SPEDAS training session (28 Mar. 2018)

Advanced course [1 of 2]

Arase/PWE OFA-MATRIX analysis

OFA-SPEC (every 1 s in the nominal mode)

OFA-SPEC is the auto-spectra of a selected channel for electric field (E_U or E_V or |E|) and magnetic field (B_α or B_β or B_γ or |B|) from a few Hz to 20 kHz.

OFA-MATRIX (every 8 s in the nominal mode)

OFA-MATRIX is the variance-covariance matrix for the propagation direction finding and polarization analyses. Since the two CPUs individually produce electric and magnetic field spectral matrices (S_E and S_B), OFA-MATRIX does not measure any cross-spectra of an electric and magnetic field signal.

$$S_E(\omega) = \frac{1}{4}\sum_{n=0}^{4}\begin{pmatrix} |E_u^{(n)}|^2 & E_u^{(n)}E_v^{(n)*} \\ E_v^{(n)}E_u^{(n)*} & |E_v^{(n)}|^2 \end{pmatrix} \quad S_B(\omega) = \frac{1}{4}\sum_{n=0}^{4}\begin{pmatrix} |B_\alpha^{(n)}|^2 & B_\alpha^{(n)}B_\beta^{(n)*} & B_\alpha^{(n)}B_\gamma^{(n)*} \\ B_\beta^{(n)}B_\alpha^{(n)*} & |B_\beta^{(n)}|^2 & B_\beta^{(n)}B_\gamma^{(n)*} \\ B_\gamma^{(n)}B_\alpha^{(n)*} & B_\gamma^{(n)}B_\beta^{(n)*} & |B_\gamma^{(n)}|^2 \end{pmatrix}$$

OFA-COMPLEX (every 8 s in the nominal mode)

OFA-COMPLEX is the complex spectra calculated using the FFT of the observed waveforms. The complex spectra for all five components are always generated simultaneously. We can calculate a Poynting flux, and determine an absolute direction of a plasma waves by using the OFA-MATRIX data and the OFA-COMPLEX data.

1. CDF読み込み

- PWE/OFA-MATRIX L2(pre)
- MGF 64Hz L2(pre)

2. 背景磁場座標系への回転行列を計算

- MGF 64HzデータをOFA-MATRIXデータの時刻に合わせて内挿

3. スペクトルマトリクス（SGI座標系）を背景磁場座標系に変換

4. スペクトルマトリクスの対角成分（パワースペクトル）をプロット

5. 伝搬ベクトルの向きを推定

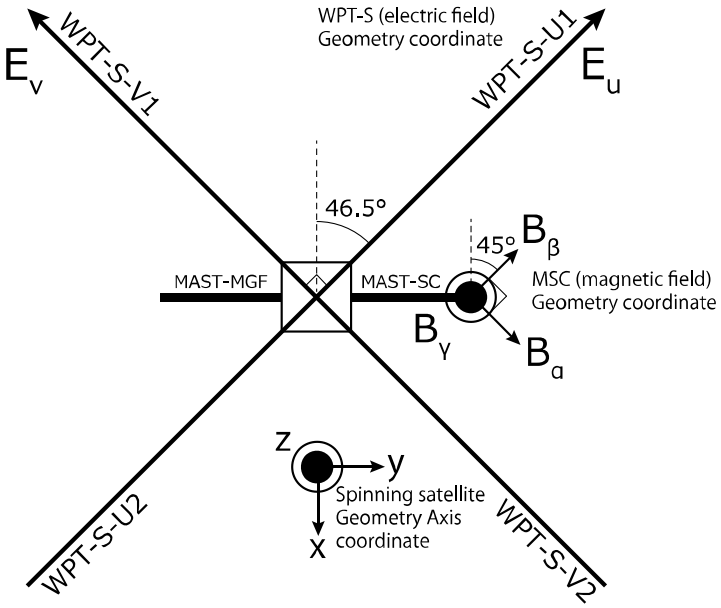
- Means' method

6. 偏波を計算

7. 解析結果をパワーでマスク

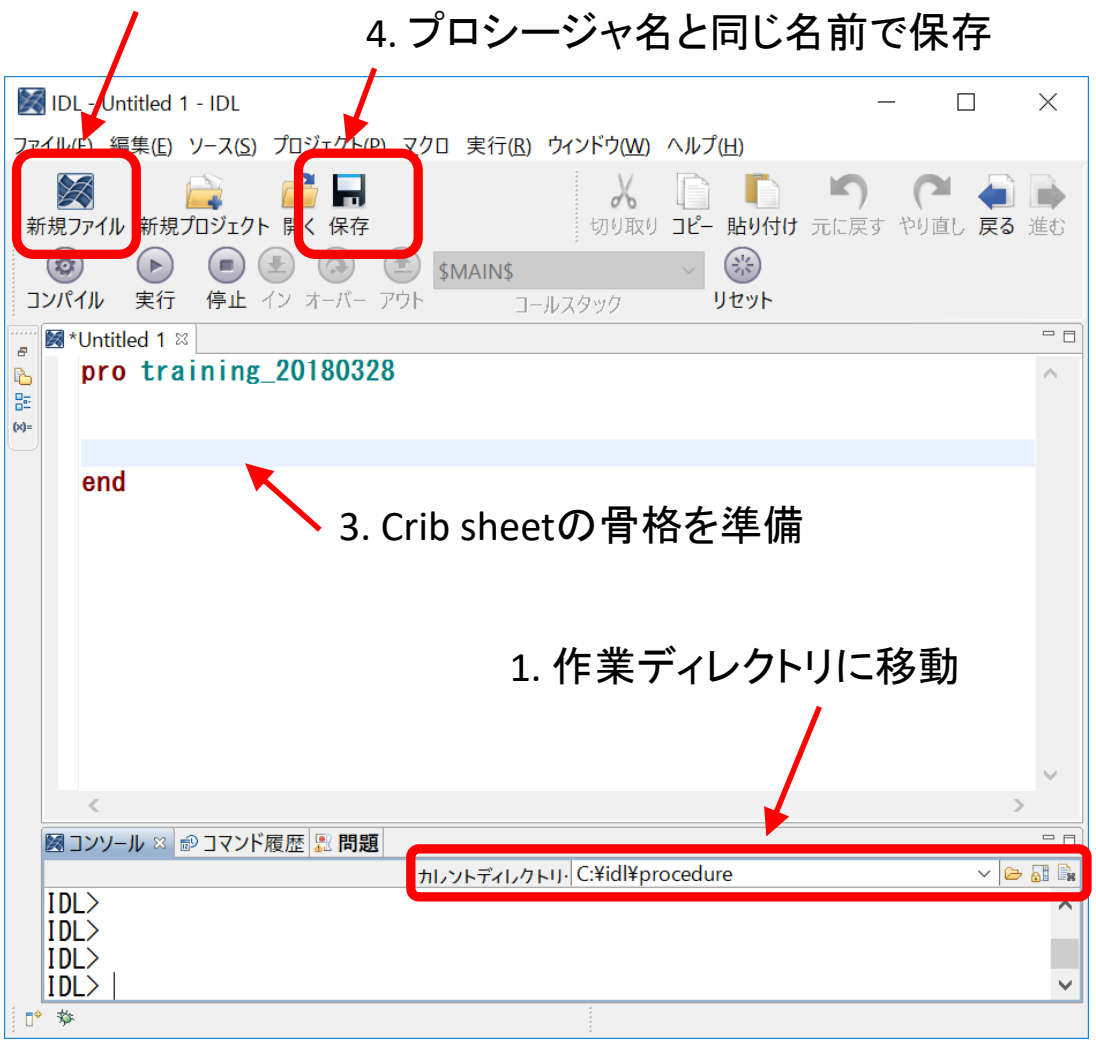
8. サイクロトロン周波数をオーバープロット

- MGF 8sec L2 CDFを読み込み



0. crib sheetを作成する準備

0.1 コンソールからの入力では，複雑な手順を記述することが困難です．
Crib sheet(手順書・スクリプト)を作成しましょう．



0.2
以下のコマンドで，crib sheetの途中で「一時停止」「再開」ができます

(Crib sheet)

```
; ****************
; test
; ****************
print, 'test1'

;stopコマンドで一時停止
stop

print, 'test2'
```

(Console)

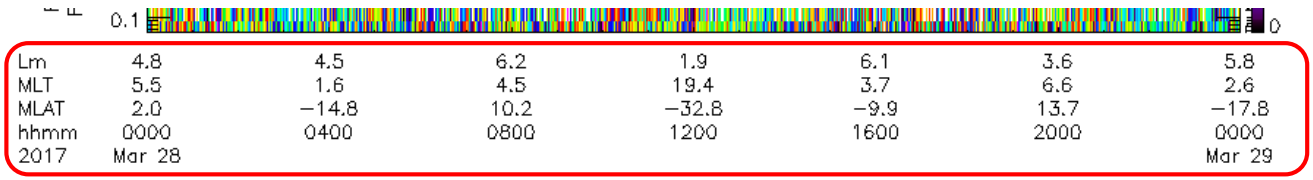
```
test1
% Stop encountered: *****
> .cont
test2
```

1. 必要なファイルの読み込み

1.1 ファイルのダウンロード設定を，講習会用に変更します．

```
; ******************
; settings for the training
; ******************
erg_init, remote_data_dir= $
        'https://ergsc.isee.nagoya-u.ac.jp/data/ergsc_training/nagoya_201803/'

uname = '***************'
pass =  '***************'

; ******************
; set time span
; ******************
timespan, '2017-03-28', 1, /day
```

1.2 軌道L2 CDFファイルを取得して，プロット下部に表示するように設定します




```
; ******************
; load orbit CDF & set var label
; ******************
set_erg_var_label
```

1. 必要なファイルの読み込み

1.3 PWE OFA-MATRIX(スペクトルマトリクス) CDFを取得します
また，磁場座標系への座標変換のためにMGF 64Hz L2 CDFを取得します

```
; *****************
; load OFA-MATRIX
; *****************
erg_load_pwe_ofa, datatype='matrix', uname=uname, pass=pass
pr_matrix = 'erg_pwe_ofa_matrix_l2_'


; *****************
; load MGF L2 CDF
; *****************
erg_load_mgf, datatype='64hz', coord='sgi', uname=uname, pass=pass
```

'spec' : パワースペクトル(OFA-SPEC)
'matrix' : スペクトルマトリクス(OFA-MATRIX)
'complex' : 複素スペクトル(OFA-COMPLEX)

'8sec' : 8秒平均データ
'64hz' : 64Hzデータ
'256hz' : 256Hzデータ

'sgi': Spinning satellite Geometry Inertia coordinate
'dsi': Despun Sun sector Inertia coordinate

> tplot_names

...
26 erg_pwe_ofa_matrix_l2_Ex_Ex_132
27 erg_pwe_ofa_matrix_l2_Ey_Ey_132
28 erg_pwe_ofa_matrix_l2_Ex_Ey_132
29 erg_pwe_ofa_matrix_l2_Ey_Ex_132
30 erg_pwe_ofa_matrix_l2_Etotal_132
31 erg_pwe_ofa_matrix_l2_quality_flag_e132
32 erg_pwe_ofa_matrix_l2_Bx_Bx_132
33 erg_pwe_ofa_matrix_l2_By_By_132
34 erg_pwe_ofa_matrix_l2_Bz_Bz_132
35 erg_pwe_ofa_matrix_l2_Bx_By_132
36 erg_pwe_ofa_matrix_l2_Bx_Bz_132
37 erg_pwe_ofa_matrix_l2_By_Bx_132
38 erg_pwe_ofa_matrix_l2_By_Bz_132
39 erg_pwe_ofa_matrix_l2_Bz_Bx_132
40 erg_pwe_ofa_matrix_l2_Bz_By_132
41 erg_pwe_ofa_matrix_l2_Btotal_132
42 erg_pwe_ofa_matrix_l2_quality_flag_b132

> tplot_names

...
43 erg_mgf_l2_date_time_64hz
44 erg_mgf_l2_mag_64hz_sgi
45 erg_mgf_l2_dyn_rng_64hz
46 erg_mgf_l2_quality_64hz
47 erg_mgf_l2_quality_64hz_gc
48 erg_mgf_l2_spin_phase_64hz
49 erg_mgf_l2_ti_64hz

1. 必要なファイルの読み込み

1.4 get_dataコマンドを用いて，スペクトルマトリクスの各成分をIDL変数に格納します

```
; *****************
; analyze OFA-MATRIX
; *****************
get_data, pr_matrix + 'Bx_Bx_132', data=s00
get_data, pr_matrix + 'Bx_By_132', data=s01
get_data, pr_matrix + 'Bx_Bz_132', data=s02

get_data, pr_matrix + 'By_Bx_132', data=s10
get_data, pr_matrix + 'By_By_132', data=s11
get_data, pr_matrix + 'By_Bz_132', data=s12

get_data, pr_matrix + 'Bz_Bx_132', data=s20
get_data, pr_matrix + 'Bz_By_132', data=s21
get_data, pr_matrix + 'Bz_Bz_132', data=s22
```

> help, s10
** Structure <155fc930>, 2 tags,
length=11489600, data length=11489600, refs=1:

X	DOUBLE	Array[10806]	*(時刻情報)*
Y	FLOAT	Array[10806, 132, 2]	*(クロススペクトル)*
V	FLOAT	Array[132]	*(第二軸のラベル)*

スペクトルマトリクスの各要素は
x(時刻) * 132(周波数分解能) * 2(実部,虚部)
の大きさの配列に格納されています

1.5 スペクトルマトリクスの各成分を用いて，3*3行列を作成します

```
rr=dblarr(3,3,n_elements(s00.x),n_elements(s00.v2),2)

rr[0,0,*,*,*]=s00.y & rr[1,0,*,*,*]=s01.y & rr[2,0,*,*,*]=s02.y
rr[0,1,*,*,*]=s10.y & rr[1,1,*,*,*]=s11.y & rr[2,1,*,*,*]=s12.y
rr[0,2,*,*,*]=s20.y & rr[1,2,*,*,*]=s21.y & rr[2,2,*,*,*]=s22.y
```

> help,rr
RR DOUBLE =
Array[3, 3, 10806, 132, 2]

2. 磁場座標系への回転行列

2.1 MGF 64Hzデータを，スペクトルマトリクスの時刻タグに合わせて内挿します

```
; *****************
; analyze MGF data
; *****************
split_vec, 'erg_mgf_l2_mag_64hz_sgi'

tinterpol_mxn, 'erg_mgf_l2_mag_64hz_sgi_x', pr_matrix+'Bx_Bx_132'
tinterpol_mxn, 'erg_mgf_l2_mag_64hz_sgi_y', pr_matrix+'Bx_Bx_132'
tinterpol_mxn, 'erg_mgf_l2_mag_64hz_sgi_z', pr_matrix+'Bx_Bx_132'
```

Tips: データを内挿する (tinterpol_mxn)

tinterpol_mxn, tplot変数1, tplot変数2, newname=tplot変数3

tplot変数2の時刻タグに合わせて，tplot変数1のデータを内挿する

内挿されたデータはtplot変数3に格納される
newnameを省略した場合は，'元のtplot変数名_interp'という名前の変数に格納される

```
get_data, 'erg_mgf_l2_mag_64hz_sgi_x_interp', data=data_x, dlim=dlim_x, lim=lim_x
get_data, 'erg_mgf_l2_mag_64hz_sgi_y_interp', data=data_y, dlim=dlim_y, lim=lim_y
get_data, 'erg_mgf_l2_mag_64hz_sgi_z_interp', data=data_z, dlim=dlim_z, lim=lim_z
```

2. 磁場座標系への回転行列

2.2 内挿したMGF 64Hzデータから,
SGI座標系 → 背景磁場座標系(B_0 // z) の変換行列を求めます

```
rotmat=dblarr(3,3,n_elements(data_x.x))
rotmat_t=dblarr(3,3,n_elements(data_x.x))

for i=0, n_elements(data_x.x)-1 do begin

  bvec=[data_x.y[i],data_y.y[i],data_z.y[i]]
  zz=[0.,0.,1.]

  yhat=crossp(zz,bvec)
  xhat=crossp(yhat,bvec)
  zhat=bvec

  xhat=xhat/sqrt(xhat[0]^2+xhat[1]^2+xhat[2]^2)
  yhat=yhat/sqrt(yhat[0]^2+yhat[1]^2+yhat[2]^2)
  zhat=zhat/sqrt(zhat[0]^2+zhat[1]^2+zhat[2]^2)

  rotmat[*,*,i]=([[xhat],[yhat],[zhat]])
  rotmat_t[*,*,i]=transpose([[xhat],[yhat],[zhat]])
endfor
```

3. 磁場座標系への回転

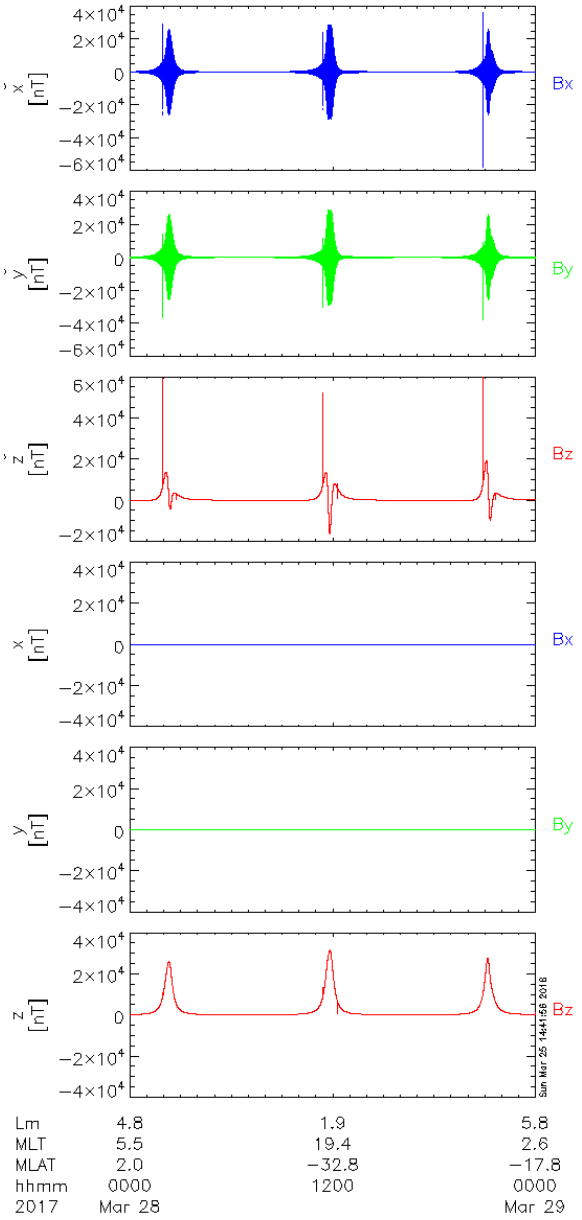
3.1 求めた回転行列を確認するために，MGFデータ(SGI座標系)を磁場座標系に変換してみます

Tips: 行列の積
(A # B, A ## B)

C = A # B → C = BA
C = A ## B → C = AB

(3.1より)
get_data, 'mag_64hz_sgi_x_interp', data=data_x
get_data, 'mag_64hz_sgi_y_interp', data=data_y
get_data, 'mag_64hz_sgi_z_interp', data=data_z

```
; MGF rotation
data_rot=dblarr(n_elements(data_x.x), 3)

for i=0, n_elements(data_x.x)-1 do begin

  data=[[data_x.y[i]], [data_y.y[i]],[data_z.y[i]]]
  data_rot[i,*] = rotmat[*,*,i] ## data

endfor

store_data, 'erg_mgf_l2_mag_64hz_sgi_rot_x',$
 data={x:data_x.x, y:data_rot[*,0]}, dlim=dlim_x, lim=lim_x
store_data, 'erg_mgf_l2_mag_64hz_sgi_rot_y',$
 data={x:data_y.x, y:data_rot[*,1]}, dlim=dlim_y, lim=lim_y
store_data, 'erg_mgf_l2_mag_64hz_sgi_rot_z',$
 data={x:data_z.x, y:data_rot[*,2]}, dlim=dlim_z, lim=lim_z

ylim, 'erg_mgf_l2_mag_64hz_sgi_rot_?', -1E4, 4E4, 0
tplot, [ 'erg_mgf_l2_mag_64hz_sgi_' + ['x', 'y', 'z'], $
      'erg_mgf_l2_mag_64hz_sgi_rot_' + ['x', 'y', 'z'] ]
```



3. 磁場座標系への回転

3.2 求めた回転行列を用いて,
OFA-MATRIXデータ(SGI座標系)を背景磁場座標系(B_0 // z)に変換します

```
; MATRIX ROTATION
for i=0, n_elements(s00.x)-1 do begin
  for j=0, n_elements(s00.v)-1 do begin
    for k=0, 1 do begin

      rr[*,*,i,j,k] = (rotmat[*,*,i] ## rr[*,*,i,j,k]) ## rotmat_t[*,*,i]

    endfor
  endfor
endfor
```

Tips: 行列の積 (A # B, A ## B)

C = A # B → C = BA
C = A ## B → C = AB

分散共分散行列の定義
$\boldsymbol{C = E(XX^T)}$

回転行列を$\boldsymbol{R}$
$\boldsymbol{X' = RX}$ とする

$$\begin{aligned} \boldsymbol{C'} &= \boldsymbol{E\left(X'X'^T\right)} \\ &= \boldsymbol{E(RXX^TR^T)} \\ &= \boldsymbol{RE(XX^T)R^T} \\ &= \boldsymbol{RCR^T} \end{aligned}$$

4. パワースペクトルをプロット

4.1 OFA-MATRIXデータ(背景磁場座標系)を読み込み，複素数を分解します
スペクトルマトリクスの対角成分は，各軸のパワースペクトルを表します
（対角成分のimaginary partは常にゼロ）

$$S = \begin{bmatrix} < B_x B_x^* > & < B_x B_y^* > & < B_x B_z^* > \\ < B_y B_x^* > & < B_y B_y^* > & < B_y B_z^* > \\ < B_z B_x^* > & < B_z B_y^* > & < B_z B_z^* > \end{bmatrix}$$

$$= \begin{bmatrix} |B_x|^2 & |B_x||B_y|e^{i(\phi_x-\phi_y)} & |B_x||B_z|e^{i(\phi_x-\phi_z)} \\ |B_y||B_x|e^{-i(\phi_x-\phi_y)} & |B_y|^2 & |B_y||B_z|e^{i(\phi_y-\phi_z)} \\ |B_z||B_x|e^{-i(\phi_x-\phi_z)} & |B_z||B_y|e^{-i(\phi_y-\phi_z)} & |B_z|^2 \end{bmatrix}$$

> help, data
** Structure <155fc1b0>, 3 tags, length=11489600, data length=11489600, refs=1:

X	DOUBLE	Array[10806]	*(時刻情報)*
Y	FLOAT	Array[10806, 132, 2]	*(クロススペクトル)*
V	FLOAT	Array[132]	*(第二軸のラベル)*

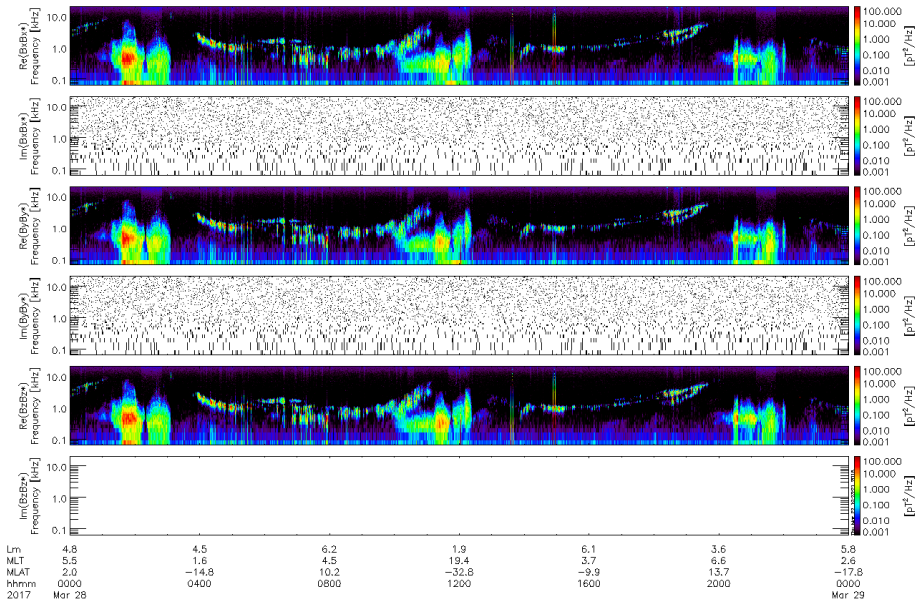
data.y[*,*,0]→ real part
data.y[*,*,1]→ imaginary part

```
; ************************************
; auto-spectra
; ************************************
get_data, pr_matrix + 'Bx_Bx_132',data=data, dlim=dlim, lim=lim
store_data, pr_matrix + 'Bx_Bx_re', data={x:data.x,y:data.y[*,*,0],v:data.v}, dlim=dlim, lim=lim
store_data, pr_matrix + 'Bx_Bx_im', data={x:data.x,y:data.y[*,*,1],v:data.v}, dlim=dlim, lim=lim
;
get_data, pr_matrix + 'By_By_132',data=data, dlim=dlim, lim=lim
store_data, pr_matrix + 'By_By_re', data={x:data.x,y:data.y[*,*,0],v:data.v}, dlim=dlim, lim=lim
store_data, pr_matrix + 'By_By_im', data={x:data.x,y:data.y[*,*,1],v:data.v}, dlim=dlim, lim=lim
;
get_data, pr_matrix + 'Bz_Bz_132',data=data, dlim=dlim, lim=lim
store_data, pr_matrix + 'Bz_Bz_re', data={x:data.x,y:data.y[*,*,0],v:data.v}, dlim=dlim, lim=lim
store_data, pr_matrix + 'Bz_Bz_im', data={x:data.x,y:data.y[*,*,1],v:data.v}, dlim=dlim, lim=lim
```

4. パワースペクトルをプロット

4.2 OFA-MATRIXデータ(背景磁場座標系)を使って，パワースペクトルをプロットします

```
options, pr_matrix + 'Bx_Bx_re', ytitle='Re(BxBx*)'
options, pr_matrix + 'Bx_Bx_im', ytitle='Im(BxBx*)'
options, pr_matrix + 'By_By_re', ytitle='Re(ByBy*)'
options, pr_matrix + 'By_By_im', ytitle='Im(ByBy*)'
options, pr_matrix + 'Bz_Bz_re', ytitle='Re(BzBz*)'
options, pr_matrix + 'Bz_Bz_im', ytitle='Im(BzBz*)'
options, pr_matrix + 'B?_B?_??', ysubtitle = 'Frequency [kHz]'
options, pr_matrix + 'B?_B?_??', ztitle = '[pT!U2!N/Hz]'
ylim, pr_matrix + 'B?_B?_??', 0.064, 20, 1  ; kHz
zlim, pr_matrix + 'B?_B?_??', 1E-3, 1E2, 1  ; nT

tplot, pr_matrix + ['Bx_Bx_re', 'Bx_Bx_im', 'By_By_re', 'By_By_im', 'Bz_Bz_re', 'Bz_Bz_im']
```



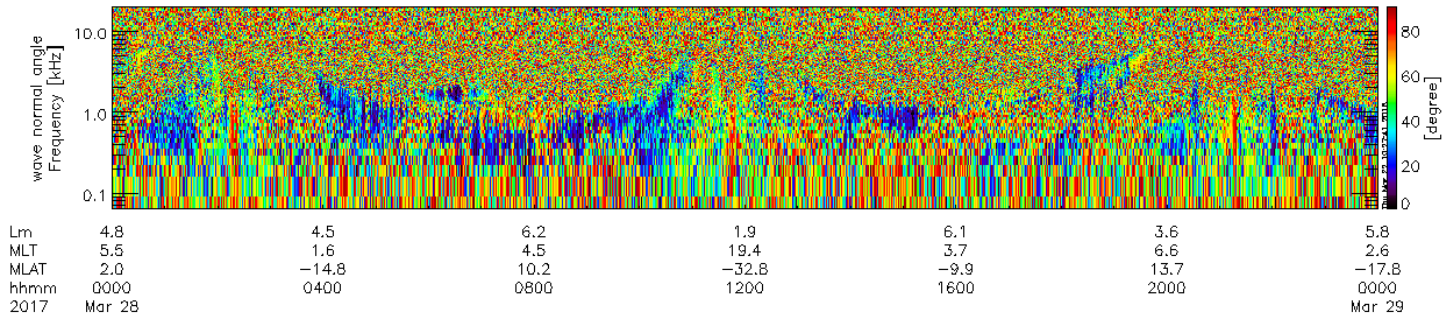
5. Means' methodによるwave normal angle推定

5.1 OFA-MATRIXデータ(背景磁場座標系)にMeans' methodを適用します

```
; ***********************************
; wave normal angle
; ***********************************
wna=dblarr(n_elements(s00.x), n_elements(s00.v))

for i=0, n_elements(s00.x)-1 do begin
  for j=0, n_elements(s00.v)-1 do begin
    ; wave normal
    wna_x = rr[2,1,i,j,1] / sqrt(rr[2,1,i,j,1]^2 + rr[0,2,i,j,1]^2 + rr[1,0,i,j,1]^2)
    wna_y = rr[0,2,i,j,1] / sqrt(rr[2,1,i,j,1]^2 + rr[0,2,i,j,1]^2 + rr[1,0,i,j,1]^2)
    wna_z = rr[1,0,i,j,1] / sqrt(rr[2,1,i,j,1]^2 + rr[0,2,i,j,1]^2 + rr[1,0,i,j,1]^2)

    wna[i,j] = abs(atan(sqrt(wna_x^2+wna_y^2)/wna_z)/!dtor)
  endfor
endfor

store_data, 'kvec', data={x:s00.x, y:wna, v:s00.v}

options, 'kvec', ytitle='wave normal angle', $
         ztitle='[degree]', ysubtitle='Frequency [kHz]', spec = 1

ylim, 'kvec', 0.064, 20, 1 ; kHz
zlim, 'kvec', 0., 90, 0    ; degree

tplot, [pr_matrix+'Bx_Bx_re', pr_matrix+'By_By_re', pr_matrix+'Bz_Bz_re', 'kvec']
```

Means et al. (1972)

$$k \parallel \begin{pmatrix} \mathrm{Im}(S_{YZ}) \\ \mathrm{Im}(S_{ZX}) \\ \mathrm{Im}(S_{XY}) \end{pmatrix}$$



6. Polarizationを計算

6.1 OFA-MATRIXデータ(背景磁場座標系)から，偏波(polarization)を計算します

```
; ***********************************
; polarization
; ***********************************
Polarization = $
  dblarr(n_elements(s00.x),n_elements(s00.v))

for i=0, n_elements(s00.x)-1 do begin
  for j=0, n_elements(s00.v)-1 do begin
    polarization[i,j] = $
          (2 * rr[1,0,i,j,1]) / (rr[0,0,i,j,0] + rr[1,1,i,j,0])
  endfor
endfor

store_data, 'polarization',data={x:s00.x, y:polarization, v:s00.v}

options, 'polarization', ytitle='polarization', ysubtitle='Frequency [kHz]', spec = 1

ylim, 'polarization', 0.064, 20., 1 ; kHz
zlim, 'polarization', -1., 1., 0    ;

tplot, [pr_matrix+'Bx_Bx_re', pr_matrix+'By_By_re', pr_matrix+'Bz_Bz_re', 'polarization']
```

磁力線に対して垂直な面における信号を，次のようにおく

$$H_x(t) = a_x(t)e^{i(wt+\phi_x(t))}$$
$$H_y(t) = a_y(t)e^{i(wt+\phi_y(t))}$$

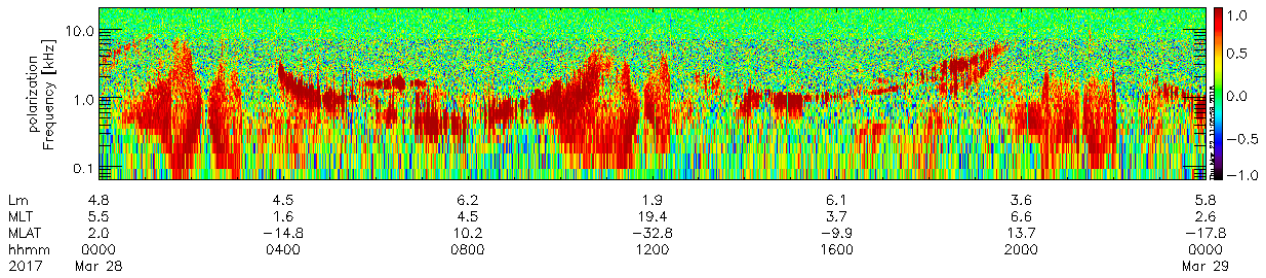
スペクトルマトリクスは

$$S = \begin{bmatrix} < H_x H_x^{*} > & < H_x H_y^{*} > \\ < H_y H_x^{*} > & < H_y H_y^{*} > \end{bmatrix} = \begin{bmatrix} < a_x{}^2 > & < a_x a_y e^{i(\phi_x-\phi_y)} > \\ < a_x a_y e^{-i(\phi_x-\phi_y)} & < a_y{}^2 > \end{bmatrix}$$

$$a_x a_y e^{i(\phi_x-\phi_y)} = a_x a_y \cos(\phi_x-\phi_y) + i a_x a_y \sin(\phi_x-\phi_y)$$

偏波率ηは

$$\eta = \frac{< 2a_x\, a_y \sin(\phi_x-\phi_y) >}{< a_x{}^2 > + < a_y{}^2 >} = \frac{2\mathrm{Im}(s_{10})}{s_{00}+s_{11}}$$



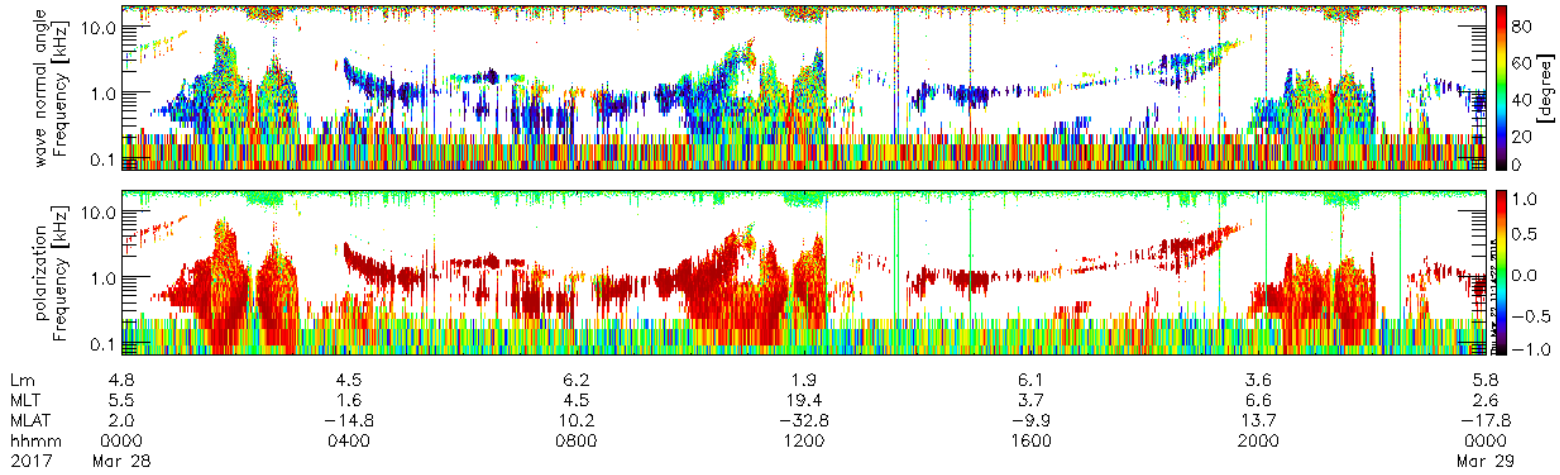
7. 解析結果をマスク

7.1　磁場のパワースペクトル(Btotal_132)を使って，解析結果をマスクします

```
; ************************************
; mask
; ************************************
get_data, pr_matrix + 'B_total_132', data=data_ref

; kvec
get_data, 'kvec', data=data, dlim=dlim, lim=lim
data.y[where(data_ref.y LT 1E-2)] = 'NaN'
store_data, 'kvec_mask', data={x:data.x, y:data.y, v:data.v}, dlim=dlim, lim=lim

; polarization
get_data, 'polarization', data=data, dlim=dlim, lim=lim
data.y[where(data_ref.y LT 1E-2)] = 'NaN'
store_data, 'polarization_mask', data={x:data.x, y:data.y, v:data.v}, dlim=dlim, lim=lim

tplot, [pr_matrix + 'Bx_Bx_re', pr_matrix + 'By_By_re', pr_matrix + 'Bz_Bz_re', $
        'kvec_mask', 'polarization_mask']
```

Tips: 条件を満たす配列要素の番号を返す (where)

ret = where(配列 <演算子> 条件)
配列の中から，引数として与えられる条件式を満たす要素の番号を，配列として返す



8. まとめてプロット

8.1　MGFデータから電子サイクロトロン周波数を計算して，重ねてプロットします

```
; ***********************************
; overplot fce
; ***********************************
get_data, 'erg_mgf_l2_magt_8sec', data=data
fce = data.y/ 10^(9.) * 1.6 * 10^(-19.) / (9.1093D * 10^(-31.)) / 2. / !pi / 1000.
fce_half = fce / 2.

store_data, 'fce', data={x:data.x, y:fce}
store_data, 'fce_half', data={x:data.x, y:fce_half}

options, 'fce', colors=fsc_color('yellow'), thick=2
options, 'fce_half', colors=fsc_color('magenta'), thick=2
```

Tips: 複数のtplot変数をマージしたtplot変数を作る (store_data)

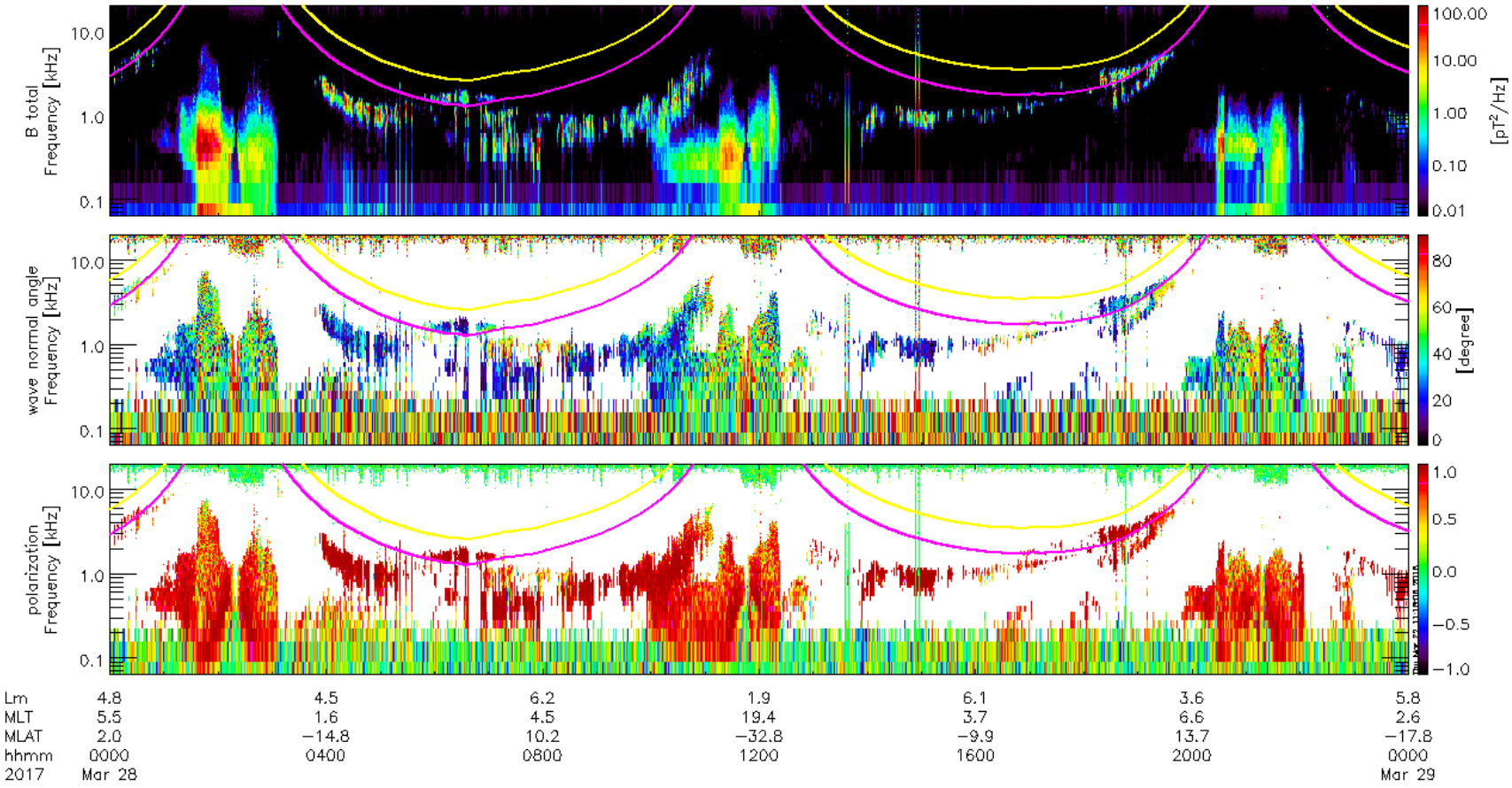
store_data, tplot変数1, data=[tplot変数2, tplot変数3]

tplot変数2とtplot変数3をマージして，tplot変数1に格納する．
といっても，tplot変数1の実体は 'tplot変数2, tplot変数3' という単なる文字列

```
store_data, pr_matrix + 'Btotal_132_gyro', $
            data=[pr_matrix + 'Btotal_132', 'fce', 'fce_half']
store_data, 'kvec_mask_gyro', data=['kvec_mask', 'fce', 'fce_half']
store_data, 'polarization_mask_gyro', data=['polarization_mask', 'fce', 'fce_half']

ylim, '*_gyro', 0.064, 20, 1 ; kHz
zlim, pr_matrix + 'Btotal_132_gyro', 1E-2, 1E2, 1 ; pT^2/Hz
options, 'erg_pwe_ofa_l2_Btotal_132_gyro', $
         ytitle='B total', ysubtitle='Frequency [kHz]', ztitle='[pT!U2!N/Hz]'
tplot, [pr_matrix + 'Btotal_132', 'kvec_mask', 'polarization_mask'] + '_gyro'
```

8. まとめてプロット

8.1　MGFデータから電子サイクロトロン周波数を計算して，重ねてプロットします



Memo

SPEDAS training session (28 Mar. 2018)

Advanced course [2 of 2]

Arase/PWE OFA-COMPLEX analysis

1. CDF読み込み

- PWE/OFA-COMPLEX L2(pre)
- MGF 64Hz L2(pre)

2. 未測定のEzを推定

- E･B=0を仮定

3. Poynting Vectorを計算する

4. SGI座標系→磁場座標系への変換行列を計算する

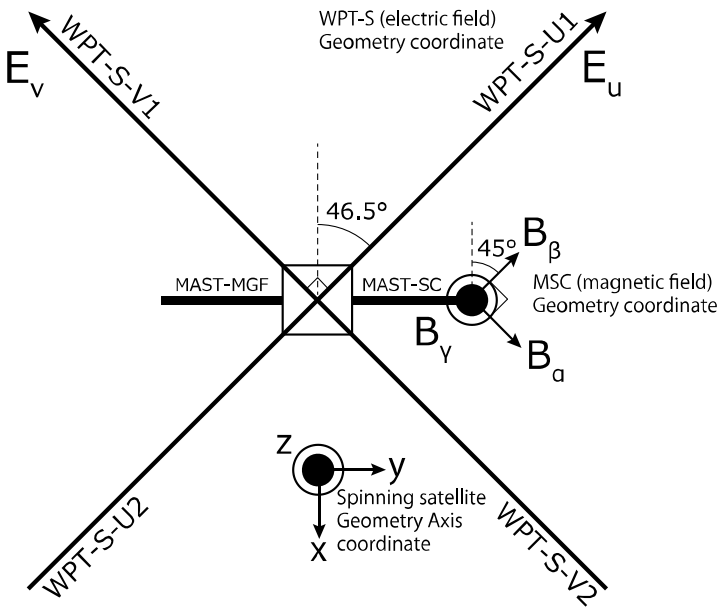
5. 求めたベクトルを磁場座標系に変換する

6. 磁場とのなす角を計算してプロットする

7. 解析結果をパワーでマスク

8. サイクロトロン周波数をオーバープロット

- MGF 8sec L2 CDFを読み込み



1. 必要なファイルの読み込み

1.1 PWE OFA-COMPLEX(複素スペクトル) CDFを取得します

```
; ******************
; load OFA-COMPLEX
; ******************
erg_load_pwe_ofa, datatype='complex', uname=uname, pass=pass
pr_complex = 'erg_pwe_ofa_complex_l2_'
```

```
> tplot_names
 ...
 92 erg_pwe_ofa_complex_l2_Ex_132
 93 erg_pwe_ofa_complex_l2_Ey_132
 94 erg_pwe_ofa_complex_l2_Etotal_132
 95 erg_pwe_ofa_complex_l2_Bx_132
 96 erg_pwe_ofa_complex_l2_By_132
 97 erg_pwe_ofa_complex_l2_Bz_132
 98 erg_pwe_ofa_complex_l2_Btotal_132
```

```
> help, data_ex
** Structure <184f9730>, 3 tags, length=11488536,
data length=11488536, refs=1:
   X       DOUBLE  Array[10803]          (時刻情報)
   Y       FLOAT   Array[10803, 132, 2] (複素スペクトル)
   V       FLOAT   Array[132]           (第二軸のラベル)
```

```
get_data, pr_complex + 'Ex_132', data=data_ex, dlim=dlim, lim=lim
c_ex = dcomplex(data_ex.y[*,*,0]*1E-3, data_ex.y[*,*,1]*1E-3)

get_data, pr_complex + 'Ey_132', data=data, dlim=dlim, lim=lim
c_ey = dcomplex(data.y[*,*,0]*1E-3, data.y[*,*,1]*1E-3)

get_data, pr_complex + 'Bx_132', data=data_bx, dlim=dlim, lim=lim
c_bx = dcomplex(data_bx.y[*,*,0]*1E-12, data_bx.y[*,*,1]*1E-12)

get_data, pr_complex + 'By_132', data=data, dlim=dlim, lim=lim
c_by = dcomplex(data.y[*,*,0]*1E-12, data.y[*,*,1]*1E-12)

get_data, pr_complex + 'Bz_132', data=data, dlim=dlim, lim=lim
c_bz = dcomplex(data.y[*,*,0]*1E-12, data.y[*,*,1]*1E-12)
```

2. Ezを推定する

2.1 未測定の電界成分 (Ez)を推定します.

```
; *****************
; calc Ez
; *****************
c_ez = dcindgen(n_elements(data_ex.x), n_elements(data_ex.v))

idx = nn(data_bx.x, data_ex.x)


for i=0, n_elements(data_ex.x)-1 do begin

  c_ez[i,*] = (-c_ex[i,*]*c_bx[idx[i],*] $
               - c_ey[i,*]*c_by[idx[i],*]) / c_bz[idx[i],*]

endfor
```

Tips:指定した時刻に最も近い時刻タグの配列番号を調べる (nn)

idx = nn(data, time)
data: データ構造体，またはtplot変数名
time: Unix time形式の時刻.
　　データ構造体の中で，この時刻に最も近いデータの配列番号がidxに返る.

$E \cdot B = 0$を仮定する

$$E_z = \frac{-E_x B_x - E_y B_y}{B_z}$$

3. Poynting vectorを計算する

```
Sx=dindgen(n_elements(data_bx.x),n_elements(data_bx.v))
Sy=dindgen(n_elements(data_bx.x),n_elements(data_bx.v))
Sz=dindgen(n_elements(data_bx.x),n_elements(data_bx.v))

; calc Poynting flux
idx = nn(data_ex.x, data_bx.x)
```

Tips:指定した時刻に最も近い時刻タグの配列番号を調べる (nn)

idx = nn(data, time)
data: データ構造体，またはtplot変数名
time: Unix time形式の時刻.
　　データ構造体の中で，この時刻に最も近いデータの配列番号がidxに返る.

Poynting vectorの定義

$$S = \frac{1}{2}(E \times H^*)$$

$$\parallel \begin{pmatrix} E_y {B_z}^* - E_z {B_y}^* \\ -E_x {B_z}^* - E_z {B_x}^* \\ E_x {B_y}^* - E_y {B_x}^* \end{pmatrix}$$

```
for i=0, n_elements(data_bx.x)-1 do begin
  for j=0, n_elements(data_bx.v)-1 do begin

    Sx[i,j]= double(c_ey[idx[i],j]*conj(c_bz[i,j])-c_ez[idx[i],j]*conj(c_by[i,j]))
    Sy[i,j]=-double(c_ex[idx[i],j]*conj(c_bz[i,j])-c_ez[idx[i],j]*conj(c_bx[i,j]))
    Sz[i,j]= double(c_ex[idx[i],j]*conj(c_by[i,j])-c_ey[idx[i],j]*conj(c_bx[i,j]))

  endfor
endfor
```

4. 磁場座標系への回転行列

4.1 MGF 64Hzデータを，複素スペクトルの時刻タグに合わせて内挿します

```
; ******************
; analyze MGF data
; ******************
split_vec, 'erg_mgf_l2_mag_64hz_sgi'

; interpolation
tinterpol_mxn, 'erg_mgf_l2_mag_64hz_sgi_x', pr_complex + 'Bx_132'
tinterpol_mxn, 'erg_mgf_l2_mag_64hz_sgi_y', pr_complex + 'Bx_132'
tinterpol_mxn, 'erg_mgf_l2_mag_64hz_sgi_z', pr_complex + 'Bx_132'

get_data, 'erg_mgf_l2_mag_64hz_sgi_x_interp', data=data_x
get_data, 'erg_mgf_l2_mag_64hz_sgi_y_interp', data=data_y
get_data, 'erg_mgf_l2_mag_64hz_sgi_z_interp', data=data_z
```

Tips: データを内挿する (tinterpol_mxn)

tinterpol_mxn, tplot変数1, tplot変数2, newname=tplot変数3

tplot変数2の時刻タグに合わせて，tplot変数1のデータを内挿する
内挿されたデータはtplot変数3に格納される

4. 磁場座標系への回転行列

4.2 内挿したMGF 64Hzデータから,
SGI座標系 → 背景磁場座標系(B_0 // z) の変換行列を求めます

```
; rotation matrix
rotmat=dblarr(3,3,n_elements(data_x.x))
rotmat_t=dblarr(3,3,n_elements(data_x.x))

for i=0, n_elements(data_x.x)-1 do begin

  bvec=[data_x.y[i],data_y.y[i],data_z.y[i]]
  zz=[0.,0.,1.]

  yhat=crossp(zz,bvec)
  xhat=crossp(yhat,bvec)
  zhat=bvec

  xhat=xhat/sqrt(xhat[0]^2+xhat[1]^2+xhat[2]^2)
  yhat=yhat/sqrt(yhat[0]^2+yhat[1]^2+yhat[2]^2)
  zhat=zhat/sqrt(zhat[0]^2+zhat[1]^2+zhat[2]^2)

  rotmat[*,*,i]=([[xhat],[yhat],[zhat]])
  rotmat_t[*,*,i]=transpose([[xhat],[yhat],[zhat]])
endfor
```

5. ベクトルを磁場座標系に変換する

5.1 ベクトルを SGI座標系 → 背景磁場座標系(B_0 // z) に変換し，
背景磁場に対する角度を計算します

```
; ******************
; Poynting vector calculation
; ******************
S=dindgen(n_elements(data_bx.x),n_elements(data_bx.v),3)

for i=0, n_elements(data_bx.x)-1 do begin
  for j=0, n_elements(data_bx.v)-1 do begin

      S[i,j,*] = rotmat[*,*,i] ## [Sx[i,j],Sy[i,j],Sz[i,j]]

  endfor
endfor
```

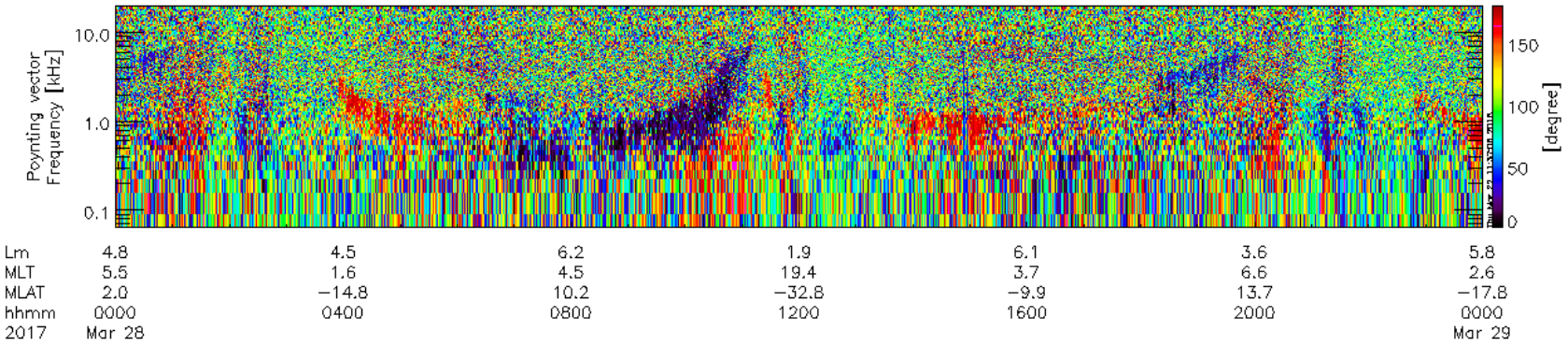
Tips: 行列の積 (A # B, A ## B)

C = A # B → C = BA
C = A ## B → C = AB

6. 磁場とのなす角を計算する

6.1 ベクトルの背景磁場に対する角度を計算します

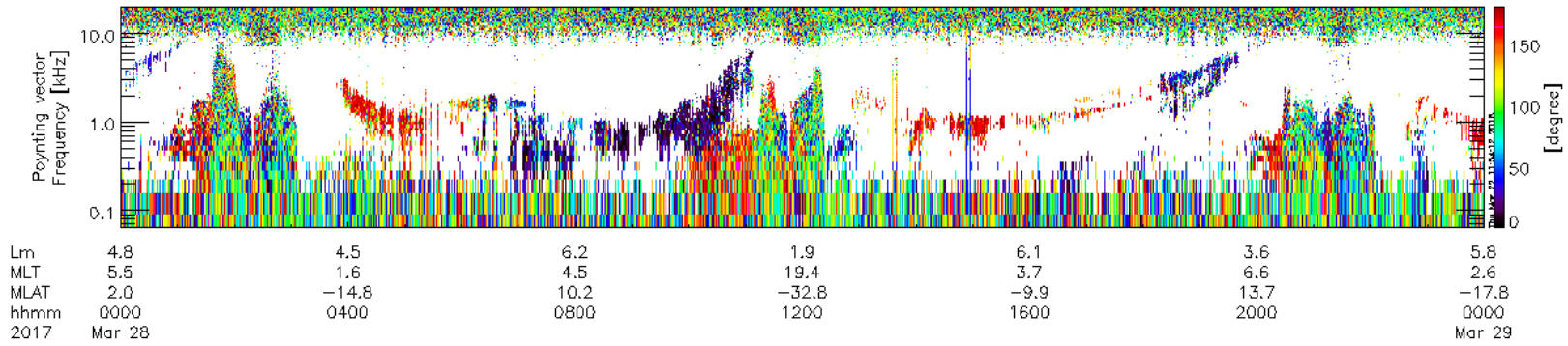
```
theta=acos(S[*,*,2]/sqrt(S[*,*,0]^2+S[*,*,1]^2+S[*,*,2]^2))/!dtor

store_data, 'S', data={x:data_bx.x, y:theta, v:data_bx.v}, dlim=dlim, lim=lim

ylim, [pr_complex + 'Etotal_132', pr_complex + 'Btotal_132', 'S'], 0.064, 20, 1
zlim, pr_complex + 'Etotal_132', 1E-7, 1E0, 1 ; mV^2/m^2/Hz
zlim, pr_complex + 'Btotal_132', 1E-2, 1E2, 1 ; pT^2/Hz
zlim, 'S', 0., 180., 0 ; degree

options, [pr_complex + 'Etotal_132', pr_complex+'Btotal_132', 'S'], $
                                        ysubtitle='Frequency [kHz]'
options, pr_complex + 'Etotal_132', ytitle='E total', ztitle='[mV!U2!N/m!U2!N/Hz]'
options, pr_complex + 'Btotal_132', ytitle='B total', ztitle='[pT!U2!N/Hz]'
options, 'S', ytitle='Poynting vector', ztitle='[degree]'

tplot, [pr_complex + 'Etotal_132', pr_complex + 'Btotal_132', 'S']
```



7. 解析結果をマスク

7.1　磁場のパワースペクトル(Btotal_132)を使って，解析結果をマスクします

```
; ************************************
; mask
; ************************************
get_data, pr_complex + 'Btotal_132', data=data_ref

; S
get_data, 'S', data=data, dlim=dlim, lim=lim
data.y[where(data_ref.y LT 1E-2)] = 'NaN'
store_data, 'S_mask', data={x:data.x, y:data.y, v:data.v}, dlim=dlim, lim=lim

tplot, [pr_complex + 'Etotal_132', pr_complex + 'Btotal_132', 'S_mask']
```



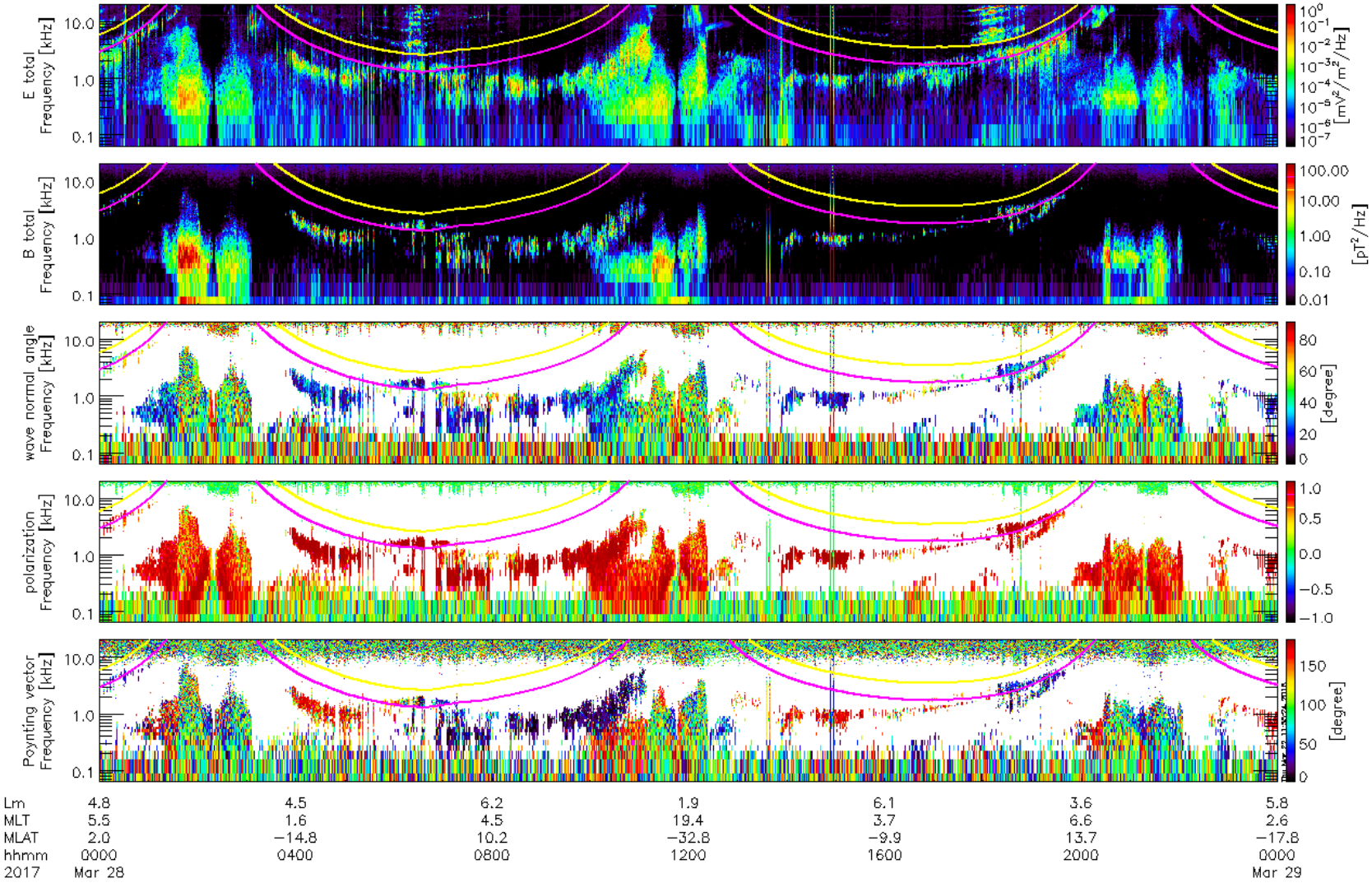
8. まとめてプロット

8.1　MGFデータから電子サイクロトロン周波数を計算して，重ねてプロットします

```
; ***********************************
; overplot fce
; ***********************************
store_data, pr_complex+'Etotal_132_gyro', $
             data=[pr_complex+ 'Etotal_132', 'fce', 'fce_half']
store_data, pr_complex+ 'Btotal_132_gyro', $
             data=[pr_complex+ 'Btotal_132', 'fce', 'fce_half']
store_data, 'S_mask_gyro', data=['S_mask', 'fce', 'fce_half']

ylim, '*_gyro', 0.064, 20, 1 ; kHz
zlim, 'erg_pwe_ofa_l2_Etotal_132_gyro', 1E-7, 1E0, 1 ; mV^2/m^2/Hz
zlim, 'erg_pwe_ofa_l2_Btotal_132_gyro', 1E-2, 1E2, 1 ; pT^2/Hz
options, pr_complex+'Etotal_132_gyro', $
         ytitle='E total', ysubtitle='Frequency [kHz]', ztitle='[mV!U2!N/m!U2!N/Hz]'
options, pr_complex+'Btotal_132_gyro', $
         ytitle='B total', ysubtitle='Frequency [kHz]', ztitle='[pT!U2!N/Hz]'

tplot, [pr_complex+'Etotal_132', pr_complex+'Btotal_132', $
        'kvec_mask', 'polarization_mask', 'S_mask'] + '_gyro'
```

8. まとめてプロット

8.1　MGFデータから電子サイクロトロン周波数を計算して，重ねてプロットします



Memo