SPEDAS training session
– Beginner course –

T. Hori, M. Teramoto, T.-F. Chang, Y. Tsugawa, M. Shoji,
S. Kurita, Y. Miyoshi, N. Umemura, T. Segawa
(ERG Science Center, ISEE, Nagoya Univ.)

Purpose of this course

- To get familiar with the SPEDAS scheme.
- To get you ready to use SPEDAS to analyze ERG data.

What will you be doing in this training?

- Learn about the data model employed by SPEDAS.
- Load/manipulate/visualize ERG satellite and other ground data.

Keep in mind

- This is a "hands-on" training for SPEDAS, not the time for e-mail check.
- Communicate with lecturers, tutors, neighboring skilled users.
- The training session today does not cover everything in the handouts due to time limitation. It is recommended to practice the uncovered parts later by yourself.
- We will proceed slowly with entry-level SPEDAS users, particularly in the beginner course. You can go faster by yourself.

What's SPEDAS?

Space Physics Environment Data Analysis Software (SPEDAS)

Data repository

CDF

THEMIS

CDF

Van Allen Probes

CDF

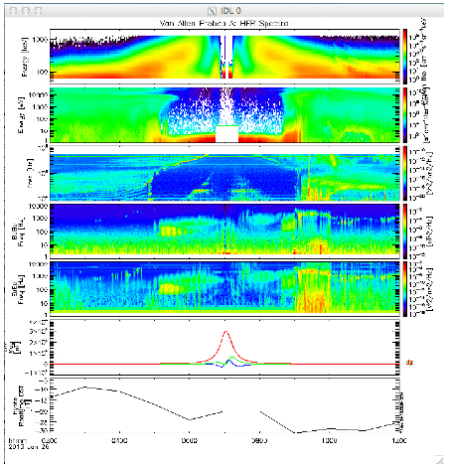
MMS

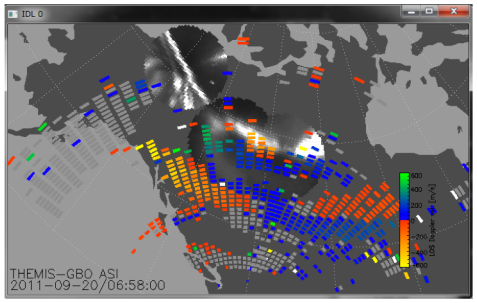
CDF

IUGONET

Each project develops and releases APIs to automatically download and read each project data

SPEDAS framework





ERG

CDF

ERG satellite data

CDF

ERG-related ground observation network data

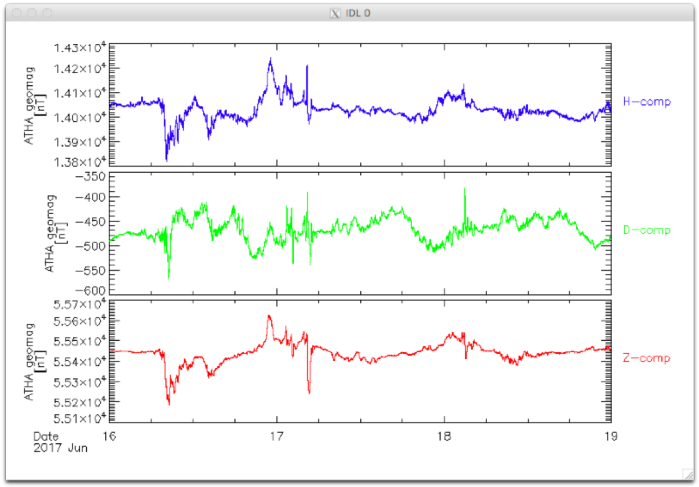
CDF HDF?

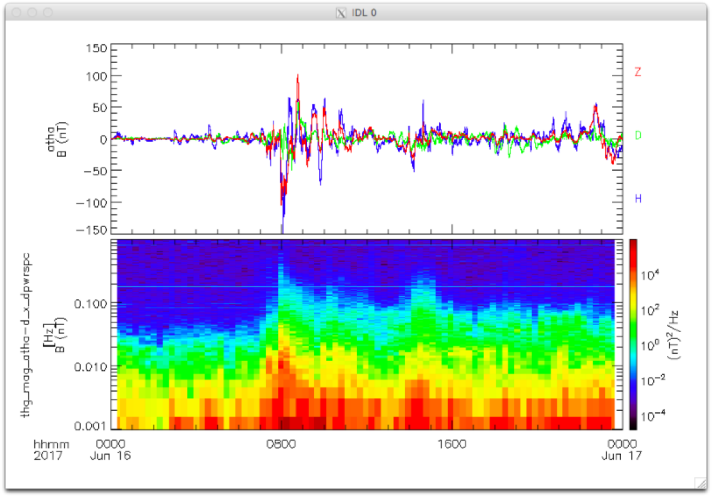
ERG simulation / modeling data

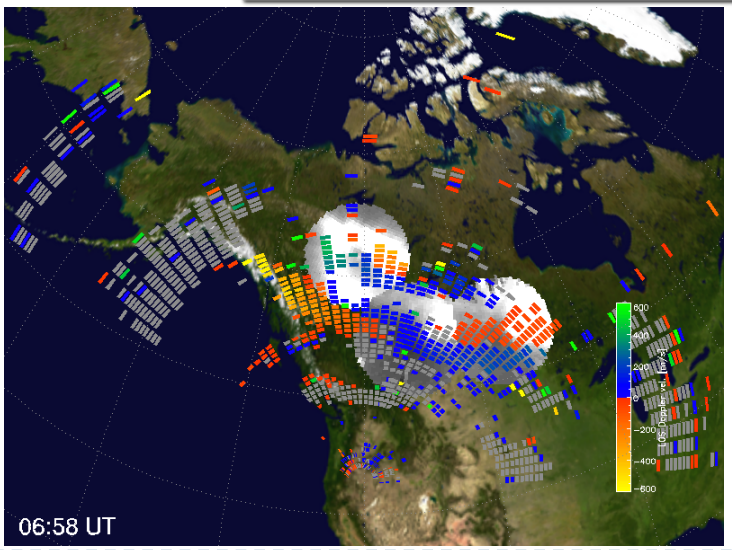
ERG-SC develops data archives and APIs to read the data from SPEDAS

What can you do with SPEDAS?

- time-series plots
- filtering of data
- frequency analysis
- mapping to the ground maps
- …







Basics of SPEDAS: tplot and tplot variable

A few basics of IDL before entering SPEDAS…

- Insert a comma (,) between a command, its arguments, and keywords.

```
IDL> tplot , 1 , title='New plot'
```

- A string is expressed as a text sandwiched by delimeters (') or (").

```
IDL> print, 'This is a text.'
```

- An array is expressed as comma-separated elements that are bracketed.

```
IDL> arr1 = [ 2, 3, 4, 5 ]
IDL> string_arr1 = [ 'text1', 'text2', 'text3' ]
```

- Typical errors beginners often encounter:

```
% Attempt to call undefined procedure: '????'.
```

→ command/routine name (????) is misspelled.

```
% Syntax error.
```

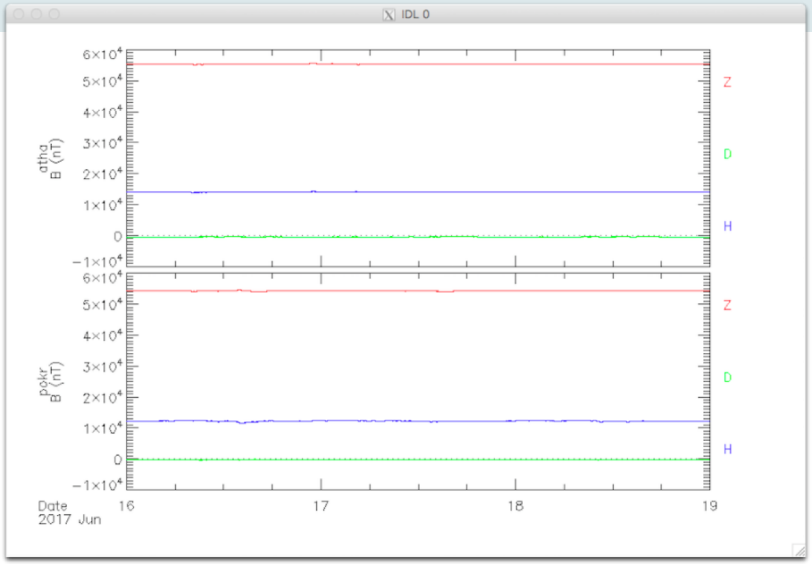
→ , ' ( ) [ ] is missing or mismatched in most cases.

- Use Up arrow key to reuse previously typed commands. You can edit them with Left/Right arrow, Backspace keys and execute!

How SPEDAS works?

One of the simplest procedures would be:

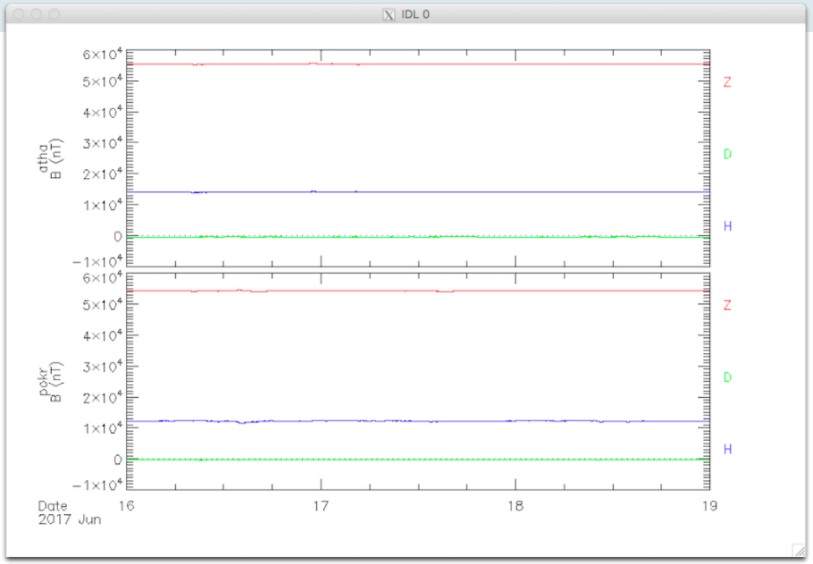
1. Run IDL
2. Initialize the SPEDAS environment on IDL
3. Set a date/time range for which data are loaded.
4. Load data
5. *(Manipulate the loaded data)*
6. Plot the data



How SPEDAS works?

In SPEDAS command lines,

```
prompt> idl
IDL> erg_init
ERG> timespan, '2017-06-16', 3
ERG> thm_load_gmag, site='atha pokr'
   (manipulate tplot variables)
ERG> tplot, [ 'thg_mag_atha', 'thg_mag_pokr' ]
```



Set a date/time range

```
ERG> timespan, timestr, N, option
```

timestr : a string expressing a particular date/time in UTC in the format of 'yyyy-mm-dd/hh:mm:ss'

N : number of time length (Default: 1)

option : unit (/day, /hour, /min, /sec, Default: /day)

For 1 day from 2017-06-16/00:00:00 UTC

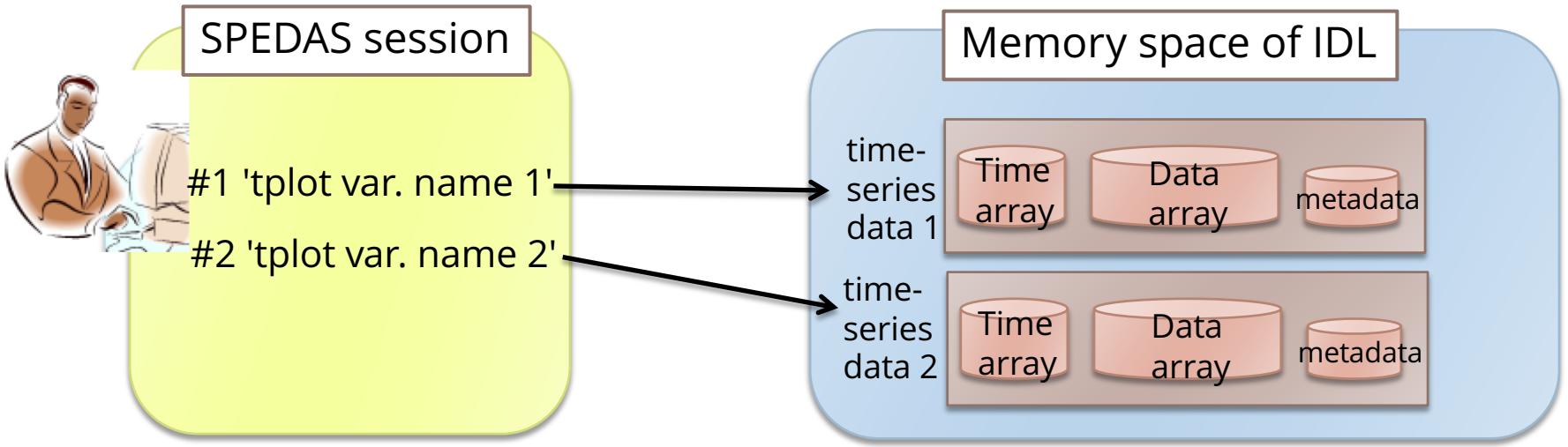
```
ERG> timespan, '2017-06-16'
```

For 10 min from 2017-06-17/01:31:41 UTC

```
ERG> timespan, '2017-06-17/01:31:41', 10, /min
```

Tplot variable as the primary data model

- 'thg_mag_atha' in prev. page is called *tplot variable*.
- "Tplot variables" bind an indexed name-string to a data structure on IDL containing time-series data with metadata.



Listing tplot data & viewing the content

```
ERG> tplot_names

ERG> print_tinfo, 'thg_mag_pokr'
```

```
ERG> tplot_names
   1 thg_mag_atha
   2 thg_mag_pokr
ERG>
```

All tplot variables are listed with unique index numbers

```
ERG> print_tinfo, 'thg_mag_pokr'
*** Variable: thg_mag_pokr
** Structure <2609eb8>, 3 tags, length=5183976, data length=5183972, refs=1:
   X               DOUBLE    Array[259198]
   Y               FLOAT     Array[259198, 3]
   V               LONG      Array[3]
Data format: [thg_mag_pokr_epoch, thg_mag_pokr_compno]
ERG>
```

The actual data structure bound to tplot variable 'thg_mag_pokr' is shown.

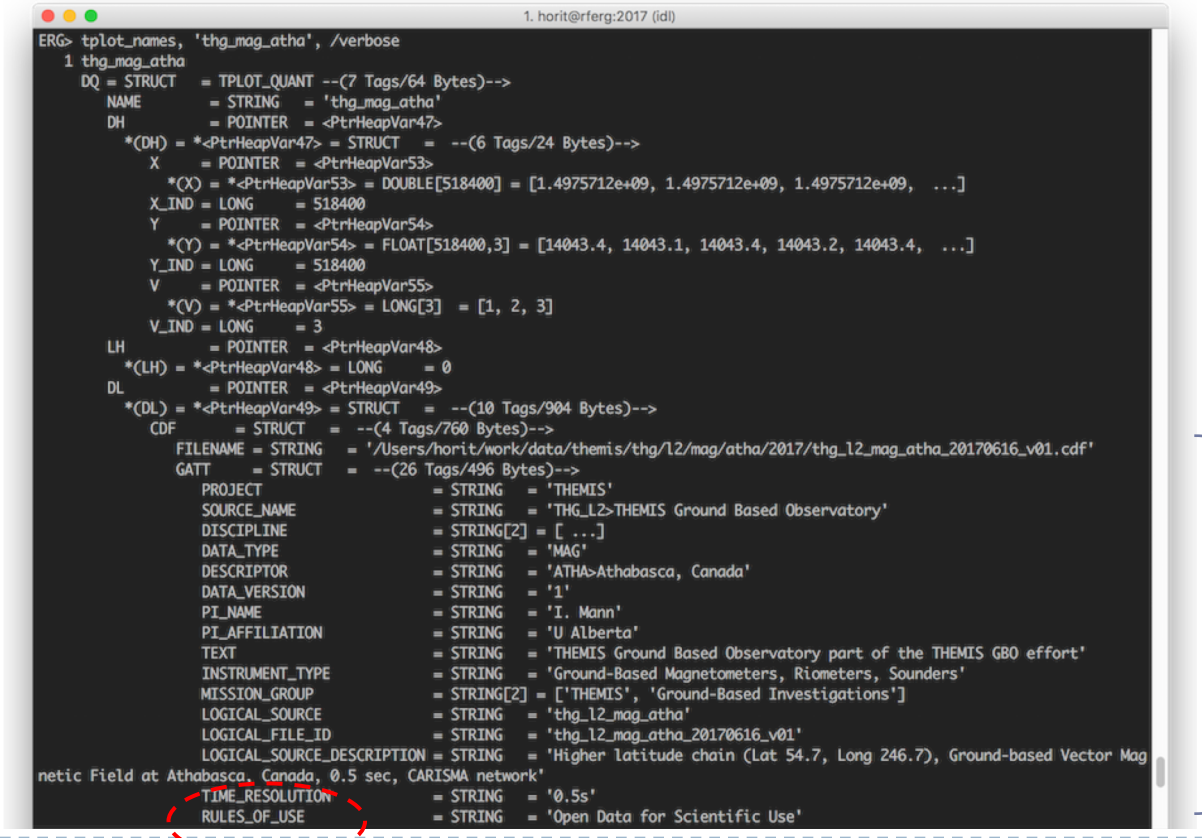
X: time array containing time labels in decimal UNIX time

Y: data array, in this case, a 2-D array of time x 3-components

V: array containing indices of the 3-components, [1 , 2 , 3]

Listing tplot data & viewing the content

```
ERG> tplot_names, 'thg_mag_atha', /verbose
ERG> tplot_names, 1, /v
```



Metadata (information on the data) are dumped.

RULES_OF_USE carries "rules of the road" in using the data.

Plotting a tplot data by *tplot*

Tplot with tplot variable names (string)

```
ERG> tplot, 'thg_mag_atha'
ERG> tplot, [ 'thg_mag_atha' , 'thg_mag_pokr' ]
```

Tplot with the index number of a tplot variable

```
ERG> tplot, 1
```

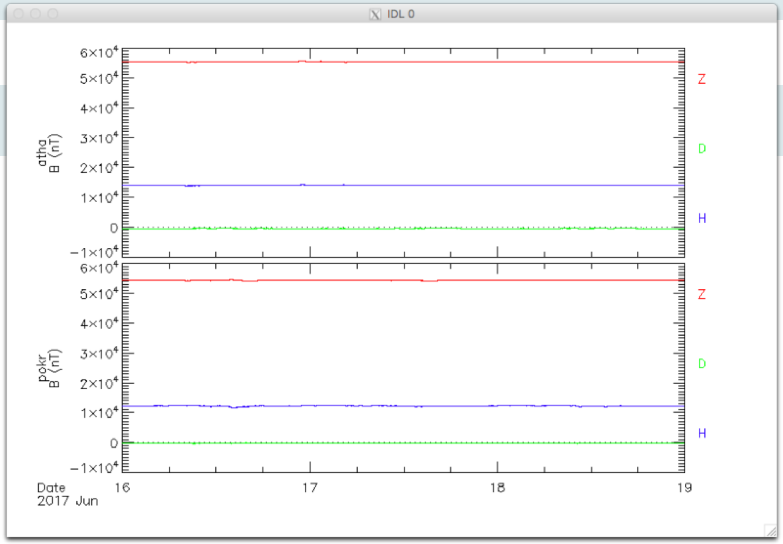
With an array of variable indices to combine multiple variable in a single plot window

```
ERG> tplot, [2,1]
```

Some wildcards can be used

```
ERG> tplot, 'thg_mag_*'
```

Tplot accepts variables as arguments in various formats.



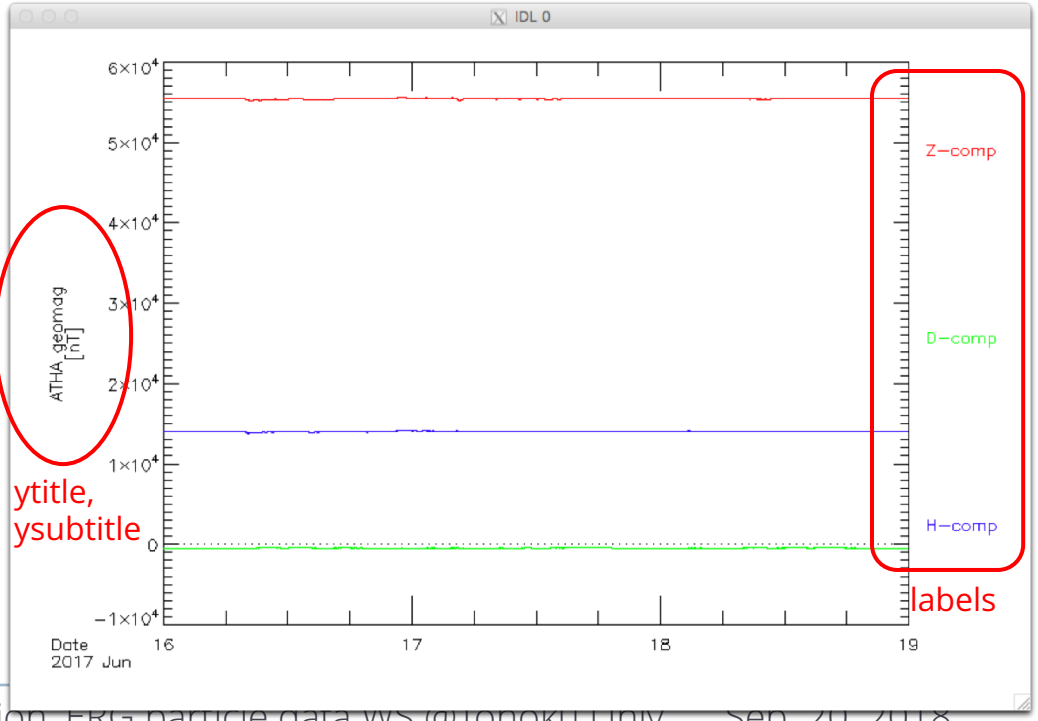
Decorate the plot panel for each tplot variable

options, varname, option1='…', option2='…', …

`varname`: tplot variable name (wildcards accepted)

`option?`: name of tplot variable attribute

```
ERG> options, 'thg_mag_atha', ytitle='ATHA geomag' , ysubtitle='[nT]'
ERG> options, 'thg_mag_????', labels=[ 'H-comp', 'D-comp', 'Z-comp' ]
ERG> tplot, 'thg_mag_atha'
```

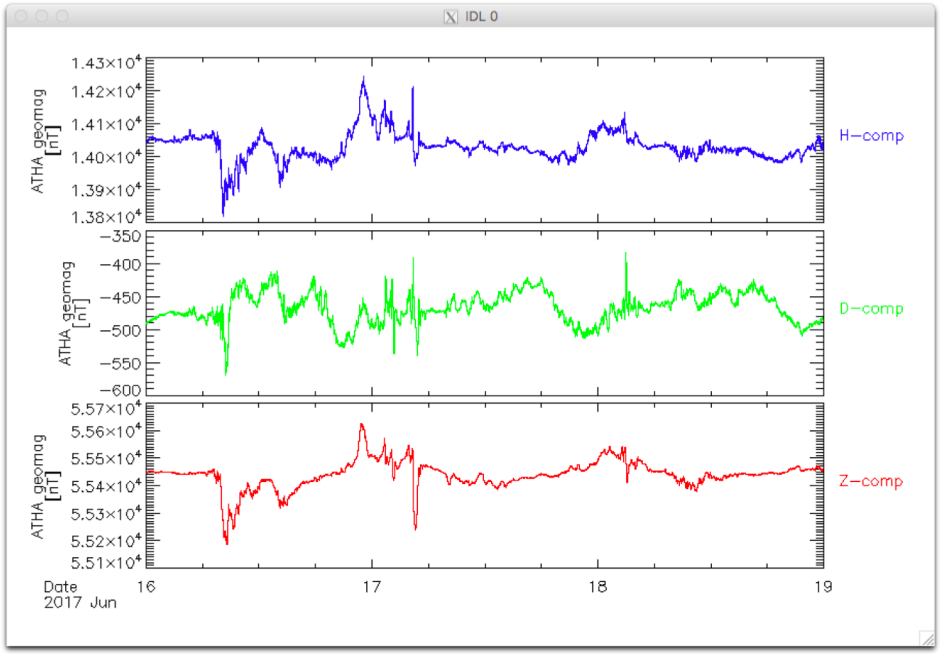


Separate a tplot variable with vector data

```
ERG> split_vec, 'thg_mag_atha'
STORE_DATA(264): Creating tplot variable: 3 thg_mag_atha_x
STORE_DATA(264): Creating tplot variable: 4 thg_mag_atha_y
STORE_DATA(264): Creating tplot variable: 5 thg_mag_atha_z

ERG> tplot, 'thg_mag_atha_?'
```

`split_vec` takes a tplot variable with vector or array data to create new tplot variables containing each component of the vector/array data.



Change the time range of a plot

Select a time period by mouse-clicks on the plot window

```
ERG> tlimit
```

Specify a time period explicitly

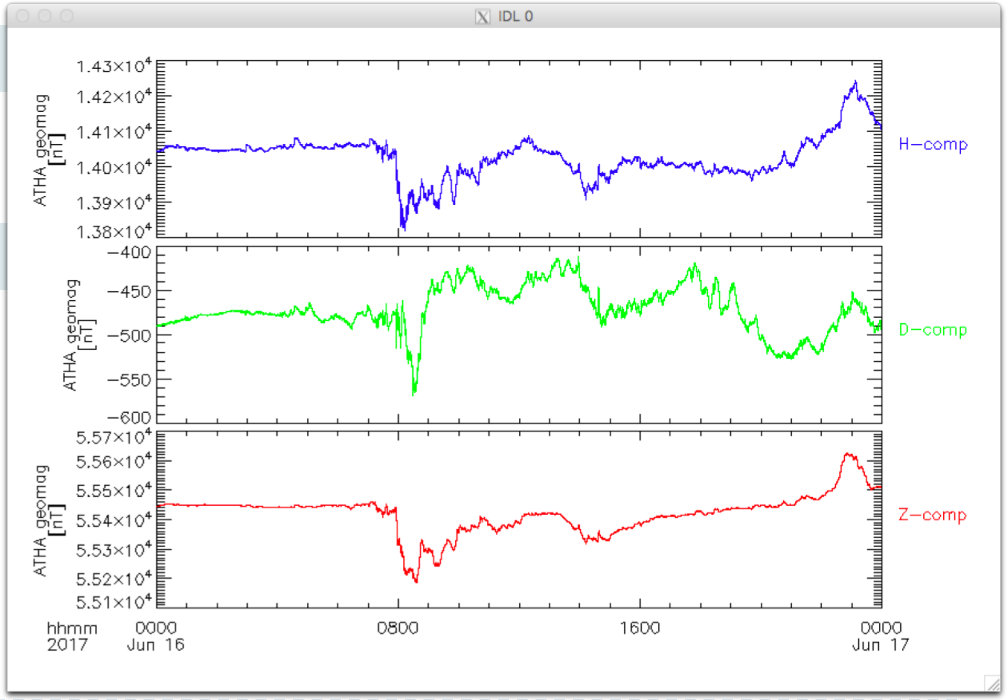
```
ERG> tlimit, '2017-06-16/00:00' , '2017-06-17/00:00'
```

Back to the last plot period

```
ERG> tlimit, /last
```

Restore the original plot period that was set by `timespan`

```
ERG> tlimit, /full
```



Change the vertical scale of a plot

ylim, varname, ymin, ymax, logflag

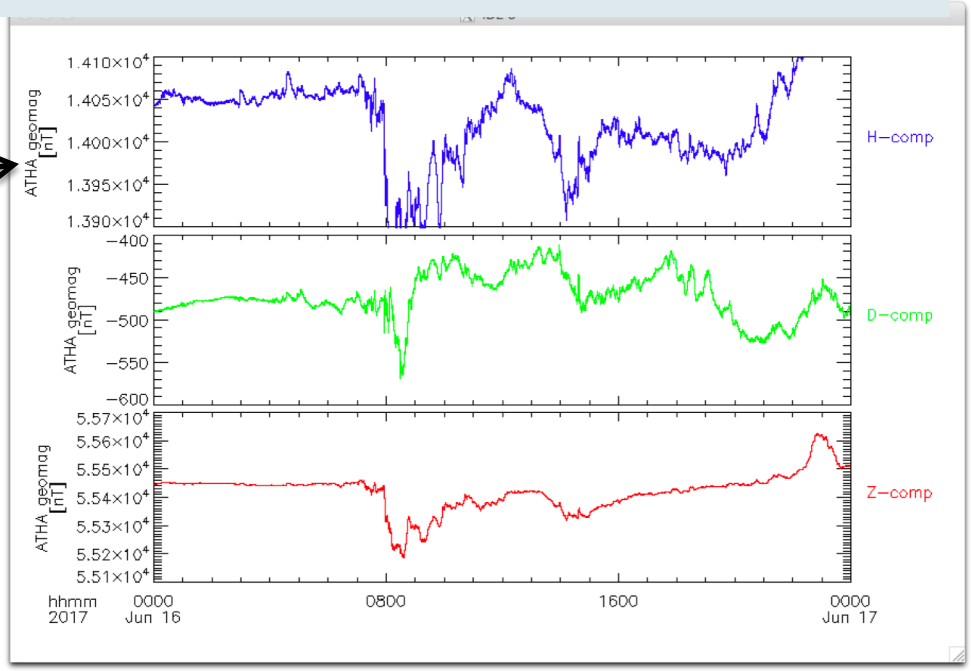
`varname` : variable name(s)

`ymin/ymax` : lower/upper limit along vertical axis
set both to 0 (zero) for plotting with auto-scale

`logflag` : set 0 (zero) for plotting on a linear scale, or 1 for a log scale

```
ERG> ylim, 'thg_mag_atha_x', 13950,14050,  0
ERG> tplot
```

Zoomed in a more limited range in the vertical scale.



Tips:
Putting 0 for both ymin and ymax sets the y range to auto-scale.

```
ERG> ylim, 'thg_mag_atha_x', 0, 0
```

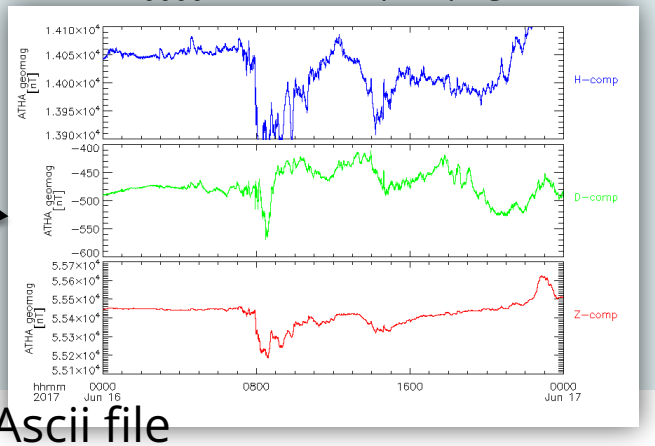
Dump to png, postscript, and Ascii files

To a png file or postscript file

```
ERG> cwd      ;Display the current directory
CWD(25): Directory changed to: /yyyy/xxxx
ERG> tplot, 'thg_mag_atha_?'
ERG> makepng, 'atha_plot'     ;→ atha_plot.png

ERG> popen, 'atha_plot'
ERG> tplot      ;Redo the last plot
ERG> pclose     ; → atha_plot.ps
```

/yyyy/xxxx/atha_plot.png



Dump the data content of a tplot variable to a Ascii file

```
ERG> tplot_ascii, 'thg_mag_atha'
                  ;--> thg_mag_atha.txt, thg_mag_atha_v.txt
```

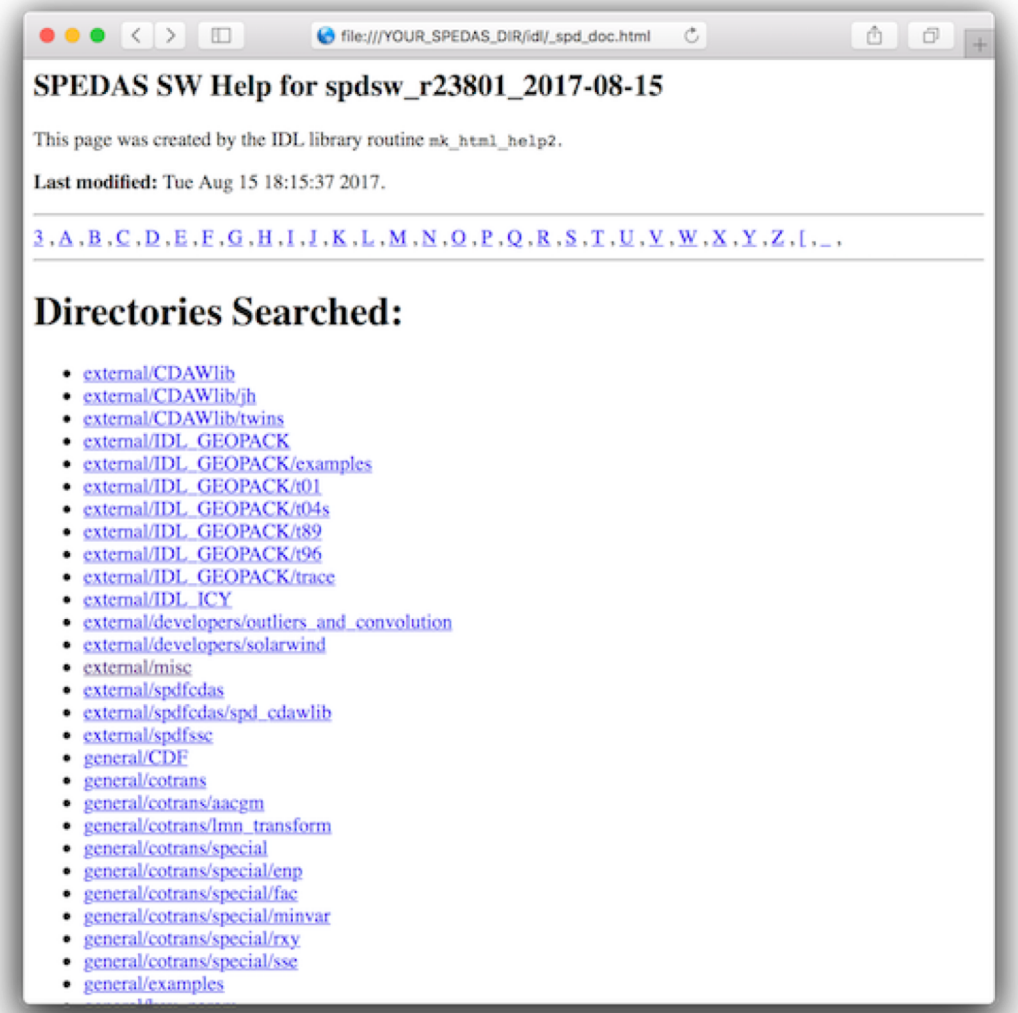
```
oval:Pictures horit$ cat thg_mag_atha.txt
2017-06-16/00:00:00.000        1.4043403e+04        -4.9043201e+02        5.5446332e+04
2017-06-16/00:00:00.500        1.4043136e+04        -4.9049600e+02        5.5446160e+04
2017-06-16/00:00:01.000        1.4043402e+04        -4.9044000e+02        5.5446324e+04
2017-06-16/00:00:01.500        1.4043237e+04        -4.9054401e+02        5.5446152e+04
2017-06-16/00:00:02.000        1.4043406e+04        -4.9041101e+02        5.5446402e+04
2017-06-16/00:00:02.500        1.4043156e+04        -4.9058200e+02        5.5446164e+04
2017-06-16/00:00:03.000        1.4043414e+04        -4.9044299e+02        5.5446363e+04
2017-06-16/00:00:03.500        1.4043179e+04        -4.9059399e+02        5.5446164e+04
2017-06-16/00:00:04.000        1.4043520e+04        -4.9045300e+02        5.5446367e+04
2017-06-16/00:00:04.500        1.4043157e+04        -4.9053601e+02        5.5446152e+04
2017-06-16/00:00:05.000        1.4043453e+04        -4.9048700e+02        5.5446316e+04
2017-06-16/00:00:05.500        1.4043178e+04        -4.9056900e+02        5.5446137e+04
2017-06-16/00:00:06.000        1.4043913e+04        -4.9058099e+02        5.5446355e+04
2017-06-16/00:00:06.500        1.4043140e+04        -4.9050400e+02        5.5446137e+04
2017-06-16/00:00:07.000        1.4043912e+04        -4.9054999e+02        5.5446410e+04
```

SPEDAS manual viewed by web browsers

- Open

  /YOUR_SPEDAS_DIR/idl/_spd_doc.html

  with your web browser to view the automatically generated documents for SPEDAS routines.



Basics of SPEDAS: Various filtering routines for tplot data

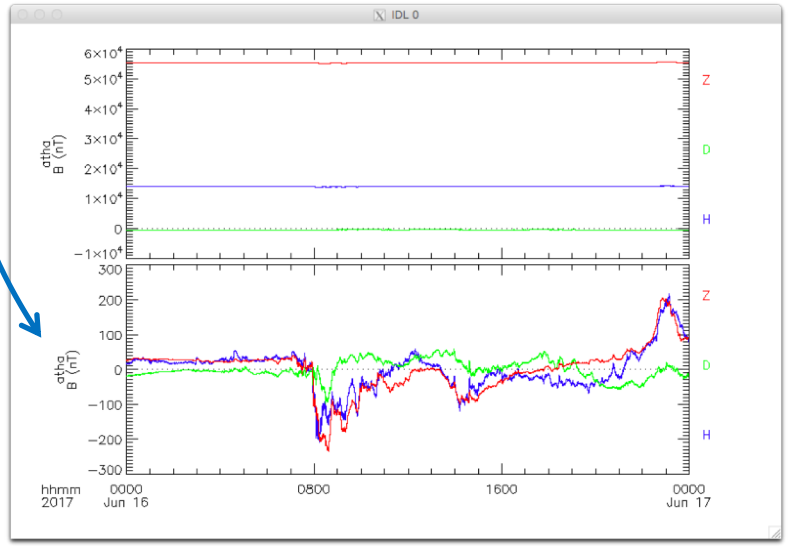
Subtract the mean level – tsub_average –

tsub_average, 'varname'

varname: tplot variable names or index numbers

```
ERG> del_data, '*'
ERG> timespan, '2017-06-16'
ERG> thm_load_gmag, site='atha'
ERG> tsub_average, 'thg_mag_atha'
ERG> tplot, ['thg_mag_atha', 'thg_mag_atha-d' ]
```

Remove all tplot variables and reload the data



boxcar-average data – avg_data –

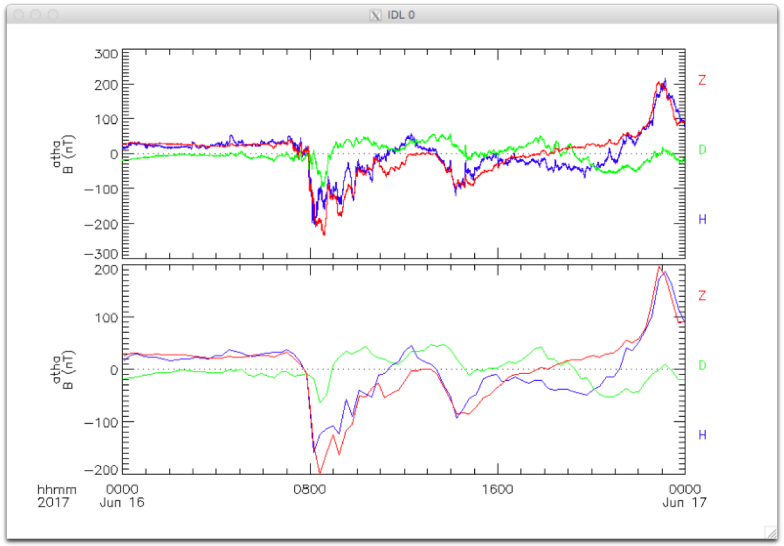
avg_data, 'varname', timebin

varname: tplot variable names or index numbers

timebin : a time window in sec with which the boxcar-averaging is applied to the data

```
ERG> avg_data, 'thg_mag_atha-d', 1000.
STORE_DATA(264): Creating tplot variable: 3 thg_mag_atha-d_avg
ERG> tplot, 'thg_mag_atha-d*'
```

The data boxcar-averaged with a time bin of 1000 second



As a result, the number of data points is reduced to every 1000 s.

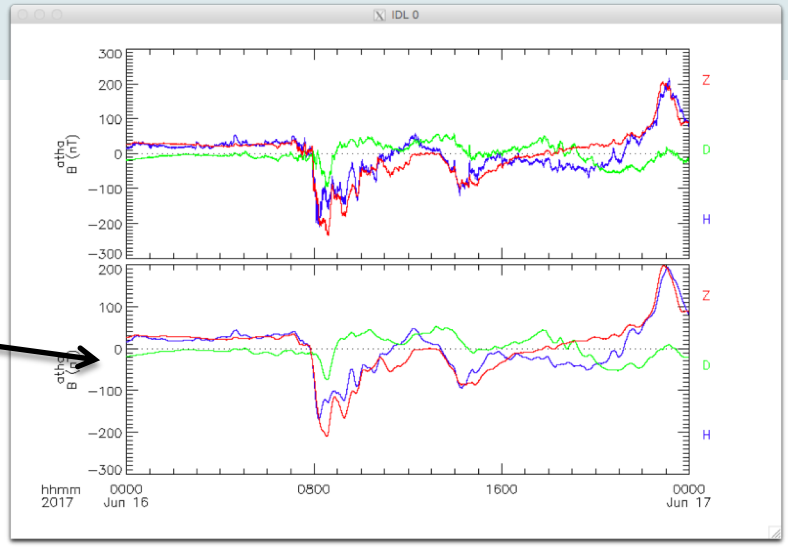
Smoothing data – tsmooth_in_time –

tsmooth_in_time, 'varname', timebin
- varname : tplot variable name(s)
- timebin : time window in second for running average

```
ERG> tsmooth_in_time, 'thg_mag_atha-d', 1000.
ERG> tplot_names
     … … … …
      2 tha_mag_atha-d
     … … … …
      4 tha_mag_atha-d_smoothed
ERG> tplot, [ 2 , 4 ]
```

The data is running-averaged with a time window of 1000 second. We can use this as a rough low-pass filter.

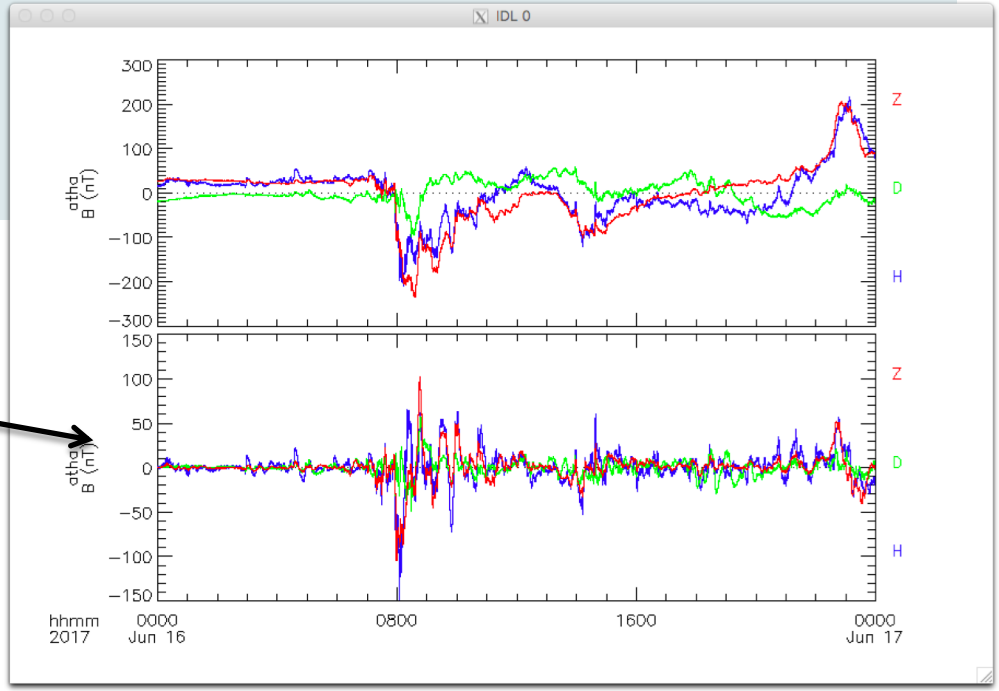
Note that the number of data points is conserved.



High-pass filter in time – thigh_pass_filter –

thigh_pass_filter, 'varname', timebin
`varname` : tplot variable name(s)
`timebin` : time window in second for running average

```
ERG> thigh_pass_filter, 'thg_mag_atha-d', 1000.
ERG> tplot_names
     … … … …
      2 tha_mag_atha-d
     … … … …
      5 tha_mag_atha-d_hpfilt
ERG> tplot, [ 2 , 5 ]
```



Time variations with periods shorter than 1000 sec are shown.

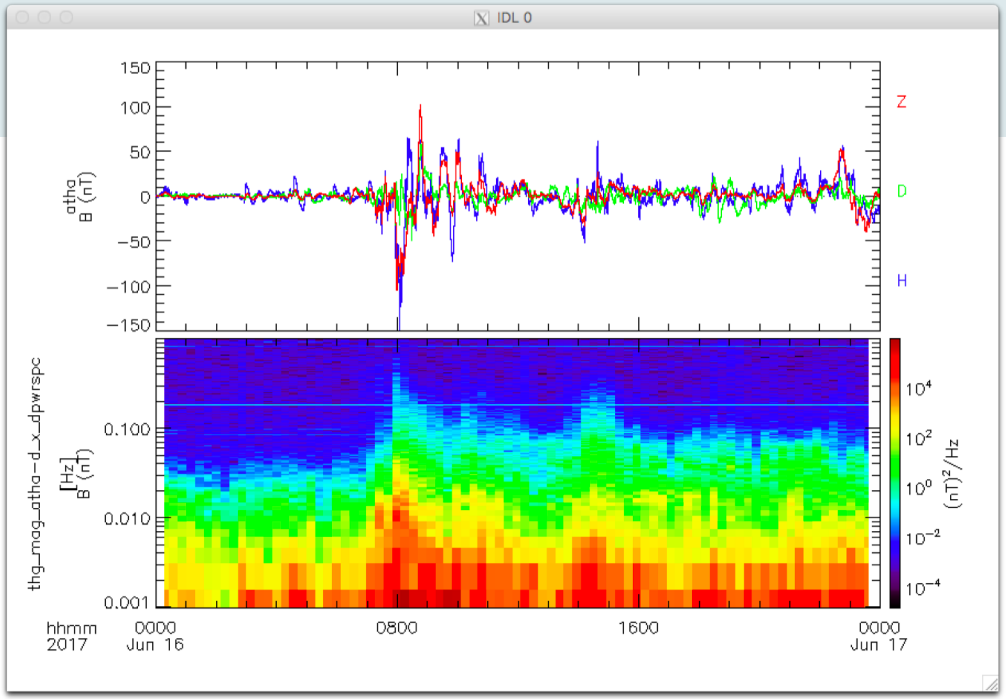
Actually this command just subtracts the low-pass-filtered values derived with `tsmooth_in_time` from the original data, not uses any digital filtering process such as FFT.

Basics of SPEDAS: Frequency analysis of tplot data

Dynamics spectra – tdpwrspc–

tdpwrspc, 'varname'
varname : tplot variable name(s)

```
ERG> tdpwrspc, 'thg_mag_atha-d'
ERG> tplot_names
     … … … …
      5 tha_mag_atha-d_hpfilt
     … … … …
      9 thg_mag_atha-d_x_dpwrspc
ERG> tplot, [ 5, 9 ]
```



FFT with the hanning window is applied to derive dynamic frequency spectra of the data.

Wavelet analysis – wav_data –

wav_data, 'varname'
varname : tplot variable name(s)

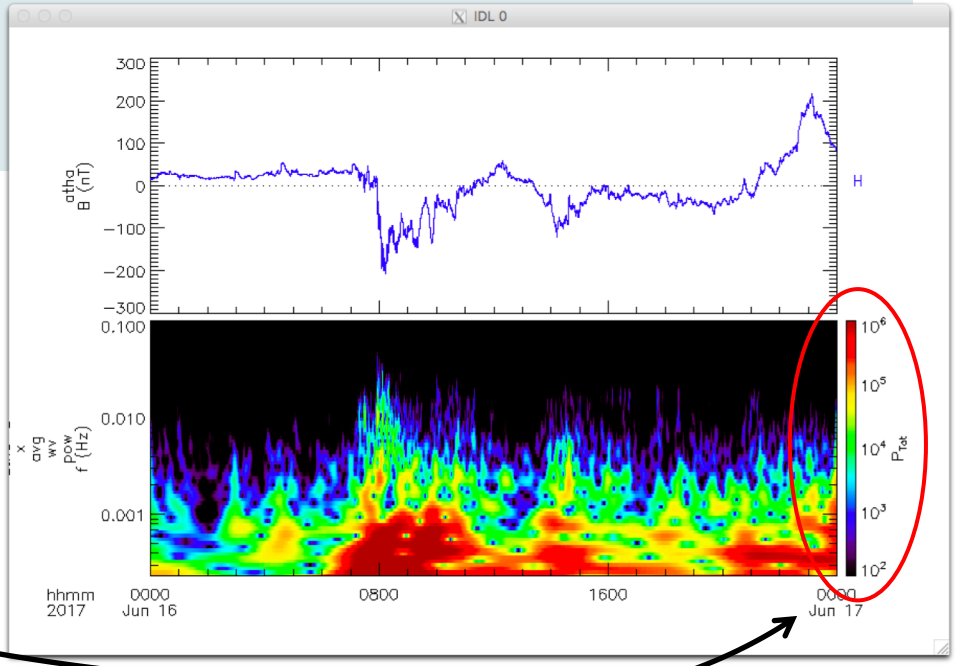
```
ERG> split_vec, 'thg_mag_atha-d'
ERG> avg_data, 'thg_mag_atha-d_x', 5.
ERG> wav_data, 'thg_mag_atha-d_x_avg'
STORE_DATA(264): Creating tplot variable: 13 thg_mag_atha-d_x_avg_wv_pow
ERG> tplot, 'thg_mag_atha-d_x_avg*'

ERG> zlim, 13, 100, 1000000, 1
ERG> tplot, 'thg_mag_atha-d_x_avg*'
```

wav_data accepts data with less than 32768 samples. The number of data points is reduced as done with avg_data in this case.

Wavelet analysis is applied to derive dynamic spectra of the data.

zlim is similar to "ylim" command, but set the lower/upper limit of the color scale for a spectrum-type plot.



Other information sources for SPEDAS

- SPEDAS wiki
  - http://spedas.org/wiki/index.php?title=Main_Page
    - User's guide, Plug-in developer's guide, tips and tricks, The list of available crib sheets, …
- Change log of the source repository for the bleeding edge of SPEDAS
  - http://spedas.org/changelog/
- Crib sheets for TPLOT in Your_SPEDAS_dir/idl/general/examples/
  - **crib_tplot.pro** -- basic tplot intro
  - **crib_tplot_annotation.pro** -- How to control annotations in tplot (labels, text, etc…)
  - **crib_tplot_export_print.pro** -- How to export tplot data and tplot plots
  - **crib_tplot_layout.pro** -- How to control tplot plot layouts
  - **crib_tplot_range.pro** -- How to control the range and scaling of tplot plots
  - **crib_tplot_ticks.pro** -- How to control tplot plot ticks. (location, size, etc…)

Sample cases of ERG data analysis

Data analysis:
3-day plot of XEP data and geomag. indices

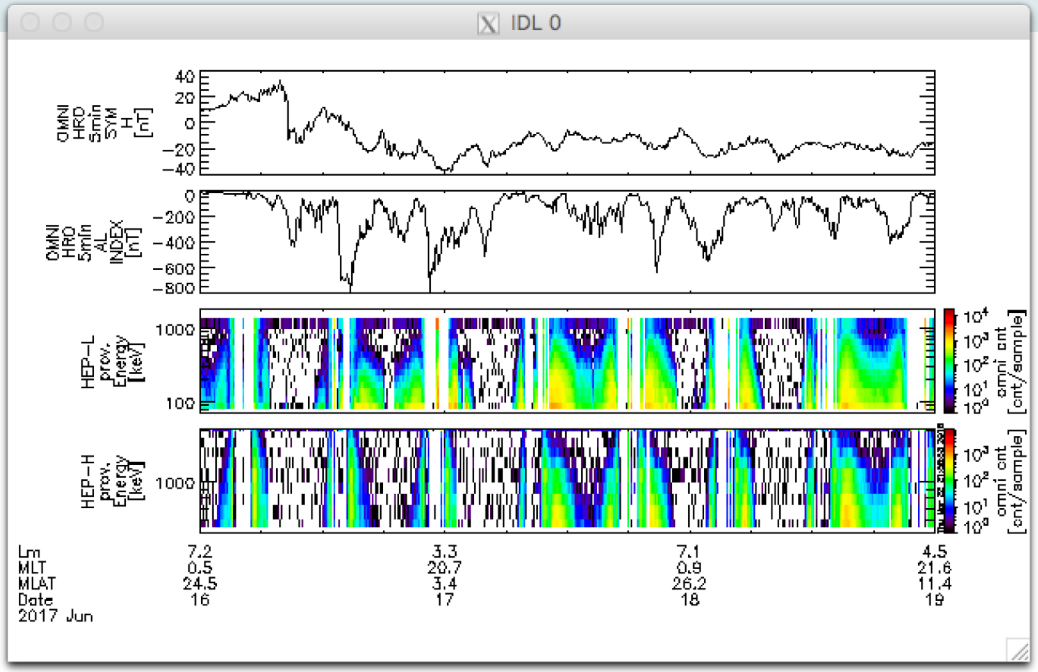
```
ERG> timespan, '2017-06-16', 3, /day
```

Load HEP data

```
ERG> erg_load_hep, datatype='omniflux'
```

Load other index data

```
ERG> omni_hro_load, /res5min
ERG> tplot,[ 'OMNI_HRO_5min_SYM_H' , 'OMNI_HRO_5min_AL_INDEX' , 'erg_hep_l2_FEDO_*' ]
```



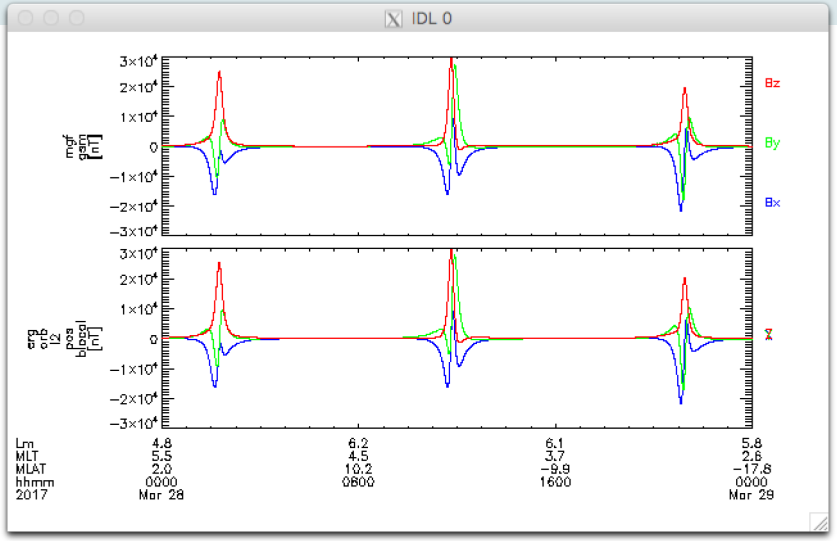
Onboard instrument data:
Coordinate transformation of MGF data

Coordinate transformation (SGA, SGI, DSI, J2000)
From J2000 to the geophysical coordinates -------> using “cotrans”

```
ERG> timespan, '2017-03-28' & erg_load_orb & erg_load_mgf
ERG> erg_cotrans, 'erg_mgf_l2_mag_8sec_dsi', 'mgf_j2000',
     in_coord='dsi', out_coord='j2000'
```

Coordinate transformation (SM<-->J2000)

```
ERG> spd_cotrans, 'mgf_j2000', 'mgf_gsm', in_coord='j2000', out_coord='gsm'
ERG> tplot, ['mgf_gsm', 'erg_orb_l2_pos_blocal']
```



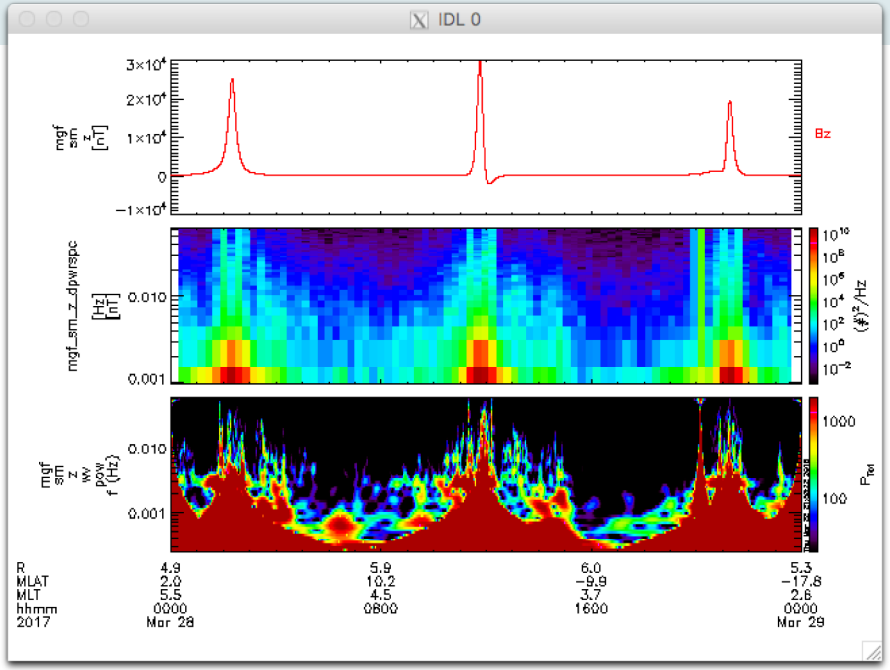
Data analysis:
Frequency analysis of MGF data

Please prepare a tplot variable "mgf_sm" containing the magnetic field vectors in SM coordinates for March 28 (hint: See the prev. page!)

```
ERG> timespan, '2017-03-28', 1, /day
ERG> tdpwrspc, 'mgf_sm'
ERG> wav_data, 'mgf_sm_z'
ERG> tplot,'mgf_sm_z*'
```

Dynamic spectra by FFT with the Hanning window

Dynamic spectra by a wavelet analysis

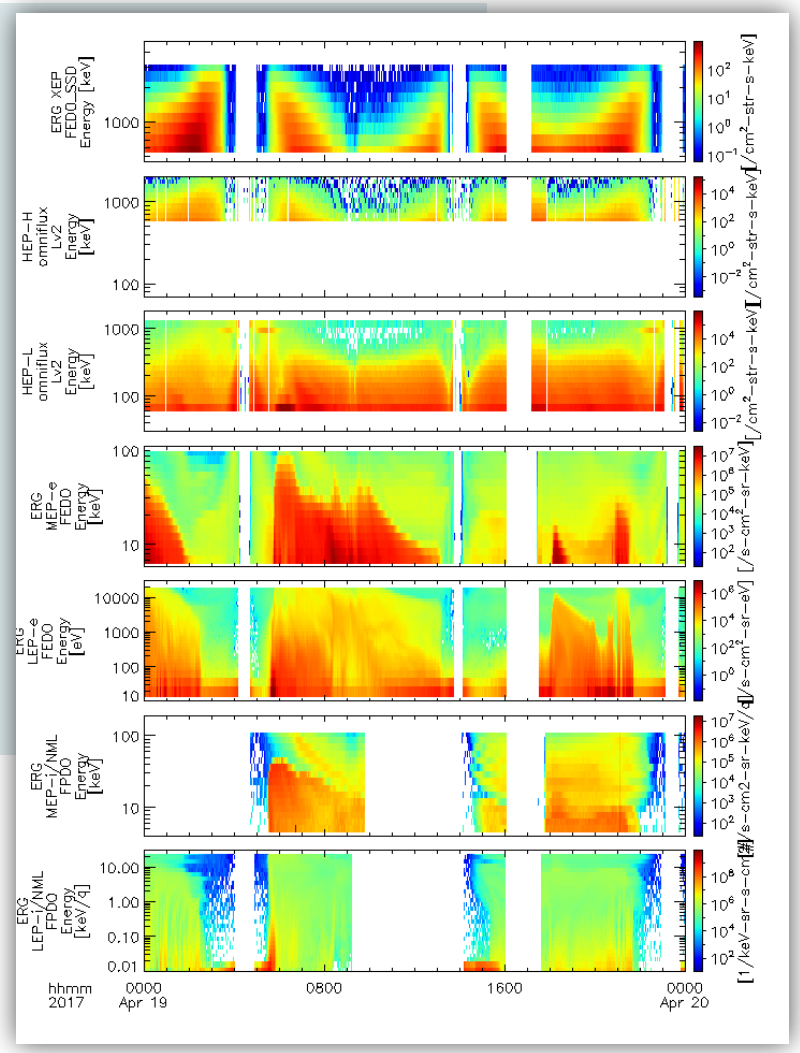


Use `tlimit` to zoom in a period between perigee points!

Onboard instrument data: Level-2 particle omni-flux data

```
;; Set the time span
timespan, '2017-04-19'
;; Load data
erg_load_xep, datatype='omniflux'
erg_load_hep, datatype='omniflux'
erg_load_mepe, datatype='omniflux'
erg_load_lepe, datatype='omniflux'
erg_load_mepi_nml, datatype='omniflux'
erg_load_lepi_nml, datatype='omniflux'
tplot_names

tplot,[ 'erg_xep_l2_FEDO_SSD', $
  'erg_hep_l2_FEDO_H','erg_hep_l2_FEDO_L', $
  'erg_mepe_l2_omniflux_FEDO', $
  'erg_lepe_l2_omniflux_FEDO', $
  'erg_mepi_l2_omniflux_FPDO', $
  'erg_lepi_l2_omniflux_FPDO'    ]
```



Change the time range of a plot: *tlimit*

Select a time period by mouse-clicks on the plot window

```
ERG> tlimit
```

Specify a time period explicitly

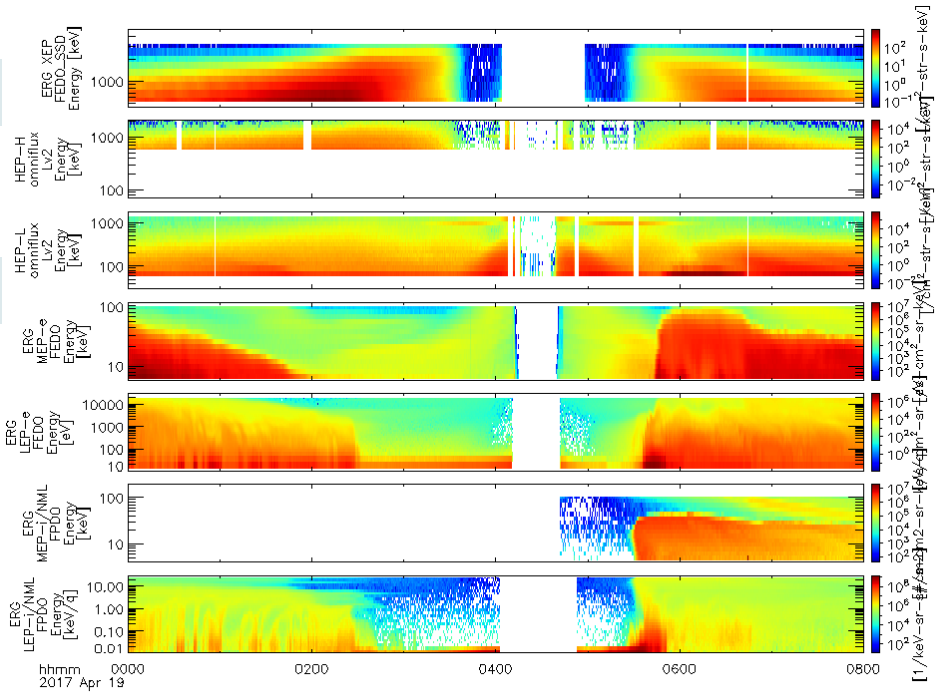
```
ERG> tlimit, '2017-04-19/00:00' , '2017-04-19/08:00'
```

Back to the last plot period

```
ERG> tlimit, /last
```

Restore the original plot period that was set by `timespan`

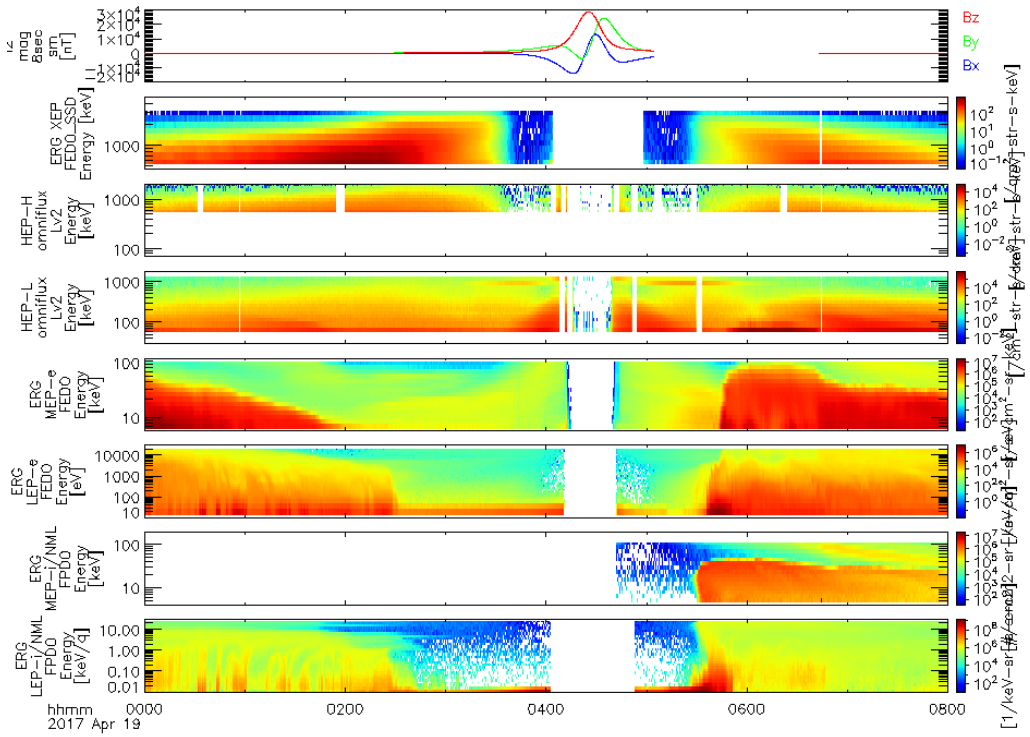
```
ERG> tlimit, /full
```



Onboard instrument data:
Add MGF data with /add keyword

Load MGF data and add it to the pre-existing particle plot

```
ERG> erg_load_mgf, uname=uname, pass=pass
ERG> tplot, 'erg_mgf_l2_mag_8sec_sm' , /add
```



Orbit data:
Set time range and load ERG orbit data

Setup the time range ('YYYY-MM-DD/hh:mm:ss')

```
ERG> timespan,'2017-03-28/00:00:00',3,/day
```

Load orbit data

```
ERG> erg_load_orb
ERG> tplot_names
```

```
ERG> tplot_names
   1 erg_orb_l2_pos_gse
   2 erg_orb_l2_pos_gsm
   3 erg_orb_l2_pos_sm
   4 erg_orb_l2_pos_rmlatmlt
   5 erg_orb_l2_pos_eq
   6 erg_orb_l2_pos_iono_north
   7 erg_orb_l2_pos_iono_south
   8 erg_orb_l2_pos_blocal
   9 erg_orb_l2_pos_blocal_mag
  10 erg_orb_l2_pos_beq
  11 erg_orb_l2_pos_beq_mag
  12 erg_orb_l2_pos_Lm
  13 erg_orb_l2_vel_gse
  14 erg_orb_l2_vel_gsm
  15 erg_orb_l2_vel_sm
  16 erg_orb_l2_spn_num
  17 erg_orb_l2_man_prep_flag
  18 erg_orb_l2_man_on_flag
  19 erg_orb_l2_eclipse_flag
ERG>
```

("erg_orb_l2" means ERG Level-2 orbit data)

(*) Using the IGRF model

pos_gse/gsm/sm	s/c position [Re] in GSE, GSM, SM coordinates
pos_rmlatmlt	Radial distance [Re], magnetic latitude [deg], local time [hr] of s/c position
pos_eq	s/c position mapped to the magnetic equator
pos_iono_north/south	Geographic latitude and longitude [deg] of s/c footprints at 100 km altitude in the northern/southern hemisphere
pos_blocal / blocal_mag	model B-field vector (blocal) and B-field strength (blocal_mag) [nT] at s/c position
pos_beq / beq_mag	model B-field vector (beq) and B-field strength (beq_mag) [nT] at s/c position mapped to the magnetic equator
pos_Lm	McIlwain's L-parameter of s/c position for pitch angles of 90, 60, and 30 deg
vel_gse/gsm/sm	s/c orbital velocity [km/s] in GSE, GSM, and SM
man_prep/man_on/eclipse_flag	flag for maneuver preparation (man_prep), maneuver on/off (man _on), and solar eclipse (eclipse)

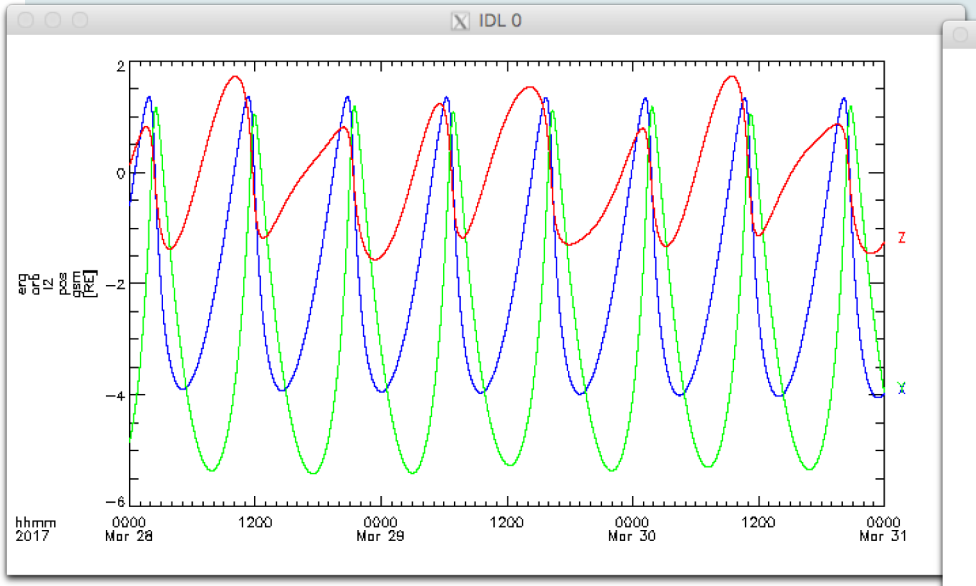
Orbit data: Definitive orbit as a time series plot: *tplot, tplotxy*

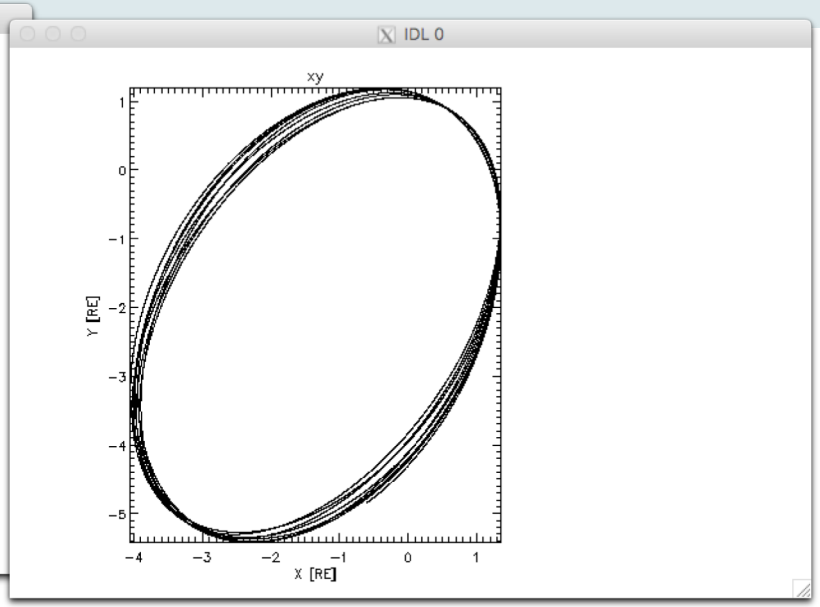
Plot orbit time series data

```
ERG> tplot, 'erg_orb_l2_pos_gsm'
```

Plot orbit data in the X-Y plane.

```
ERG> tplotxy, 'erg_orb_l2_pos_gsm'
```



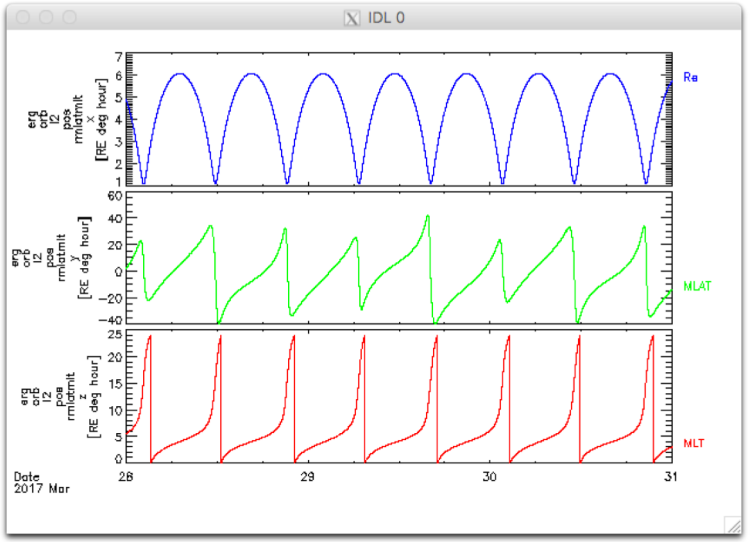


Separate a tplot variable with vector data: *split_vec*

```
ERG> split_vec, 'erg_orb_l2_pos_rmlatmlt'
STORE_DATA(264): Creating tplot variable: ...
STORE_DATA(264): Creating tplot variable: ...
STORE_DATA(264): Creating tplot variable: ...

ERG> tplot, 'erg_orb_l2_pos_rmlatmlt_?'
```

`split_vec` takes a tplot variable with vector or array data to create new tplot variables containing each component of the vector/array data.



Orbit data:
Insert orbit values below a time-series plot

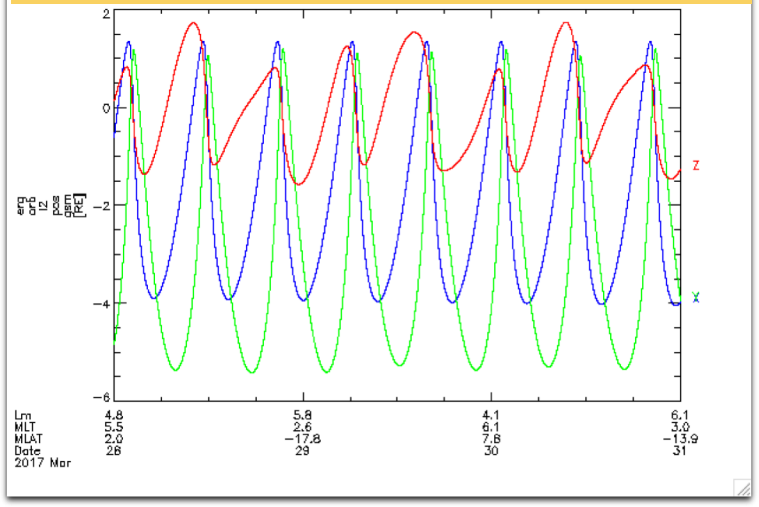
```
ERG> set_erg_var_label
ERG> tplot      ;just type "tplot" to replot the previous panels

ERG> split_vec, 'erg_orb_l2_pos_rmlatmlt'
ERG> options, 'erg_orb_l2_pos_rmlatmlt_x', ytitle='R'
ERG> options, 'erg_orb_l2_pos_rmlatmlt_y', ytitle='MLAT'
ERG> options, 'erg_orb_l2_pos_rmlatmlt_z', ytitle='MLT'
ERG> tplot_options, var_label=['erg_orb_l2_pos_rmlatmlt_z',
'erg_orb_l2_pos_rmlatmlt_y', 'erg_orb_l2_pos_rmlatmlt_x']
ERG> tplot
```

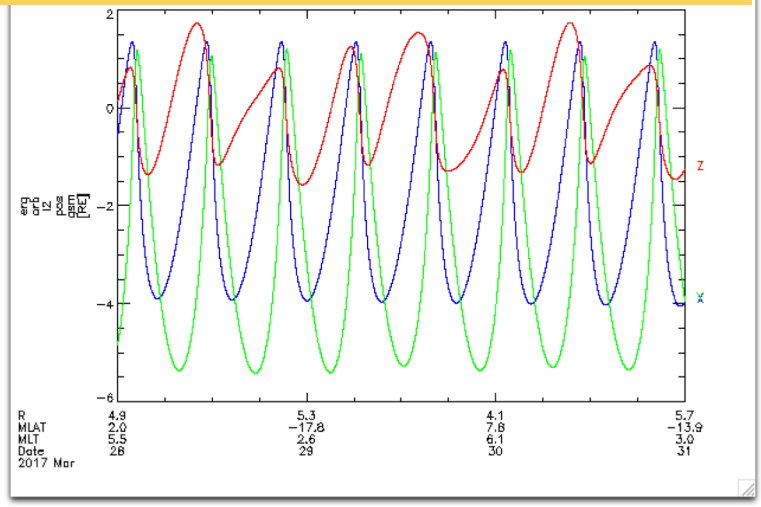
Using subroutine to add labels (Lm,MLT,MLAT) with time

An example to add new labels (R,MLAT,MLT)

(Lm, MLAT, MLT)



(R, MLT, MLAT)



Appendix 1:
Access the data structure in a tplot variable and create a new tplot variable (get_data, store_data)

Access the data structure in a tplot variable – get_data –

```
ERG> get_data, 'thg_mag_atha', data=data, dlimits=dlimits, lim=lim
```

`data`: the data structure is stored
`dlimits`: most of metadata are stored
`lim`: some plot properties are stored

```
ERG> help, data.x , data.y
```

```
ERG> help, data
** Structure <272b308>, 3 tags, length=3456016, d
   X               DOUBLE    Array[172800]
   Y               FLOAT     Array[172800, 3]
   V               LONG      Array[3]
```

"get_data" extracts the data structure of a tplot variable and saves in a structure "data" of IDL session in the above case, so that users can access them by referring to as "data.x" or "data.y", for example.

```
ERG> help, dlimits
** Structure <382fa08>, 10 tags, length=904, data length=890, refs=8:
   CDF             STRUCT    -> <Anonymous> Array[1]
   SPEC            BYTE        0
   LOG             BYTE        0
   COLORS          INT       Array[3]
   CONSTANT        FLOAT           0.00000
   LABELS          STRING    Array[3]
   LABFLAG         INT             1
   YSUBTITLE       STRING    'B (nT)'
   YTITLE          STRING    'atha'
   DATA_ATT        STRUCT    -> <Anonymous> Array[1]
```

The information on the original CDF data file (CDF) and metadata, and various plot properties are extracted into a structure "dlimits".

Create a new tplot variable – store_data –

```
ERG> store_data, 'varname' , data = { x:timearr, y:datarr }
```

varname: name of a newly created tplot variable

timearr: 1-D array containing time values in SPEDAS time of time-series data

datarr: 1-D or 2-D array containing the data values of time-series data. The size of 1st dimension should be identical to that of timearr.

- SPEDAS time is the UNIX time in double-precision floating-point values. UNIX time is the elapsed second since 00:00 UTC on January 1, 1970.
- Usually we use `time_double()` function to calculate a SPEDAS time value from a time string such as '2017-06-16/12:30:00'.
- SPEDAS time values can easily be converted to time strings with `time_string()` function.

```
ERG> timestr='2017-06-16/12:30:00'
ERG> spedastime = time_double( timestr )
ERG> print, spedastime
   1.4976162e+09
ERG> print, time_string( spedastime )
2017-06-16/12:30:00
```

Please refer to the SPEDAS wiki at http://spedas.org/wiki/index.php?title=Time_handling for more details of the time handling in SPEDAS.

Appendix 2: Various options for SPEDAS use

Options for SPEDAS use

- Using SPEDAS source code
  - All SPEDAS IDL source code including both the command-line tools and graphical user interface (GUI) tools.
  - To use it, you need to have IDL installed with a proper license.
  - The latest plug-ins can be used by installing by yourself.
  - Users can develop, edit, and run their own scripts.
- Using SPEDAS save file
  - binary files working with the IDL virtual machine (IDL-VM) environment
  - IDL-VM can be downloaded for free from Exelis/Harris Geospatial.
  - No command-line tools: only GUI is available.
  - only plug-ins implemented to the save file are available for use. Some latest plug-ins may not be included yet.
- Using SPEDAS executable files
  - A package bundling IDL-VM and the SPEDAS save file, basically equivalent to the above.

- We use the source code version in this training session.