Introduction to Geophysics Analysis Package (GAP)

prepared by Eugen Sorbalo

Jacobs University Bremen

April 4, 2014

The following exercises demonstrates how to:
- download Cluster's magnetic field data and read CDF files
- convert between reference frames (e.g. GSE to GSM)
- perform a Minimum-Variance Analysis (MVA)
- estimate currents using Curlometer

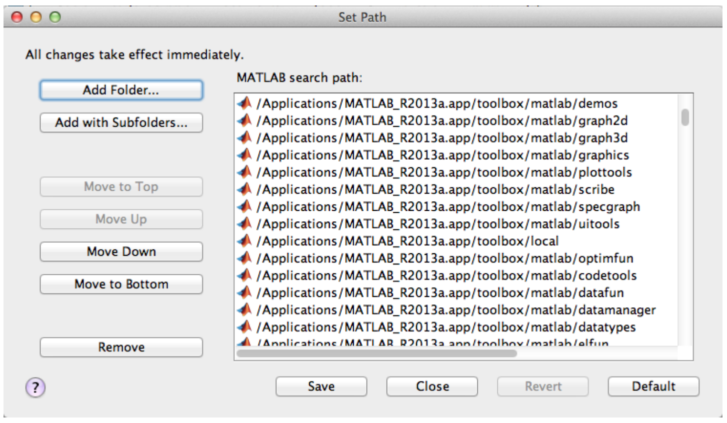
Geophysics Analysis Package [1] (GAP) is a package formed of MatLab's functions, whose aim is to facilitate the processing of data from magnetospheric and ionospheric satellite missions. Of special interest are the multi-satellites missions: *Cluster II*, which comprises 4 satellites flying at separations of ∼100s km to 10000 km, *Double Star* (two Cluster-type satellites), *Themis*, whose five satellites orbit in the elliptic orbits in the equatorial plane, and *Swarm*, which has three satellites orbiting in circular polar orbits at altitudes of 300km to 500 km above the surface of the Earth. The data from single missions, such as *CHAMP*, *FAST*, *ACE*, *Geotail*, etc. present as well an important interest.

The demonstrations included here will show the first steps in using GAP: from downloading and reading data files, to obtaining current estimations and performing a minimum variance analysis.

[1]It can be downloaded from: `http://sourceforge.net/projects/geophysicsanalysispackage/`

1 Introduction and conventions

The package can be downloaded from
`http://sourceforge.net/projects/geophysicsanalysispackage`.
There you can also find the information on how to install and use the package. The installation normally comprises of unarchiving the downloaded ZIP file and setting the path in MatLab to this folder and its subfolders. In order to set the paths using the graphic interface of MatLab, type `pathtool` in the *Command Window*. The window as in figure 1 will appear. Click on *Add with Subfolders...* button, find the `gap_library` folder and click on *Open*. At the end, *Save* the new configuration.



All changes take effect immediately.

MATLAB search path:

```
/Applications/MATLAB_R2013a.app/toolbox/matlab/demos
/Applications/MATLAB_R2013a.app/toolbox/matlab/graph2d
/Applications/MATLAB_R2013a.app/toolbox/matlab/graph3d
/Applications/MATLAB_R2013a.app/toolbox/matlab/graphics
/Applications/MATLAB_R2013a.app/toolbox/matlab/plottools
/Applications/MATLAB_R2013a.app/toolbox/matlab/scribe
/Applications/MATLAB_R2013a.app/toolbox/matlab/specgraph
/Applications/MATLAB_R2013a.app/toolbox/matlab/uitools
/Applications/MATLAB_R2013a.app/toolbox/local
/Applications/MATLAB_R2013a.app/toolbox/matlab/optimfun
/Applications/MATLAB_R2013a.app/toolbox/matlab/codetools
/Applications/MATLAB_R2013a.app/toolbox/matlab/datafun
/Applications/MATLAB_R2013a.app/toolbox/matlab/datamanager
/Applications/MATLAB_R2013a.app/toolbox/matlab/datatypes
/Applications/MATLAB_R2013a.app/toolbox/matlab/elfun
```

Figure 1: *Set Path* window in MatLab with graphic interface. It is used to tell MatLab where to look for GAP's functions.

In the command line MatLab, without graphic interface, the path can be added using the function `addpath()`. For example, in order to add the GAP folder and its subfolders to the MatLab's search path on Linux, use the following line within MatLab:

```
addpath(genpath('/path_to_package/gap_library'))
```

where `/path_to_package` should be the actual path to the package on your computer. If the folder `gap_library` is located in your home directory, then the command would be:

```
addpath(genpath('¬/gap_library'))
```

In order to not give the above command every time when MatLab starts, write the line into the file `startup.m` (create if it does not exist) and save the file in the current directory.

Once the path is set, all functions of the package can be used as stand alone. They might have interdependencies, but there is no need for any initializations and cleaning in your code. Just call any function with input parameters and read output parameters. The input parameters from all functions follow the same format:

scalars are arrays Nx1 (N rows and 1 column), where N is the number of measurements.

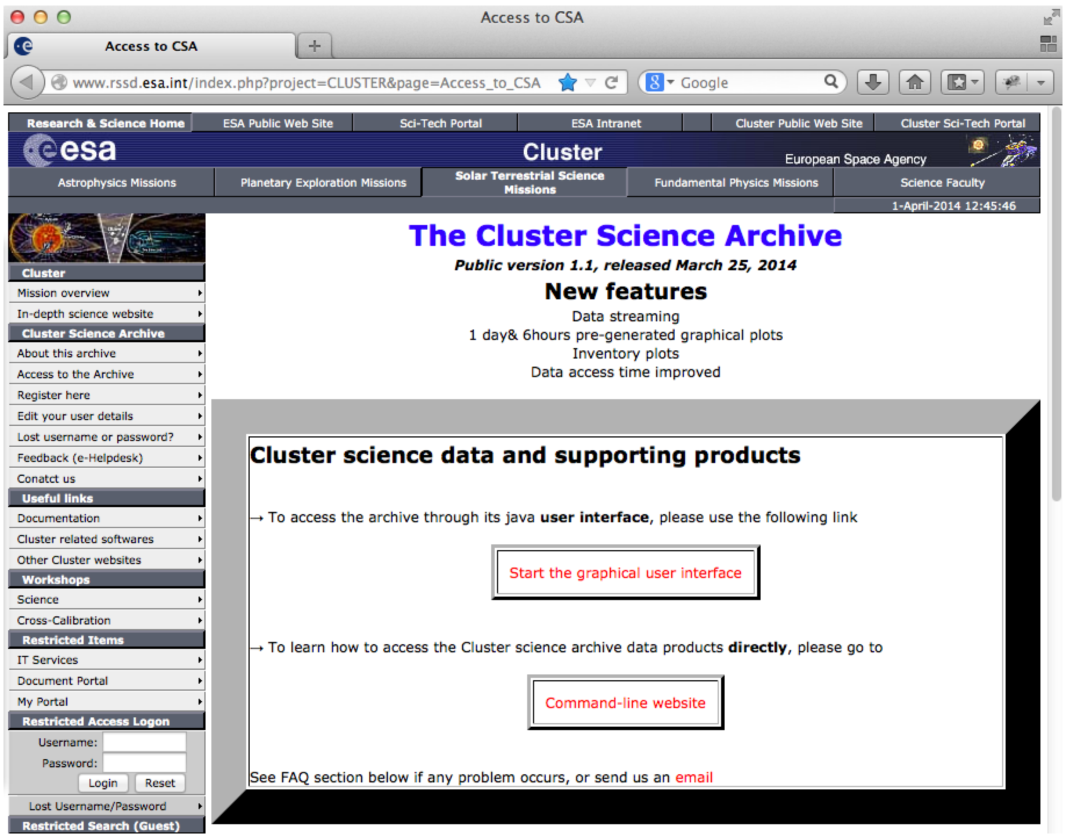
vectors are arrays Nx3 with each row representing a vector. This format was chosen so that it is easier to visualize variables in the *Variable* viewer in the graphical interface of MatLab.

time tags are arrays of scalar Nx1 corresponding to the N measurements. Each time tag is given in the MatLab's `datenum` format, i.e. indicates number of days since 01-January-2000. Given time tags `t1` and `t2`, then the number of seconds between them can be computed as `diff_in_sec = (t2-t1)*24*60*60`.

2 Downloading and importing data

In the first example it is shown how to download the magnetic field data, produced by the Cluster II satellites, read the data and plot the magnetic field and positions. Here we used an event from 2001-June-11 20:05 to 20:23. The event was studied by Dunlop and Balogh [2005].

The data can be downloaded from the *Cluster Science Archive*: `http://www.rssd.esa.int/index.php?project=CLUSTER&page=Access_to_CSA`. The page looks like in the figure 2.



The Cluster Science Archive

Public version 1.1, released March 25, 2014

New features

Data streaming
1 day& 6hours pre-generated graphical plots
Inventory plots
Data access time improved

Cluster science data and supporting products

→ To access the archive through its java **user interface**, please use the following link

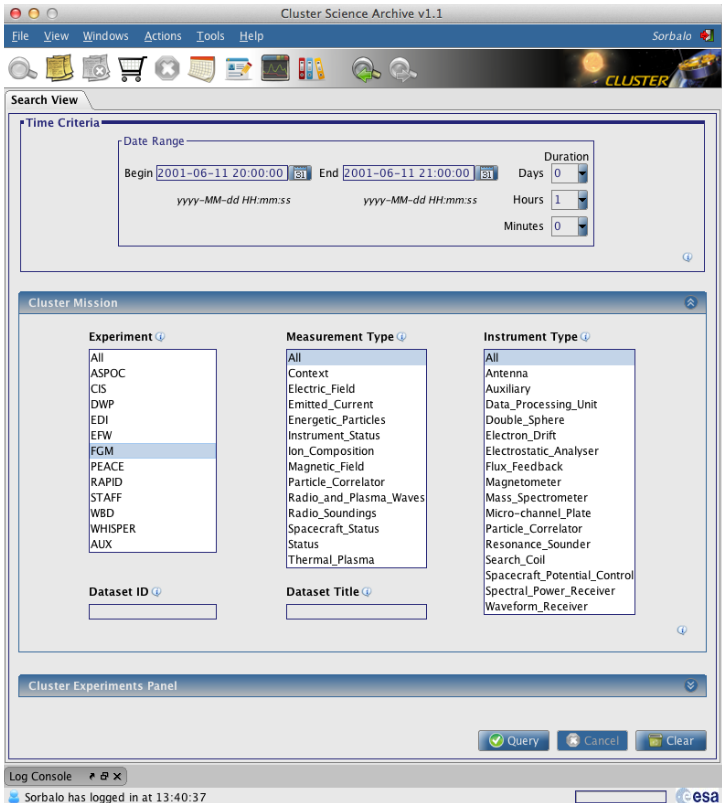
→ To learn how to access the Cluster science archive data products **directly**, please go to

See FAQ section below if any problem occurs, or send us an email

Figure 2: Cluster Science Archive website

In order to be able to lunch the data selection application, the website requires that the user has Java Runtime Engine installed and active on the computer. We click on *Start the graphical user interface* and we choose to open the *csa.jnlp* file with *Java Web Start*. The application should look like in the figure 3.

Registration is required in order to be able to download data. It can be done by going to the menu *Actions* and then to *Register As New User*. The user can sign in by going to the menu*Actions* $->$ *Login*. Then, we selected the *Date Range* from 2001-06-11 20:00:00 and set *Duration* to one hour. The starting time can be specified by clicking on the position of hours or minutes and then using the up- and down- keys on the keyboard. We selected the *FGM* experiment and finally clicked on *Query*. On the resulting page we click on the green arrow, in order to see subitems, and



Begin 2001-06-11 20:00:00 End 2001-06-11 21:00:00
yyyy-MM-dd HH:mm:ss yyyy-MM-dd HH:mm:ss
Duration: Days 0, Hours 1, Minutes 0
Experiment: FGM
Sorbalo has logged in at 13:40:37

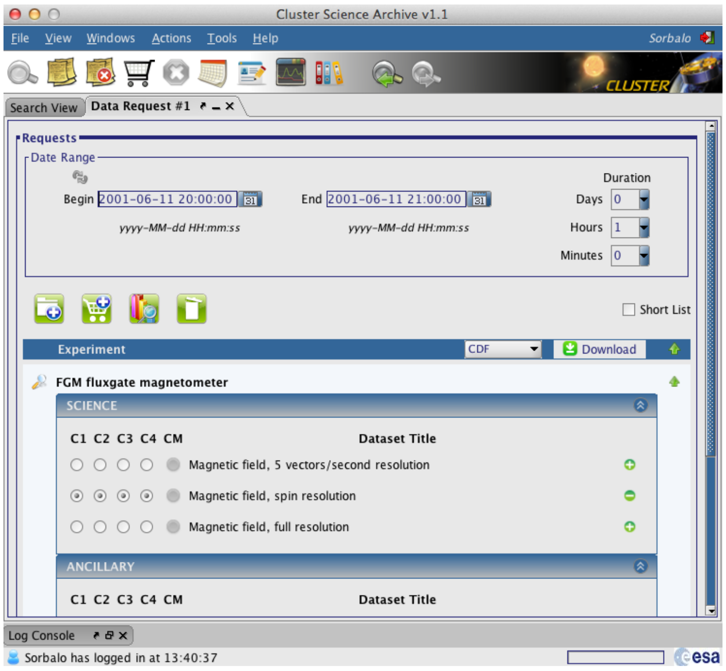
Figure 3: Cluster Science Archive

we select the CDF file format (a binary data format developed by NASA and is supported by most data analysis software; the CEF format was developed by ESA and is in the open text format). In this exercise we will plot data only from one satellite, but we will use data from all four satellites in the following exercises, so we select *Magnetic field, spin resolution* for all four satellites. The result is shown in the figure 4.

Finally we click on *Download* and select *All*, then click on *OK* and save the archive somewhere. Once saved, we unpack the archive and we find the folder `CSA_Download_20140401_1617` (actual name will depend on the date and time of the data download), which contains four subfolder ending in `_FGM_SPIN` and four subfolders ending in `_FGM_CAVF`. The former subfolder contain CDF files with magnetic field and positions with the spin resolution (one measurement per 4 seconds). Here we will consider the file `C1_CP_FGM_SPIN__20010611_200000_20010611_210000_V070906.cdf` located in the folder `C1_CP_FGM_SPIN`, in which we will write our first MatLab script:

```
%% Read data from CDF file:
filename = 'C1_CP_FGM_SPIN__20010611_200000_20010611_210000_V070906';
[times, B_gse, pos_gse] = gap_convert_Cluster_fgm2mat(filename);
    % times   - array Nx1 with time tags in the MatLab's datenum format
    %                (number of days since 01-January-2000).
    % B_gse   - array Nx3 with each line having a magnetic field vector.
    % pos_gse - array Nx3 with each line having a position vector.

%% Plot data
subplot(2,1,1);                % The upper plot
plot(times, B_gse);
legend('x', 'y', 'z');         % Describes the lines on the plot
```



Begin 2001-06-11 20:00:00 End 2001-06-11 21:00:00

yyyy-MM-dd HH:mm:ss yyyy-MM-dd HH:mm:ss

Duration: Days 0, Hours 1, Minutes 0

Experiment: CDF

FGM fluxgate magnetometer

C1	C2	C3	C4	CM	Dataset Title
○	○	○	○	●	Magnetic field, 5 vectors/second resolution
◉	◉	◉	◉	●	Magnetic field, spin resolution
○	○	○	○	●	Magnetic field, full resolution

C1	C2	C3	C4	CM	Dataset Title

Sorbalo has logged in at 13:40:37

Figure 4: Cluster Science Archive: request of the magnetic field data from FGM instrument.

```
ylabel('B_{C1,GSE}');        % Describes the vertical axis
datetick('x');               % Auto—formats the x axis to display hours
                             % and minutes

subplot(2,1,2);              % The lower plot
plot(times, pos_gse);
legend('x', 'y', 'z');
ylabel('Pos_{C1,GSE}');
xlabel('time');              % Describes the horizontal axis.
datetick('x');
```

The result of the script is shown in the figure 5.

The important function here is

```
[times, B_gse_vec, pos_gse_vec] = gap_convert_Cluster_fgm2mat(cdf_filename, ...
    mat_filename)
```

which takes in the path to the CDF file as input parameter, and outputs the time tags, magnetic field vectors and position vectors. If the optional parameter `mat_filename` is provided, then the data will be also stored in the MatLab's native storage file.

3 Reading data from any CDF file

In the previous example we read the magnetic field data from Cluster satellites using one of GAP functions. Other instruments and other satellites provide data in slightly or totally different formats. Lately, more and more data is provided as CDF files, which can be read natively by MatLab. At this stage we should look at the content of the function `gap_convert_Cluster_fgm2mat()`, which has one line for reading data and three lines for converting data from cells into arrays of floats (real numbers with 16 bit precision):

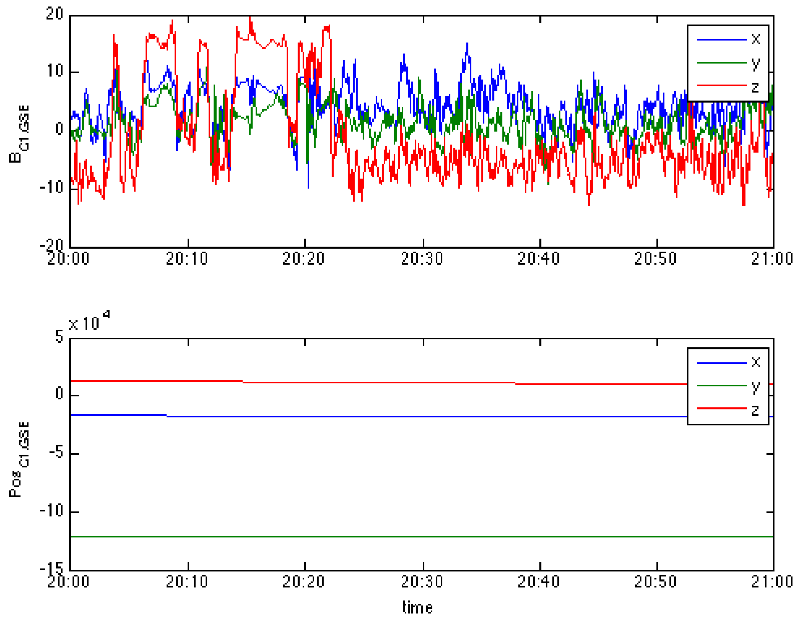


Figure 5: The result of the first exercise: magnetic field in GSE frame from FGM instrument on the satellite C1 from Cluster mission.

```
[data, info] = cdfread(cdf_filename, 'ConvertEpoch', true);
times = cell2mat(data(:,1));
B_gse_vec = cell2mat(data(:,3)')';
pos_gse_vec = cell2mat(data(:,5)')';
```

The function `[data, info] = cdfread(path_to_cdf_file)` can read any CDF file. The data are contained in the first variable, `data`, which is an array of cells. When the content of the CDF file is not known, then we should have a look at the second variable, `info`, which provides us with all metadata. More specifically, `info.Variables` provide us with information about contained variables and if each measurement is a scalar, vector or matrix. Another important member is `info.VariableAttributes`, which gives information about units, frames, axis labeling, etc of the corresponding variables.

From the `info.Variables` we understand that Cluster's CDF files from the flux magnetometer (FGM) instrument has time tags of the measurements in the first column, vector of magnetic field on the third column, magnitude of the magnetic field on fourth column and the position vectors on the fifth column of the `data` variable. This variable is an array of cells. A cell can contain any kind of variables and an array of cells is a useful method to combine together data of different kinds (doubles, integers, arrays, strings, etc). In our case, any cell of the third column will represent a matrix 3x1, which has a measurement of the magnetic field. In order to convert those cells into an array of singles, we use the function `cell2mat()`.

Note that CDF files do not have the time defined as numbers. Instead they are defined in the so called *Epoch* format. There is a function to convert *Epoch* to datenum, however it can be done automatically by specifying the pair `'ConvertEpoch', true` in the function `cdfread()`.

4 Conversion between reference frames

The conversion between reference frames is done using functions:

```
[vec_end] = gap_convert_vector(vec_orig, longlat, times, ...
    orig_system, end_system);
[xyz_end, rotM] = gap_convert_position(xyz_orig, times, ...
    orig_system, end_system);
```

The input variables `orig_system` and `end_system` can be any of `'GSE'`, `'GSM'`, `'GEO'`, `'GEI'`, `'SM'`. The first function also supports converting to and from the `'NEC'` frame, in which case the `longlat` variable should have meaningful data. Otherwise this variable can be anything, as is shown in the example below. Definitions of these frames can be found in the help of these functions: just type `help gap_convert_vector` in the *Command Window*.

Here is how we modified the code in the first exercise in order to convert data from the `'GSE'` to the `'GSM'` frame:

```
%% Read data from CDF file:
filename = 'C1_CP_FGM_SPIN__20010611_200000_20010611_210000_V070906';
[times, B_gse, pos_gse] = gap_convert_Cluster_fgm2mat(filename);
    % times   - array Nx1 with time tags in the MatLab's datenum format
    %                 (number of days since 01-January-2000).
    % B_gse   - array Nx3 with each line having a magnetic field vector.
    % pos_gse - array Nx3 with each line having a position vector.

%% Convert the reference frame
[B_gsm] = gap_convert_vector(B_gse, 0, times, 'GSE', 'GSM');
%B_gsm is a Nx3 array
[pos_gsm] = gap_convert_position(pos_gse, times, 'GSE', 'GSM');
%pos_gsm is a Nx3 array
```

5 Minimum variance analysis (MVA)

Here we take the example from the first exercise and modify it in order to: 1) select a subset of the data which corresponds to the actual crossing of a discontinuity and 2) do the MVA analysis on the selected data in order to determine the normal. From the paper [Dunlop and Balogh, 2005] we know that there was one discontinuity at 20:14:00. We will apply the MVA to the interval of the discontinuity: 20:13:20 to 20:15:00. The function to select a subset of data is:

```
[selected_times, varargout] = gap_select_time_range(times, trange, varargin)
%WHERE:
%   times - Nx1 array with time tags in the datenum format.
%   trange - 2x1 or 1x2 vector with two elements: start and end of the
%                    time span. Both should be given in the datenum format.
%   variable input - any number of data variables in the format NxM, so that
%                   the rows correspond to time tags given in the vector 'times'.
%OUTPUT:
%   variable output - data variables which correspond to the selected time tags.
```

The minimum variance analysis is done using the function:

```
[n, RotMatrix, variance, B_mva] = gap_mva(B)
% n         - array 1x3 indicating the normal vector.
% RotMatrix - rotation matrix: B_mva = B_selected * RotMatrix. The normal
%             direction is the first column of this matrix.
% variance  - array 3x1 with eigenvalues
% B_mva     - is the input field rotated into the MVA frame.
```

More information can be found by invoking the help of these functions. The entire code is:

```
%% Read data from CDF file:
filename = 'C1_CP_FGM_SPIN__20010611_200000_20010611_210000_V070906';
[times, B_gse, pos_gse] = gap_convert_Cluster_fgm2mat(filename);
    % times   - array Nx1 with time tags in the MatLab's datenum format
    %                (number of days since 01-January-2000).
    % B_gse   - array Nx3 with each line having a magnetic field vector.
    % pos_gse - array Nx3 with each line having a position vector.

% Define the time interval:
tstart = datenum([2001 06 11 20 13 20]);  %the array with 6 elements
        % is in the datevec format: [year month day hour minutes seconds].
        % The conversion is done using datenum(). To convert backward,
        % use datevec(tstart).
tend   = datenum([2001 06 11 20 15 00]);
% Select the time interval:
[sel_times, B_selected, pos_selected] ...
    = gap_select_time_range(times, [tstart tend], B_gse, pos_gse);
    % sel_times    - array Nx1 is a subset of 'times' in the specified
    %                interval.
    % B_selected   - a subset of B_gse corresponding to time tags in
    %                sel_times.
    % pos_selected - a subset of B_gse corresponding to time tags in
    %                sel_times.

%% Perform the Minimum Variance Analysis (MVA):
[n, RotMatrix, variance, B_mva] = gap_mva(B_selected);
    % n         - array 1x3 indicating the normal vector.
    % RotMatrix - rotation matrix: B_mva = B_selected * RotMatrix
    % variance  - array 3x1 with eigenvalues
    % B_mva     - is the input field rotated into the MVA frame.

disp(n);    % Display the normal vector

%% Plot data
plot(sel_times, B_mva);
legend('min var', 'intermediate', 'max var');
                            % Describes the lines on the plot
ylabel('B_{C1,MVA}');       % Describes the vertical axis
xlabel('time');
datetick('x');              % Auto-formats the x axis to display hours
                            % and minutes
```

The resulting plot of the script is shown in the figure 6. The program showed the normal to be equal to `[0.2874, -0.9569, -0.0417]`, which is close to the the normals provided in the paper. We can further use the available information in order to see the hodograms of the magnetic field vector in the MVA frame:

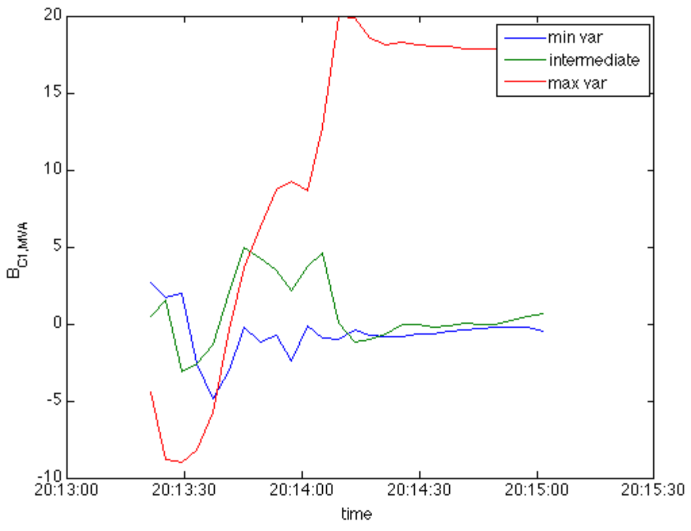


Figure 6: The result of the MVA analysis: the magnetic field from satellite C1 in the MVA frame.

```
gap_plot_hodogram(sel_times, B_mva, [tstart tend], 'window title', variance);
% sel_times       - time tags.
% B_mva           - vector to be plotted.
% [tstart tend]   - indicate the time range to be plotted.
% 'window title' - is any information, which will appear on the plot in the
%                   lower right corner.
% variance        - array with eigenvalues, so that the plot can show the
%                   ratios between variances.
```

The result is shown in the figure 7

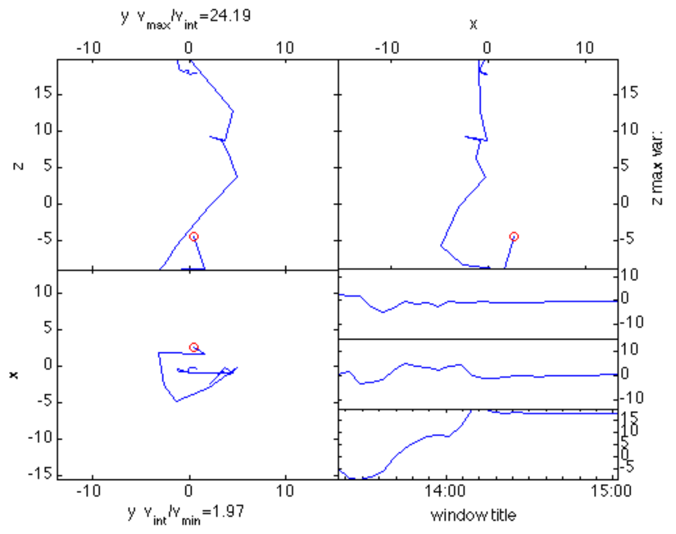


Figure 7: The result of the MVA analysis: the hodogram.

6 Estimation of currents using the Curlometer

The magnetic field measurements are not done simultaneously by the four satellites of the Cluster mission. There is always some difference in time tags between the data from various instruments and satellites. However, some gradient estimation techniques require that the data have same time tags before the procedures can be applied. The joining of data can be done using GAP's function

```
[output_data] = gap_time_join(j_times, time_gap, multi_times, method, ...
    input_data)
% j_times - final time tags
% time_gap - A double specifying the maximum tolerable time interval
%            between time tags.
% multi_times - array of S cells with time tags from S data sources.
% input_data - array of S cells describing data from S data sources.
% method - if put to 0, a nearest-point join will be applied
%          'boxcar': performs boxcar average on N nearest points
%          'linear': linear interpolation between two points
```

The Curlometer method uses the magnetic field data and positions of the four satellites in order to estimate the full gradient matrix. The function is:

```
[curlB, K, divB, GradB] = gap_curlometer(pos_vec4s, B_vec4s)
%where pos_vec4s and B_vec4s are array of 4 cells, each cell
%   representing data from the four satellites. The data has to
%   be already joined, i.e. same number of measurements.
% curlB - Nx3 array with the curl of the data provided in B_vec4s.
% K - reciprocal vectors for each measurement.
% divB - Nx1 divergence at each measurement.
% GradB - full gradient matrix at each measurement.
```

The program goes through four major steps: reading data, joining time series, estimating currents, plotting currents:

```
%% Read data from CDF files:
filename1 = 'C1_CP_FGM_SPIN__20010611_200000_20010611_210000_V070906.cdf';
filename2 = 'C2_CP_FGM_SPIN__20010611_200000_20010611_210000_V070906.cdf';
filename3 = 'C3_CP_FGM_SPIN__20010611_200000_20010611_210000_V070906.cdf';
filename4 = 'C4_CP_FGM_SPIN__20010611_200000_20010611_210000_V070906.cdf';
[times1, B1, pos1] = gap_convert_Cluster_fgm2mat(filename1);
[times2, B2, pos2] = gap_convert_Cluster_fgm2mat(filename2);
[times3, B3, pos3] = gap_convert_Cluster_fgm2mat(filename3);
[times4, B4, pos4] = gap_convert_Cluster_fgm2mat(filename4);

% Group variables into cells
c1234_times = {times1, times2, times3, times4};
c1234_B = {B1, B2, B3, B4};
c1234_pos = {pos1, pos2, pos3, pos4};


%% Join time series of data
joined_times = c1234_times{1};
time_gap = 10/24.0/3600.0; % tolerate maximum 10 seconds time gap
c1234joined_pos = ...
    gap_time_join(joined_times, time_gap, c1234_times, 'linear', c1234_pos);
```

```
c1234joined_B = ...
    gap_time_join(joined_times, time_gap, c1234_times, 'linear', c1234_B);

%% Currents: Curlometer
[curlB, K_recipr, divB] = gap_curlometer(c1234joined_pos, c1234joined_B);
J = curlB * 1e-12 / (1.25663706e-06);  %Convert curl into currents,
                                       % from nT/km to A/m^2.
J = J * 1e9;    % Convert from A/m^2 to nA/m^2.

%% Plot data
subplot(3,1,1);              % The upper plot
plot(joined_times, J);
legend('x', 'y', 'z');
ylim([-10 10]);              % Some limits, taken from the paper.
ylabel('J_{GSE} 10^{-9} Am^{-2}');
xlabel('time');              % Describes the horizontal axis.
datetick('x');               % Auto-formats the x axis to display hours
                             % and minutes

subplot(3,1,2);              % The second plot
plot(joined_times, abs(divB ./ vnorm(curlB,2)) * 100.0);
                             % vnorm(B,2) computes norms of vectors B
ylim([0 200]);               % Some limits, taken from the paper.
ylabel('divB %');
datetick('x');

subplot(3,1,3);              % The third plot
plot(times1, vnorm(B1,2), times2, vnorm(B2,2) ...
    , times3, vnorm(B3,2), times4, vnorm(B4,2));
legend('C1', 'C2', 'C3', 'C4');  % label the lines
ylim([0 30]);                % Some limits, taken to look good.
ylabel('|B| nT');
datetick('x');
```

The result is shown in the figure 8

7 Estimation of currents using two satellites

If only two satellites are given, the gradients can be estimated using the function:

```
[time_tags, curlB] ...
    = gap_curl_2sat_fd(j_times, B2joined, pos2joined, assumption, estim_e ...
     , jump_displace)
%WHERE
%   j_times    - Nx1 array with time tags, where N is the nr of
%               measurements and each time tag is in the datenum format
%               (i.e. number of days since 2000-Jan-01).
%   B2joined   - is a cell-array with 2 cells, corresponding to magnetic
%               field data from the two satellites. Each cell is Nx3
%               array. The data has to be already time joined before
%               calling this function.
%   pos2joined - is a cell-array with 2 cells, corresponding to position
%               data from the two satellites. Each cell is Nx3 array.
%   assumption - is one of options to estimate the component of the
%               gradient, which is perpendicular to the plane formed by the
```

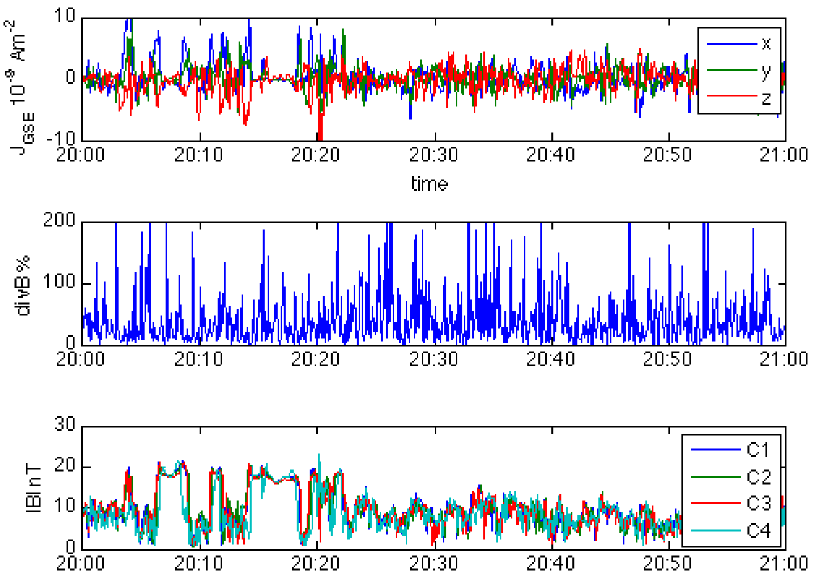


Figure 8: The result of the the Curlometer.

```
%                 two satellites and their velocity vector.
%                 'no assum'  — means that the normal component of the
%                     gradient will not be estimated. The result will contain
%                     only the gradient in the plane of the two satellites.
%                 'perp'      — gradient is assumed to be perpendicular to a
%                     given vector 'estim_e'.
%                 'paral'     — gradient is parallel to the vector 'estim_e'.
%                 'force free'— assumes that the curl of the field is
%                     parallel to the background field (j x B = 0). The
%                     background field is given in the variable 'estim_e'.
%   estim_e — array 1x3 representing the vector used in various
%                 assumptions. See the 'assumption' variable.
%   jump_displace — is an indicator for the time interval:
%                 \Delta t (seconds) = (j_times(1+jump_displace) —
%                                      j_times(1)) * 24.0 * 3600.0
%                 It is assumed that the gradient is constant at the scales
%                 of 2*\Delta t. If it is smaller, then the round—off
%                 errors are larger.
%OUTPUT
%   time_tags — Nx1 of datenums. Time tags of the resulting current estimations.
%   curlB — Nx3 the estimated curl of the magnetic field. The unit of this
%                 estimation is equal to units of B2joined divided by units
%                 of pos2joined.
%                     J = curlB * 1e—12 / (1.25663706e—06);
```

8 Other useful functions

There are a number of functions, which are might be useful to the geophysics community, and which can be found either build-in MatLab or on MatLab's FileExchange website.

```
samexaxis('xmt','on','ytac','join','yld',1);
% Joins a number of subplots to have same x axis.

savefig(filename, figure_handle, '-r300', 'pdf');
% The figures can be saved directly from the figure window. In other times
% you might want the script to save figures automatically for the input data.

hline(0, 'k--');
% Draws a horizontal line on the plot at y=0.

errorbar()
% Draws error bars on the plot.
```

9 Save/Restore workspace

The work can (and should) be saved at at some stages. MatLab allows to save the variables from the working list. For this, we can simply type `save filename.mat` in order to save everything from the workspace into the file `filename.mat`. Pictures will not be saved.

The session can be restored later by typing `load filename.mat` in the command editor.

References

M. W. Dunlop and A. Balogh. Magnetopause current as seen by Cluster. *Annales Geophysicae*, 23:901–907, March 2005.