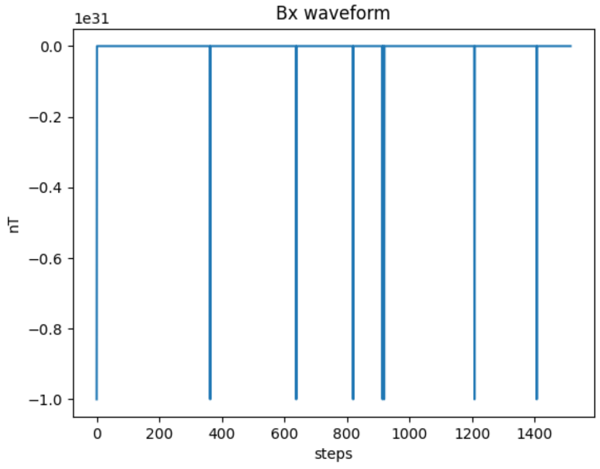
A view of the contents is shown below.



```
>>> import spacepy.pycdf as cdf
>>> data = cdf.CDF('erg_pwe_wfc_l2_b_65khz_2017032902_v00_01.cdf')
>>> data
<CDF:
Bx_waveform: CDF_REAL4 [1516, 8192]
By_waveform: CDF_REAL4 [1516, 8192]
Bz_waveform: CDF_REAL4 [1516, 8192]
ap_id: CDF_UINT1 [1516]
attr_id: CDF_UINT1 [1516]
cal_table_id: CDF_UINT1 [1516]
cat_id: CDF_UINT1 [1516]
ccsds_hdr: CDF_UINT1 [1516, 22]
ch1: CDF_UINT1 [1516]
ch2: CDF_UINT1 [1516]
ch3: CDF_UINT1 [1516]
cmp: CDF_UINT1 [1516]
dr_id: CDF_UINT1 [1516]
epoch: CDF_TIME_TT2000 [1516]
fir_coef: CDF_UINT1 [1516]
fm_hdr: CDF_UINT1 [1516, 8]
gain: CDF_INT1 [1516]
head_id: CDF_UINT1 [1516]
lofo_id: CDF_UINT1 [1516]
lpf: CDF_UINT1 [1516]
mscpa_gain: CDF_INT1 [1516]
obs_cal: CDF_UINT1 [1516]
quality_flag: CDF_UINT4 [1516]
sampling_rate: CDF_REAL4 [1516]
source: CDF_UINT1 [1516]
swpia_cnt: CDF_UINT4 [1516]
ti_corrected: CDF_UINT4 [1516]
ti_original: CDF_UINT4 [1516]
time_offsets: CDF_REAL4 [8192] NRV
>
```

Figure 2. Contents of WFC data

WFC Data from the Waveform Capture of ERG satellite is accompanied with large value pulses which are corresponding to the different capturing time, as shown below.



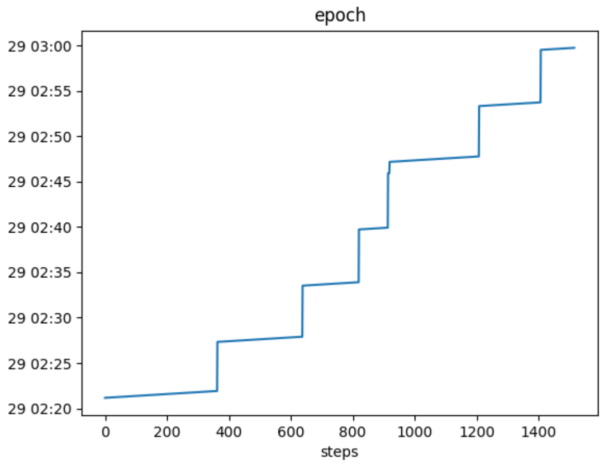


Figure 3. Original waveform data

Figure 4. Epoch of waveform data

From the relation between waveform data and epoch data, I split them to different segments which have continuous waveform and time. The splitting method is based on the large time interval as shown in Figure 4.

The epoch data is stored as an NumPy array of *datetime.datetime* type. There are several ways to extract the gap positions, like manipulation on the original type or convert them to float number. Here I chose the latter by using a built-in "*timestamp*" method. Figure 5 shows how large of each epoch interval by taking a differentiation, and positions with value larger than 10 are considered as a time interval simply.

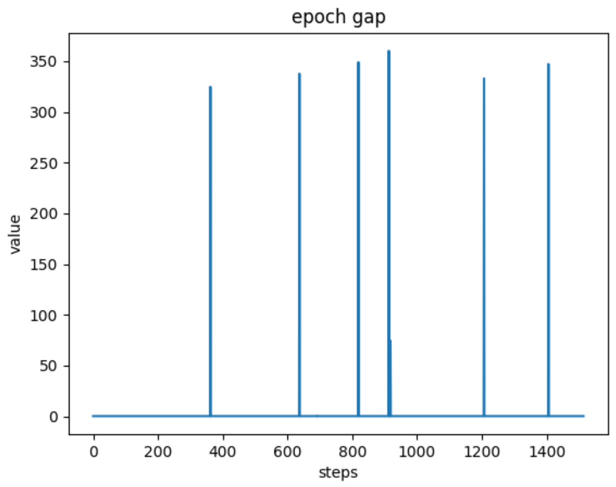


Figure 5. Epoch gap value

Although we can see the obvious gaps from waveform data as well, it is not appropriate to find the splitting positions from them, since the waveform interval will last for many steps, while the time interval will only occupy one single step. Here, an important parameter of value 1021 is observed, which is the total lasting steps for a waveform gap. A diagram about this is shown in Figure 5.

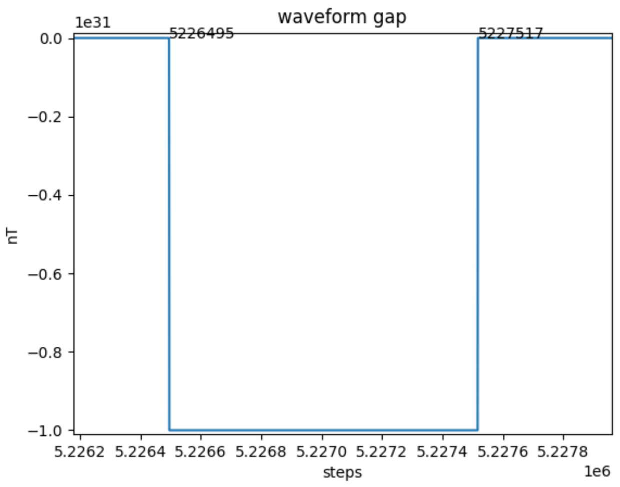


Figure 6. Waveform gap

The original waveform data is in a matrix of steps x 8192, in which 8192 is the length of time_offsets data, meaning 8192 data is captured in each step. Figure 7 shows a total waveform data after segmentation and the split positions.

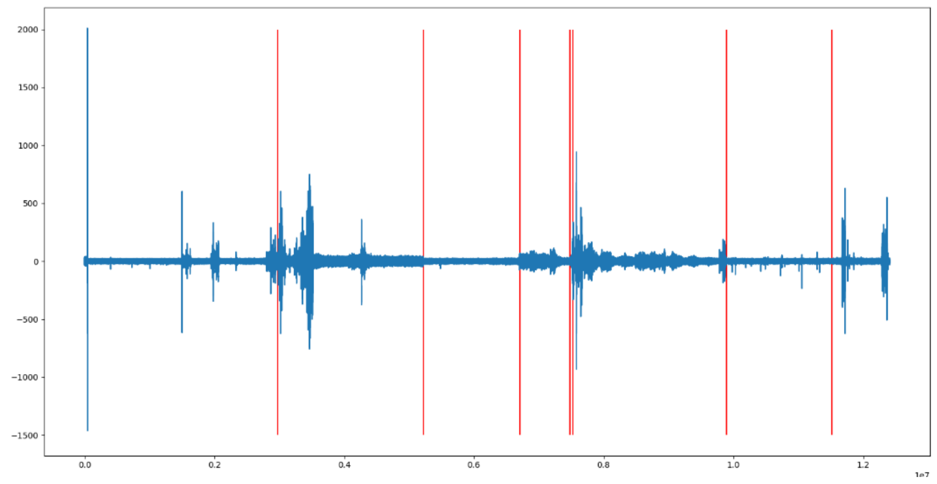


Figure 7. waveform data after segmentation

Base on the split waveform and epoch data, by implementing FFT, we are able to get their spectrums.

2.2 FFT setting

Since the nominal time resolution is 1/65536 s in the WFC waveform data, the sampling frequency is set to 65536 Hz. In addition, a Hanning window function with length NFFT=8192 is implemented. The number of points of overlap between blocks is set to 4096.

2.3 Specgram Method

The first method chose here is the *specgram* of Matplotlib package. For a full description, please refer the website below.

https://matplotlib.org/api/_as_gen/matplotlib.pyplot.specgram.html?highlight=specgram#matplotlib.pyplot.specgram

By implementing the specgram method on the split waveforms, we can easily obtain their energy spectrums, while the problem is the correspondence between spectrum and epoch data is missing.

Here, from the theory of FFT, we have a relation of

$$t_{spectrum} = (i+1) \times (NFFT - overlap) \times \frac{1}{Fs},$$

in which i is the index of time in spectrum $t_{spectrum}$.

Thus, the time ticks are calculated by the equation above and added to the spectrum manually. A final result is shown below.

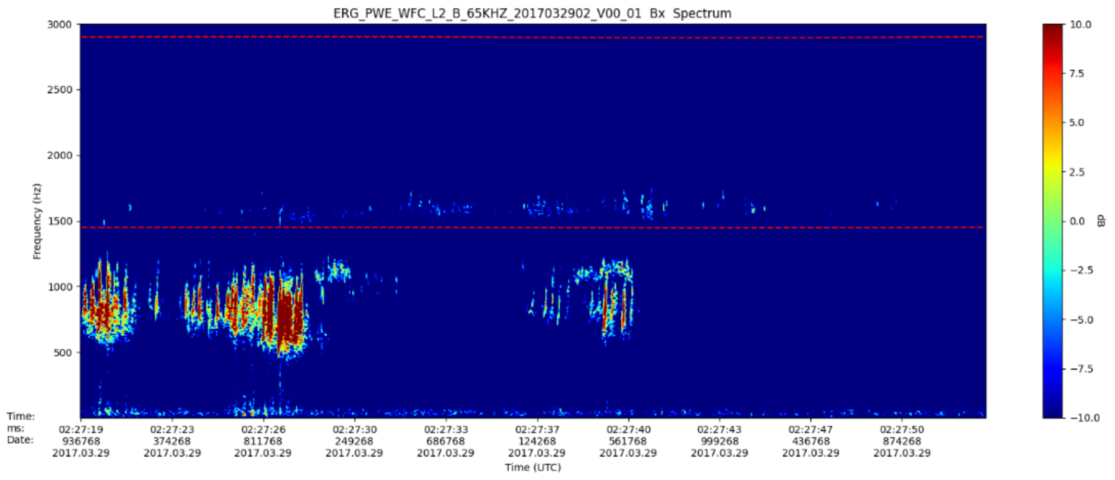


Figure 8. Wave spectrum by specgram method

2.4 Pcolormesh Method

Another method implemented is the *pcolormesh* from Matplotlib package, referring the website below,

https://matplotlib.org/api/_as_gen/matplotlib.axes.Axes.pcolormesh.html#matplotlib.axes.Axes.pcolormesh

The pcolormesh method offers a method to connect epoch and spectrum, hence no need to add time ticks manually. In the method, the spectrum data from the specgram method is used. Thus, the specgram method is necessary in present program.

However, one point should care is the time ticks calculated above need to be transferred to the *datetime64[ns]* type, and an error between UTC and local time should be taken into account.

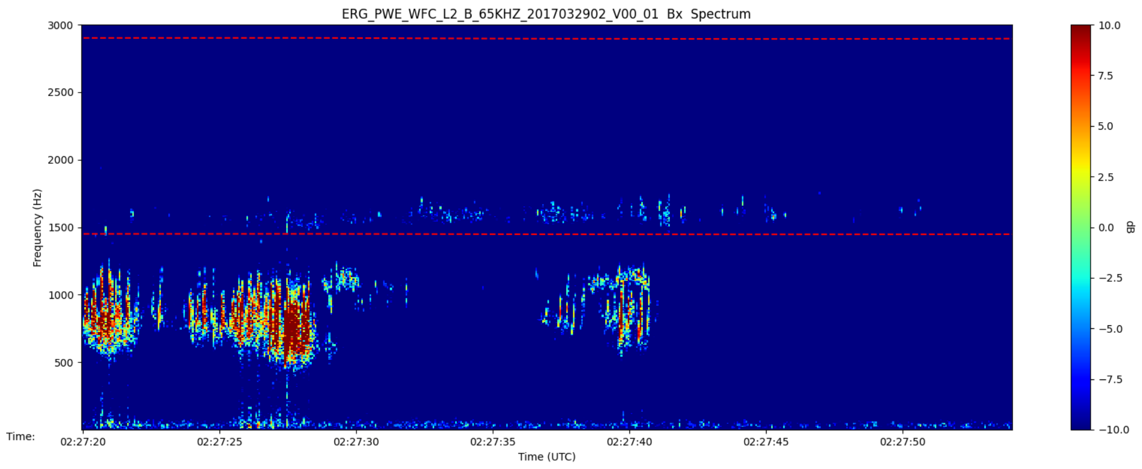


Figure 9. Wave spectrum by pcolormesh method

Comparing the spectrums in Figure 8 and 9, we find the specgram method provides a clearer result, yet the time ticks of it are added manually, meaning when we enlarge the figure, there may not exit time tick, while for the pcolormesh method, we can find the information for time at each point.

2.5 f_{ce} Line

A standard spectrum of plasma wave needs the electron cyclotron frequency line f_{ce} and its half $0.5f_{ce}$ line. In this step, we have to use the magt_8sec and epoch_8sec data from MGF file, representing the total magnetic strength and the total time in a single day with a normal time resolution of 8s.

The calculation equation from magt_8sec to f_{ce} is given by

$$f_{ce} = Q/(M * 2\pi) * magt\_8sec * 10^{-9},$$

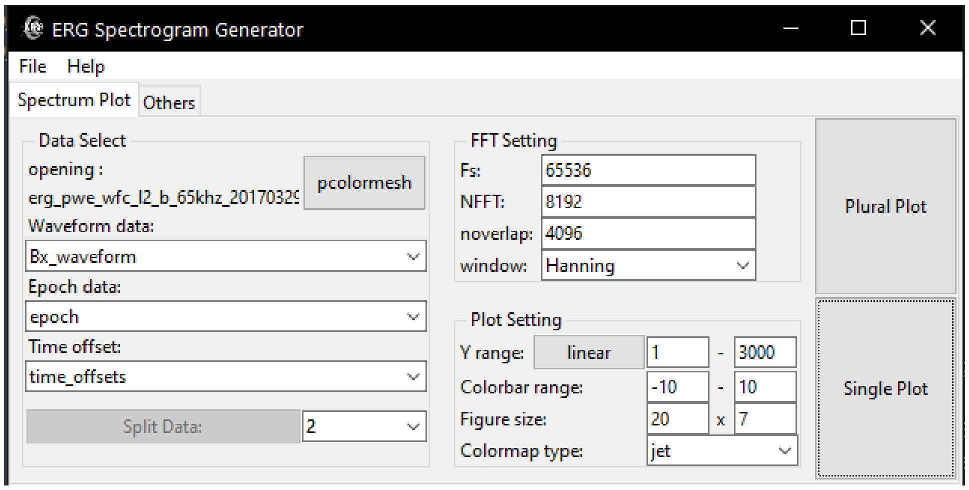
where Q and M are the charge and mass of one electron respectively.

Noticing that the difference of time resolution between WFC and MGF data is huge, we have to extract the very tiny time segment corresponding to the epoch in WFC data from the epoch_8sec in MGF data. Besides, for the time axis of specgram method has to be added manually, we need calculate the time ticks of f_{ce} line in order to plot it. In the pcolormesh method, a simply operation of plotting epoch_8sec versus magt_8sec will be enough.

The f_{ce} and $0.5f_{ce}$ line are also presented in Figure 8 and 9 with red dotted format.

3. GUI Introduction

A simple python GUI based on Tkinter package is created for a user-friendly purpose, as shown below.



opening : erg_pwe_wfc_l2_b_65khz_2017032
Waveform data: Bx_waveform
Epoch data: epoch
Time offset: time_offsets
Split Data: 2
Fs: 65536
NFFT: 8192
noverlap: 4096
window: Hanning
Y range: linear 1 - 3000
Colorbar range: -10 - 10
Figure size: 20 x 7
Colormap type: jet

Figure 10. GUI for spectrum generation

Main designing ideas in the GUI is referred to <Python GUI Programming Cookbook> by Burkhard A. Meier. The resource code of the 3rd edition is published in the GitHub,

https://github.com/PacktPublishing/Python-GUI-Programming-Cookbook-Third-Edition

In this report, trivial parts of setting each widget are omitted, and the operating procedure is mainly focused, showing below.

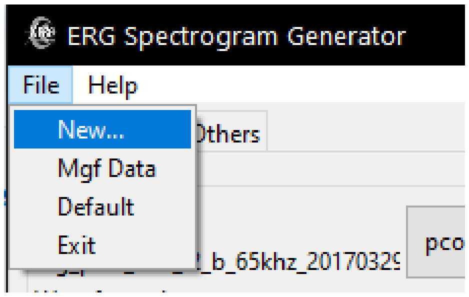
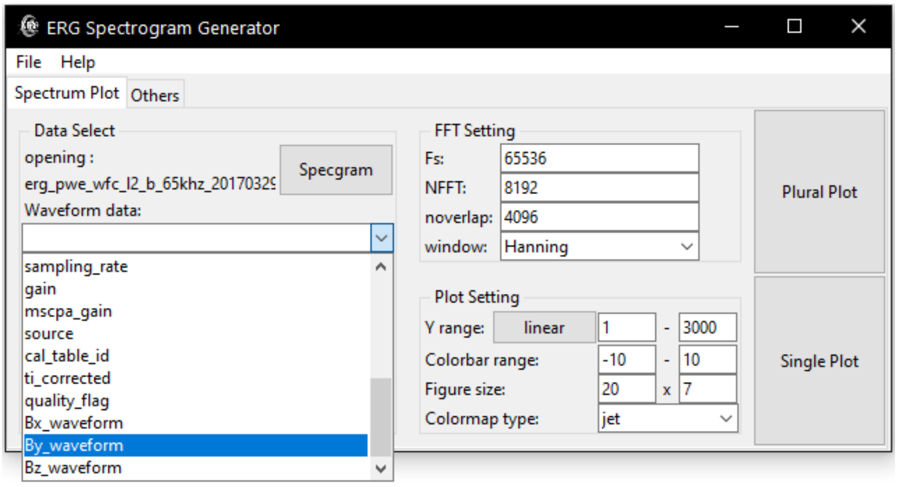


Figure 11. The file menu

a. By clicking "New…" button, we can choose a CDF file of WFC data;
b. In order to plot f_{ce} line, we need click "Mgf Data" button to choose a CDF file of MGF;
c. By clicking the "Default", the main parameters will be filled;
d. In the main panel shown in Figure 12, we need choose a waveform data to generate its spectrum;
e. Click the "Split Data" button to finish the data segmentation and choose a segment in the box on the right;
f. Click the "Specgram" button to switch plotting method between specgram and pcolormesh;
g. Click the "linear" button to switch the y scale between linear and log;
h. Clicking the "Single Plot" button will generate a single spectrum of current waveform;
i. Clicking the "Plural Plot" button will generate the spectrums of all waveforms in the opening data;
j. The "Colormap type" box provides several types of colormap in order to present the details in the spectrum;
k. The "Others" tab is left for the additional functions in the future.



opening : erg_pwe_wfc_l2_b_65khz_2017032

Waveform data:

sampling_rate
gain
mscpa_gain
source
cal_table_id
ti_corrected
quality_flag
Bx_waveform
By_waveform
Bz_waveform

Fs: 65536
NFFT: 8192
noverlap: 4096
window: Hanning

Y range: linear 1 - 3000
Colorbar range: -10 - 10
Figure size: 20 x 7
Colormap type: jet

Figure 12. Main panel in GUI

Several results are shown below:

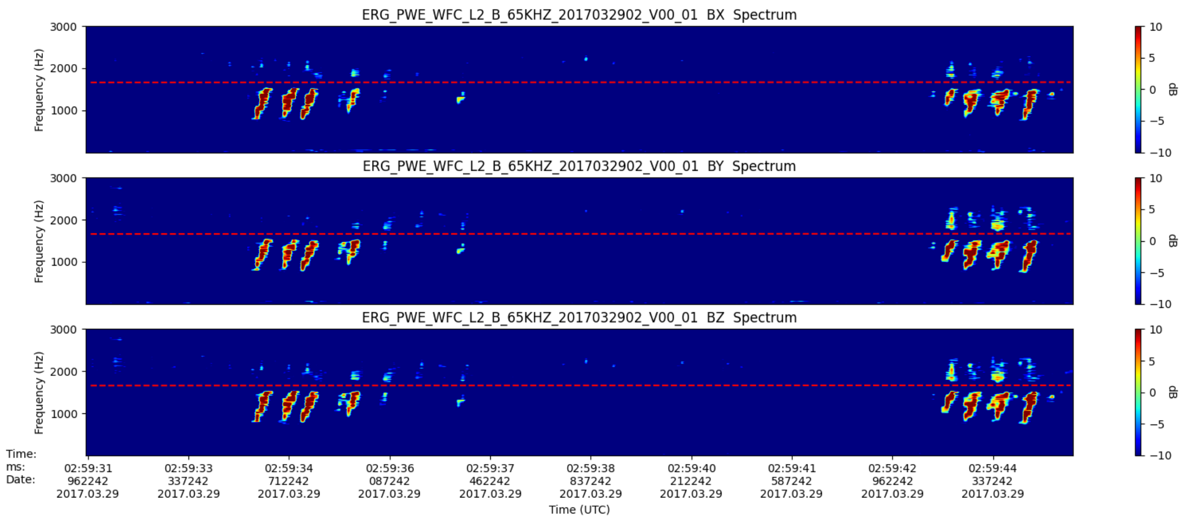


Figure 13. Spectrums of magnetic waveform by specgram method

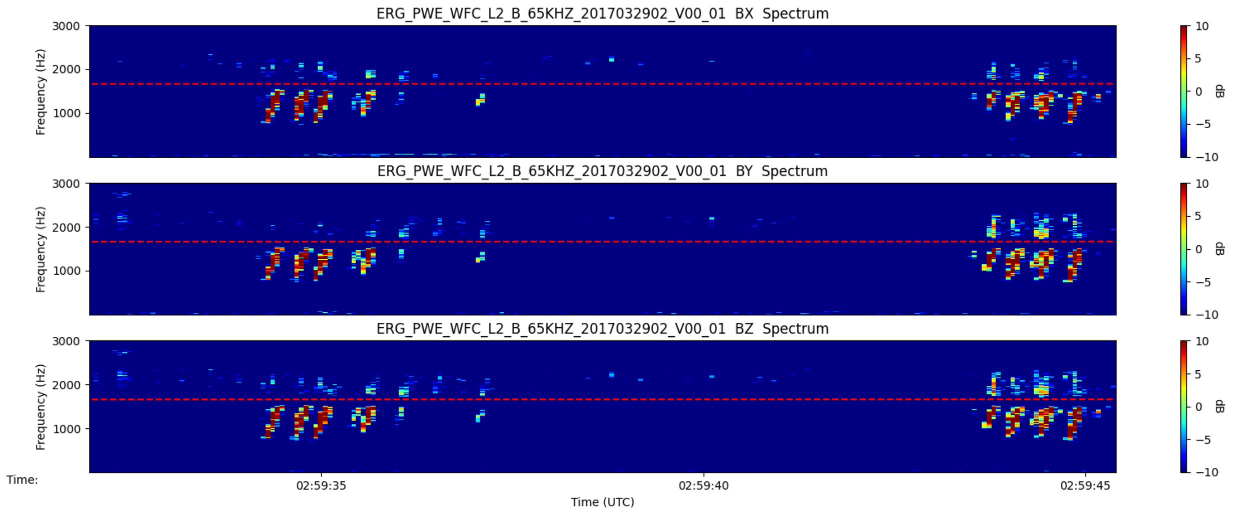


Figure 14. Spectrums of magnetic waveform by pcolormesh method

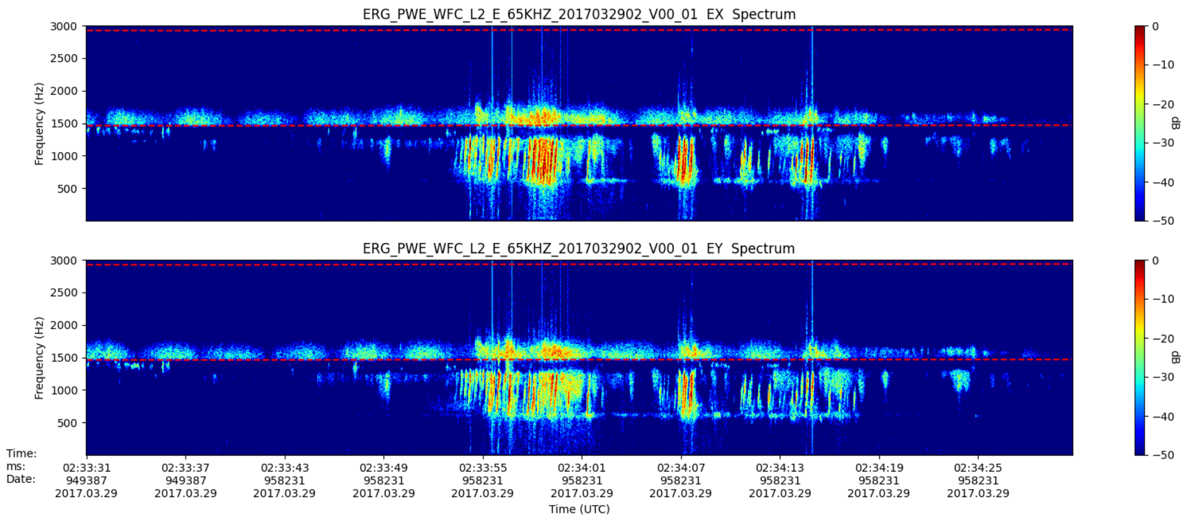


Figure 15. Spectrums of electric waveform by specgram method with linear y scale

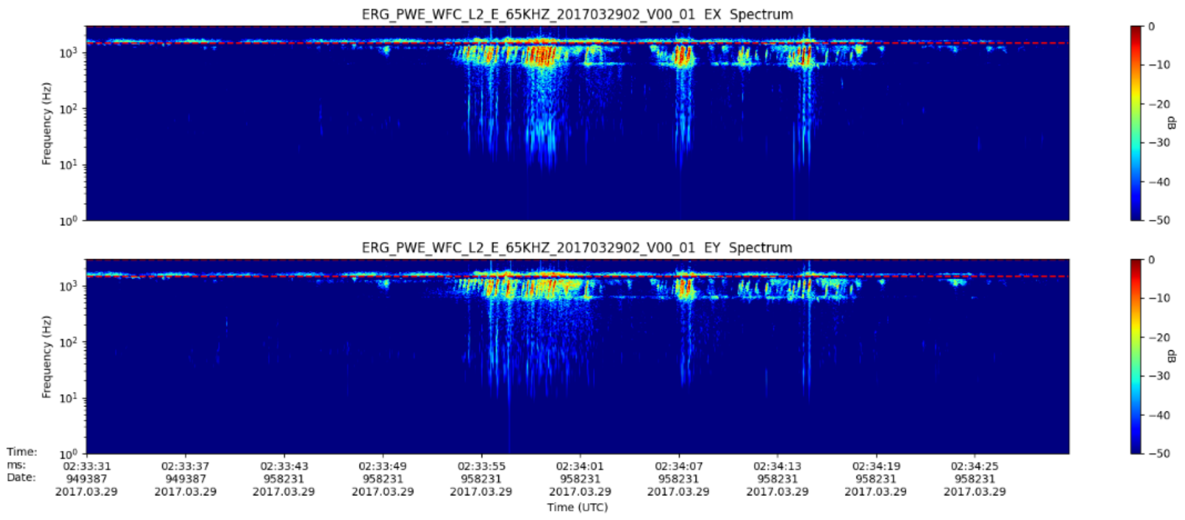


Figure 15. Spectrums of electric waveform by specgram method with log y scale