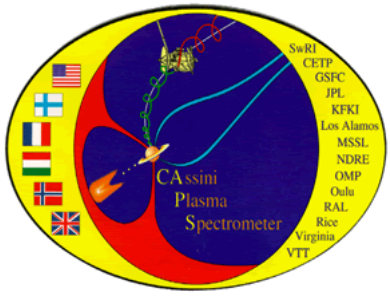

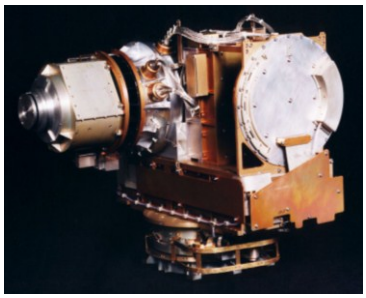

Cassini Plasma Spectrometer (CAPS) PDS User's Guide

Version
1.00
(2012 August 31)

ID. IO-AR-017

R.J. Wilson[1],
F. Crary[2,1], L.K. Gilbert[3],
D.B. Reisenfeld[4], J.T. Steinberg[5] and R. Livi[2]

1. Laboratory for Atmospheric and Space Physics, University of Colorado Boulder, Colorado, USA
2. Southwest Research Institute, San Antonio, TX, USA
3. Mullard Space Science Laboratory, University College London, UK
4. Department of Physics and Astronomy, University of Montana, Missoula, Montana, USA
5. Los Alamos National Laboratory, Los Alamos, New Mexico, USA

The SwRI ITAR representative has determined that this document does not meet the definition of "technical data" at section 120.10 of the ITAR. It is our contention, therefore, that the use of this document and the analysis of the science data from CAPS does not fall under ITAR jurisdiction.

1 Contents

1.1 Table of Chapters

1.2 Table of Figures

2 Purpose

This document is designed to instruct a novice to CAPS to load in the binary data in order that they can make basic plots and understand the field-of-view of the instrument at a given time. This is more complicated than one might imagine. It does not explain how to navigate the PDS. Nor does it explain how to do high level processing of the data (i.e. extracting plasma parameters such as density, temperature or plasma velocity). For those see the references in chapter 20, however this document provides the information needed so that one can begin to write their own analysis codes.

This document will focus on analysis of the SNG dataset, however as ELS and ION are so similar those are also covered. Basic IBS and TOF details will be given. LOG data are not covered at all.

The PDS contains two catalogs of Cassini CAPS data, COCAPS_1SAT and COCAPS_1SW. This document is about the COCAPS_1SAT dataset, the data just prior to arriving at Saturn (SAT) until end of mission. COCAPS_1SW is the solar wind (SW) dataset for the cruise phase in getting Cassini from Earth to Saturn and is not covered in this initial document, see section 4.3.3.

3 Point of Contact

Any technical questions about Cassini CAPS should go to the instrument PI, who can redirect you to the relevant person. The PI of the CAPS instrument was originally Dave Young at SwRI, who passed the role on to Frank Crary (also at SwRI then, now at LASP) in 2010. In early 2012, Dave Young took over PI responsibilities again, until he retired. From October 2012 Hunter Waite at SwRI is acting PI.

Any technical queries about the PDS files (missing/corrupt files) should go to the PDS team (see the contact us links on the PDS web pages).

Rob Wilson at the University of Colorado Boulder wrote the bulk of this document, with the aid of the CAPS team. John Steinberg (Los Alamos National Laboratory) wrote the document for IBS chapter 12. Dan Reisenfeld (University of Montana) wrote most of TOF chapter 10. Roberto Livi (SwRI) wrote chapter 19. Particular thanks go to the CAPS PIs and operation team at SwRI, the ELS operations team at MSSL, and the general CAPS science team for answering questions here and there and helping edit this document.

4 The PDS CAPS Data Files and Their Contents

The CAPS dataset is technically described by the Software Interface Specification:
CAPS STANDARD DATA PRODUCTS AND ARCHIVE VOLUME SOFTWARE INTERFACE SPECIFICATION
(CAPS Archive Volumes SIS)
SIS ID: IO-AR-017
Furman, J. et al., 2005

This is provided on the PDS under the COCAPS_1SAT volume, sub-directories DOCUMENTS>CAPS_SIS, which (at time of writing) is at the following URL:
http://ppi.pds.nasa.gov/search/view/?id=pds://PPI/COCAPS_1SAT/DOCUMENT/CAPS_SIS

4.1 Typical file name structure

Filenames generally look something like this: SNG_200528412_U3.DAT
The name itself contains 4 pieces of information, separated by underscores and a dot.

{Data product}_{TIME}_{version}.{file type}

Data product would be the dataset type, SNG, ELS, etc.

TIME is a string providing the quarter of the day the file covers. It is made up by year (4 digits) followed by day of year (3 digits) followed by hour (2 digits). This yyyyDOYHH is sometimes referred to as the DAYQ (day quarter).

Version is the file version and you should always use the largest file version number you can find (this is version 3 at time of writing). The number is pre-pended with a letter U, for Uncalibrated data – as these are the raw counts from the spacecraft. If the letter is C that would mean calibrated and likely a higher order data product. The ACT dataset does not have a letter at all.

File type is LBL, FMT, TAB or DAT, to indicate the type of information in the file, which is described in the next section.

4.2 Types of PDS files

There are 4 different types of PDS file relating to CAPS data, each having their own file extension. These are DAT, FMT, LBL and TAB. No file is self-contained, and data are always described using two or three of those file types.

The LBL file, short for label, is your starting point and is a plain text file that describes the corresponding DAT or TAB file and how to decode the data for that product. It gives you such information as the date range the data was taken from, possible targets (i.e. moons) and the number of records to expect in the corresponding file. If it's describing a TAB file it will also describe what the columns are, their units, min/max ranges, etc. However, often this description text is given its own format file, FMT, and the LBL file will tell you which FMT file to use.

The spacecraft data are always within the files with the DAT or TAB extension (and the LBL file will tell you which and what its filename is); if the data are binary the extension is DAT, if the data are human readable ASCII text, it's a TAB file – and it will always be one or the other. Most of the spacecraft data are binary, so DAT files, due to the sheer volume of data that would be too unwieldy for text files. However the majority of associated calibration files (such as energy tables) are TAB files.

For instance, to decode the first 6 hours of data on 2005-284 you need 3 files:
SNG_200528400_U3.LBL
SNG_U3.FMT
SNG_200528400_U3.DAT

But to decode an energy sweep table you'd need two files:
IMS_SWEEP_TABLE_2_Std.LBL
IMS_SWEEP_TABLE_2_Std.TAB

The different types of file are discussed more in the following subsections. A quick note for now, filenames tend to include date information in them. For instance, file SNG_200528418_U3.LBL will contain data around year 2005, day of year 284, hour of day 18 onwards (yyyyDOYHH format).

4.2.1 *.LBL

Label (LBL) files are plain text files. The following is the top part of an example SNG LBL file, SNG_200528400_U3.LBL. The filename itself tells you when the range data starts, year 2005, day of year 284, hour of day 00.

```
PDS_VERSION_ID               = PDS3
DATA_SET_ID                  = "CO-E/J/S/SW-CAPS-2-UNCALIBRATED-V1.1"

STANDARD_DATA_PRODUCT_ID     = "SNG UNCALIBRATED"
PRODUCT_ID                   = "SNG_200528400_U3"
PRODUCT_TYPE                 = "DATA"
PRODUCT_CREATION_TIME        = 2010-274T23:59

RECORD_TYPE                  = FIXED_LENGTH
RECORD_BYTES                 = 40
FILE_RECORDS                 = 45045

START_TIME                   = 2005-284T00:00:19
STOP_TIME                    = 2005-284T05:59:47
SPACECRAFT_CLOCK_START_COUNT  = "1/1507681735.000"
SPACECRAFT_CLOCK_STOP_COUNT   = "1/1507703303.000"

INSTRUMENT_HOST_NAME         = "CASSINI ORBITER"
INSTRUMENT_HOST_ID           = "CO"
TARGET_NAME                  = {"SATURN"}
INSTRUMENT_NAME              = "CASSINI PLASMA SPECTROMETER"
INSTRUMENT_ID                = "CAPS"
DESCRIPTION                  = "
     This file contains Cassini CAPS Singles data from the IMS sensor
     acquired at SATURN between
     2005-284T00:00:19.000 and 2005-284T05:59:47.000 (orbit 016)."
```

Note it contains easily readable information such as the UTC start and stop times (that of the first and last record in the data file, note that these are not exactly midnight nor 06 hours, and might cross over slightly into the next 6 hour period – see section 7.4), that there are 45045 records in that time, what that time is in spacecraft clock units, the target (generally Saturn) and a description. It also tells you that each record is fixed length (they all are) and in this case there are 40 bytes per record.
There is then some MD5 checksum info if you wish to validate you downloaded the file correctly, and then continues:

```
^TABLE                       = "SNG_200528400_U3.DAT"
OBJECT                       = TABLE
  INTERCHANGE_FORMAT           = "BINARY"
  ROWS                         = 45045
  COLUMNS                      = 11
  ROW_BYTES                    = 40
  ^STRUCTURE                   = "SNG_U3.FMT"
  DESCRIPTION                  = "
     The file SNG_U3.FMT describes the column structure and content
     of the data file."
END_OBJECT                   = TABLE
END
```

The "^TABLE" line gives you the correct filename for the file containing the actual data.
The "INTERCHANGE_FORMAT" tells you the file in binary, but you knew that from the ^TABLE name ending in DAT.

The INTERCHANGE_FORMAT could be ASCII too; telling you it's a plain text file. Beware in that sometimes these values have the quotes around them, sometimes they do not.

"ROWS" restates the number of records (FILE_RECORDS) expected in the file. (In principle one LBL file count refers to several different tables, however in CAPS case it will only ever refer to one table, hence ROWS = FILE_RECORDS always.)

"COLUMNS" is self explanatory for text files, usually with spaces or a comma separating columns. For a binary file it means the number of objects in one record.

The LBL file will either go on to describe those columns or point you to a FMT file. If the latter, this is in the "^STRUCTURE" line, giving you the filename of the file you need to refer to. However, if the format information is included in the LBL file it will be listed here, but looks identical to the FMT file's contents. As such please read the FMT section for more information.

4.2.2 *.FMT

Format (FMT) files are plain text files that accompany the LBL files for the CAPS data (but usually not the calibration data). Their file names do not include a date string in them (i.e. SNG_U3.FMT) because the format is fixed for the entire mission, hence a FMT file from 2004 for SNG will be the same as a FMT file from 2008, etc.

Each 'column' in the data file will be listed as an object, meaning multiple objects in the data file. These are in column order, but as an example here is the last 'column' of the SNG_U3.FMT file:

```
  OBJECT               = COLUMN
      NAME             = DATA
      DATA_TYPE        = MSB_UNSIGNED_INTEGER
      START_BYTE       = 25
      UNIT             = COUNTS
      ITEMS            = 8
      ITEM_BYTES       = 2
      BYTES            = 16
      MISSING_CONSTANT = 65535
      VALID_MINIMUM    = 0
      VALID_MAXIMUM    = 27500
      DESCRIPTION      = "Counts in elevations 1 through 8"
  END_OBJECT           = COLUMN
```

This is a fully described object, but not all objects have all these entries. For instance most do not have ITEMS listed at all, in which case a singular item exists (ITEMS = 1, but not shown).

Immediately its seen that list 'column' contains 8 values (ITEMS = 8), a give away that the data file is binary. Had it been a text file each column would have had its own object.

NAME gives the name of the variable – in this case just 'DATA'. These are usual descriptive (and one word), but there is also a DESCRIPTION line near the end that gives you more information.

DATA_TYPE tells you what type of data this object is.
For a TAB file, this will be either ASCII_INTEGER or ASCII_REAL, clearly telling you if the number is an integer or a real (i.e. double/float).
For a DAT file there are more options, which also depend on the number of bytes per ITEMS, however they are one of:

- MSB_UNSIGNED_INTEGER (1, 2 or 4 bytes long, [uint8, uint16 or uint32])
- MSB_INTEGER (2-bytes long only [int16])
- IEEE_REAL (float [4-bytes] or double [8-bytes])

These files are all big-endian files (hence the MSB pre-pended names, and IEEE_REAL is also big endian by design), if you are running on a little endian machine (i.e. Intel chips) you'll have to swap the byte ordering for any object with more than 1 byte. This sounds confusing but is pretty simple, see the chapters on how to read in the binaries (section 4.5, 4.6, 4.7) for example code.

START_BYTE tells you where in that record the first byte starts, while BYTES tells you how many bytes are in that 'column', and ITEM_BYTES tells you how many values are in that column. Often ITEM_BYTES = 1 and is simply not listed (hence the default is 1).
In the above example there are 8 items in this 'column', each 2-bytes long, therefor the first item is located in the record on bytes 25 and 26, and the last item is on bytes 39 and 40 (all these numbers being of type uint16).

For further help see the general PDS documentation if the sample CAPS code provided (section 4.7) still does not make sense.

UNIT gives the unit the data are in, counts in this example.

VALID_MINIMUM and VALID_MAXIMUM give the valid ranges the data could be, helpful to know if you want to automate your axis limits on plots. These also provide a good error check on your binary reading script, if your numbers are outside the valid range then you have an issue in your code.

MISSING_CONSTANT tells you the fill-value used in this particular data set. Note the MISSING_CONSTANT value varies a lot over different objects, and is not always an obvious number like 65535 or -999. Make sure you take note of this value for your object of interest and check it is not in your data.

There is no line that indicates if you should plot the data on a log or linear scale – that's your choice based on the data you are examining. Often we plot counts on a log scale, however if the values are low then a linear scale is sometimes more useful.

4.2.3 *.TAB

These are ASCII files of data – human readable and columns are obvious. Spaces are often left between columns and the FMT description often does not mention them. (Unlike the DAT files where there is no gap between objects, there are in TAB files.)

The OBJECT descriptions (giving type of data, units, limits and missing constants) are in the same style to those of DAT files and as such will not be repeated here.

If the LBL file was called IMS_SWEEP_TABLE_2_Std.LBL, then the TAB file will have the same name but with the TAB extension: IMS_SWEEP_TABLE_2_Std.TAB

4.2.4 *.DAT

This is the binary file that contains all the raw measured data from CAPS.
If the LBL file was called SNG_200528400_U3.LBL, then the DAT file will have the same name but with the DAT extension: SNG_200528400_U3.DAT

4.3 What Data Files Exist?

4.3.1 Binary Data Files

There are 8 types of binary data provided in the PDS and a brief description of each follows, but explained in far greater detail in later sections. For more information read the PDS files themselves and the CAPS Instrument paper (Young, D.T. *et al.* (2004)).

ACT: The ACTuator angle at a resolution of every 1 or 8 seconds.

ANC: ANCillary files provide information such as spacecraft position, velocity, orientation, telemetry mode, sweep tables to be used for a given A-cycle and given sensor and mcp levels.

ELS: The electron spectrometer data, counts per accumulation for each of the 8 anodes, along with information on how the data was summed onboard (if at all).

IBS: The Ion Beam Spectrometer, counts per accumulation for each of its 3 anodes.

ION: ION data contains several datasets for different ion species (usually including a proton and a water group), which takes the onboard TOF in real time and uses that to partition the ion counts into the different species over all 8 anodes. It is only available in higher telemetry modes.

LOG: LOG data are rarely used by anyone and is outside the scope of this document. For more information see the LOG FMT files.

SNG: The singles ion counting data, counts per accumulation for each of the 8 IMS start anodes, along with information on how the data was summed onboard (if at all). While the number of ion counts at a given m/q are known, their species are not.

TOF: Time of Flight ion data, gathered over longer intervals (B-cycles) and 1 anode.

The ION, LOG, SNG and TOF datasets are all from the IMS sensor.

[People from the CAPS team may mention housekeeping, HSK, files. They were for operations use and are not included in the PDS. However all values from those files required for science use are repeated in the other files, usually the ANC file, so no information is missing from this PDS collection.]

4.3.2 Calibration Tables

Sweep tables (needed to get an energy/q table) and other calibration tables (geometric factors) may be found in the PDS directory CALIB under the COCAPS_1SAT collection of data.

4.3.3 PDS Cruise Phase data collection (pre-Saturn)

Cassini took data on the way to Saturn from Earth. This is stored in a separate PDS collection COCAPS_1SW and covers the solar wind phase from 1999 through to 2004-135.

This collection is outside the scope of this document, however the information for the main mission will help greatly. However note that some data sets may have slightly different formats, so code from this document may not be compatible.

Be aware that any 1999 data where the flight software version (see section 7.5) is 3 is test data (created at a much later date), and not real solar wind data. (Flight software versions 1 and 2 were used in to 2003, so version 3 could not have been in use in 1999.)

4.3.4 PDS Saturn data collection (the main mission)

All the CAPS data for the main mission is located in the PDS under the COCAPS_1SAT collection. This begins 2004-136 (before Saturn Orbit Insertion, SOI) and runs to mission end.

4.3.5 DAYQ – quarter day files

The data files are large; having one file per day would be too large. For CAPS data the days are split into quarters; 00-06 hours, 06-12 hours, 12-18 hours and 18-24 hours. No data in each file will begin before its start hour, however the end time may over-run a few minutes into the next strict quarter. See section 7 for more details.

Filenames tend to contain a time string, i.e. SNG_200528412_U3.LBL has the date 200528412 in the name. This breaks down to year 2005, day of year 284, start hour of file 12. No end hour needs to be given as there are only 4 options.

This date (yyyyDOYHH) may be referred to as the DAYQ (day quarter) number.

4.4 Why are the Data files Binary?

People may prefer having human readable text files, but those are unwieldingly massive. i.e. the number 65535 takes 5 characters to write out in text, likely with a space before it to separate it from the previous number. That's 6 bytes. However 65535 is a two-byte uint16 number, 3 times smaller.

This smaller file size is why binary files are attractive, and when binary files are still 100 Meg in size it's easy to see why text files are not suitable for the CAPS data. In addition, a 8 x 63 x 32 element array in ASCII, for example, is not human readable so very little usability is lost by making binary files.

4.5 Data Types in CAPS and Endianness

CAPS binary data contain many records and each record is made up of many byte-words (or just single bytes); these may be floats (4-byte real), doubles (8-byte real), integer (2-bytes) and unsigned integers (1, 2 or 4 bytes).

The binary files used for CAPS stores the data in big endian format (most significant bit first), whereas most Windows, Macs and Linux computers operate in little endian format (while SPARC/Solaris machines tend to be big endian). As such when you

read in a real value (double or float) or a multi-byte integer you may have to swap the byte order from big endian to little endian for your system.

A good explanation is found on Wikipedia:
http://en.wikipedia.org/wiki/Endianness

However the basic issue is explained below.

If you have a 1-byte value then endianity is unimportant.
However if your byte-word is 2, 4 or 8 bytes long endianity is vital to be aware of.

Consider the two-byte unsigned integer for 2. The most significant byte will be all zeros (00000000) and the least significant byte would be 00000010.
However as a 2-byte word, which byte do you list first? The most or least significant? Would 2 be represented as:
`00000000 00000010` (i.e. most significant byte listed first, and most significant bits in each byte listed first), or
`00000010 00000000` (i.e. least significant byte listed first, and most significant bits in each byte listed first)
?

The answer is that both can be correct, the top one is the big-endian representation, the bottom one the little-endian representation.

All CAPS binaries are big-endian, but most of us use little endian machines and unless they're told otherwise they will assume any binary file is little endian.
Consider the former example of two being represented by `00000000 00000010`, the big endian way (as with CAPS data). If a little endian machine assumed those 16 bits in that order were little endian, it would decode it incorrectly as 512.
Clearly confusing 2 for 512 would be an issue so it is important to get little endian machines to expect big-endian numbers. IDL and Matlab can be told this when you open the file to read it. If you code in c you'll have to reverse the order of bytes yourself if you need to change the endianity.

For Matlab you tell it the file is in big endian when you open the file. This is very easy, and just needs 'ieee-be' added to the open command.
i.e. `fid = open(filename.DAT,'r','ieee-be');`
You can then read in uint8, uint16, uint32, int16, floats and doubles trivially and Matlab will correctly convert them for you if it needs to.

For IDL you open the file, but if you need to change endianity you do so on each read command with a /swapbyteorder. Also the SWAP_ENDIAN function with the /SWAP_IF_LITTLE_ENDIAN keyword may be useful:

```
value = swap_endian(value, /SWAP_IF_LITTLE_ENDIAN)
```

Later versions of IDL (tested on IDL version 8.0) allow the endianity to be described when opening the file so that all future file reads adjust endiantity for you:
`openr, fid, filename.DAT, /get_lun, /SWAP_IF_LITTLE_ENDIAN`
More tips can be found at this website:
http://www.idlcoyote.com/tips/endian_machines.html

4.6 Checking your Endianness Conversion

A quick check if you have correctly applied any Endianity conversion can be carried out using the A-cycle numbers. Each day the A-cycle number resets and start counting from 1. Therefore take any binary file that has A-cycles for the DAYQ ending in 00 (i.e. first quarter of the day) and read in the file.

If the first record has an A-cycle number of 1 then your code is correct. However if your first number is 256 then you need to correct for Endianity by telling your code the binary file is big endian formatted, and check that you successfully return 1.

Another test is to read in the first record's TIME variable and convert that to UT time to see if it matches the year and day of year of the DAYQ filename. If so then you have correctly accounted for endianess.

You can check the endianness of your computer with a little bit of code, and then if it's little endian you'll have to swap the byte order. One such example code is provided here in c. It creates a two-byte word then looks at the first byte to see if it's the least or most significant byte. *(Code courtesy of David Simpson at NASA Goddard Space Flight Center.)*

```
/*****************************************************************************/
/*  little_endian()                                                          */
/*                                                                           */
/*  Return 1 for a little-endian computer, 0 for big-endian.                 */
/*****************************************************************************/

int little_endian (void)
{
union {
   unsigned char iarr[2];
   short s;
   } u;

u.s = 0x1234;                          /* store hex 1234 into short int  */
if (u.iarr[0] == 0x34)                 /* if x34 appears in 1st element..*/
   return 1;                           /* ..then little endian           */
else
   return 0;                           /* else big endian                */
}
```

4.7 Examples of How to Read in the Binaries .DAT files

This document is paired with code files that give example source code to read in a binary file and write it out as a text file. This may be used as a basis for your future code, instead of writing out text files you can read the data into arrays/matrices directly. They are not written for speed nor efficiency – but for clarity of the reader to aid the understanding of what is happening. Each script writes out a text file called output_{language}.txt. The columns are tab delimited, and for symmetry every object has a tab after it, including the final object of the line, before the line break. This final tab to nowhere is unnecessary, but makes the DATA for loop simpler.

The same files below are to read in SNG binary files, however as ELS binary files are exactly the same file structure they also work with ELS binary files. Just change the filename to the SNG or ELS file you desire.

The following codes expect that the binary files have 40 bytes per record (see LBL file and RECORD_BYTES value) – which is correct for both SNG and ELS. If you use this code for other binary files (i.e. ION) you must change this number appropriately.

The language code examples are listed in order of how long the code is. Higher level languages of Matlab and Python have relatively short scripts, IDL is not too bad, but the low level language c is very long and the endianity conversion require more code.

The *correct* way to write generic code would be to write code to read the LBL files, from that find the correct FMT file, read that in and pull out the details of each object, then apply that to the binary file listed in the LBL file. That's complex though, hence these hard-wired examples that are quick and easy to get you going.

4.7.1 Sample code: Matlab

```
filename = 'SNG_200528400_U3.DAT'; % Could be an SNG or ELS file
% Get number of records
D = dir(filename);
n = D.bytes/40;  % Assumes 40 bytes per record
%
fid_i = fopen(filename,'r','ieee-be'); % Open the Big Endian input file
fid_o = fopen('output_matlab.txt','w'); % output file
% loop through binary file and write out text file
for z=1:n
    fprintf(fid_o,'%u\t', fread(fid_i,1,'uint16') ); % B_CYCLE_NUMBER
    fprintf(fid_o,'%u\t', fread(fid_i,1,'uint16') ); % A_CYCLE_NUMBER
    fprintf(fid_o,'%f\t', fread(fid_i,1,'double') ); % TIME
    fprintf(fid_o,'%u\t', fread(fid_i,1,'uint8' ) ); % TELEMETRY_MODE
    fprintf(fid_o,'%u\t', fread(fid_i,1,'uint8' ) ); % SPARE if SNG, COLLAPSE_FLAG if ELS
    fprintf(fid_o,'%u\t', fread(fid_i,1,'uint16') ); % OFFSET_TIME
    fprintf(fid_o,'%u\t', fread(fid_i,1,'uint16') ); % FIRST_ENERGY_STEP
    fprintf(fid_o,'%u\t', fread(fid_i,1,'uint16') ); % LAST_ENERGY_STEP
    fprintf(fid_o,'%u\t', fread(fid_i,1,'uint16') ); % FIRST_AZIMUTH_VALUE
    fprintf(fid_o,'%u\t', fread(fid_i,1,'uint16') ); % LAST_AZIMUTH_VALUE
    for zz=1:8
        fprintf(fid_o,'%u\t', fread(fid_i,1,'uint16') ); % DATA
    end
    fprintf(fid_o,'\n');
end
% close files
fclose(fid_i);
fclose(fid_o);
```

4.7.2 Sample code: Python

```
#! /usr/bin/env python

# This python script will read in ELS or SNG binary files and convert to a text file.

from struct import unpack, calcsize

fid_i = open('SNG_200528400_U3.DAT','rb')  # Open input file, ELS or SNG
fid_o = open('output_python.txt','w')      # Open file for output

# Format of Binary Data in Input File
fmt_i = '>HHdBBHHHHHHHHHHHHH'  # the > means Big Endian, H = uint16, d = double, B =
uint8

# Desired Format of Data in Output File
fmt_o = '%u\t%u\t%f\t%u\t%u\t%u\t%u\t%u\t%u\t%u\t%u\t%u\t%u\t%u\t%u\t%u\t%u\t%u\t%u\t%u\t\n'

fmtsz = calcsize(fmt_i) # this should be 40 for SNG and ELS files

while True:
    entry = fid_i.read(fmtsz) # read file
    if not entry:
        break  # EOF yields null string, exit loop
    data = unpack(fmt_i,entry)  # unpack the record using the defined input format
    fid_o.write(fmt_o%data)     # write to output file using the defined output format

fid_i.close() # close input file
fid_o.close() # close output file
```

4.7.3 Sample code: IDL

```
pro SNG_example_idl

filename =  'SNG_200528400_U3.DAT' ; Could be ELS or SNG file

; Set up different data types expected in the file
uint8  = byte(0)
uint16 = uint(0)
d = double(0);

tab = string(9B);

; find file size
File_Info_Struct = file_info(filename)
; SNG and ELS have 40 bytes per record, make it a long integer
n_records = File_Info_Struct.size/40L


; open files
openr, fid_i,filename , /get_lun
openw, fid_o,"output_idl.txt" , /get_lun

; loop through binary writing out text file
for i= 1L, n_records do begin ; Ensure i is a long integer
  ; B_CYCLE_NUMBER
  readu, fid_i, uint16
  uint16 = swap_endian(uint16, /SWAP_IF_LITTLE_ENDIAN)
  printf, fid_o, FORMAT = '(I0,A1,$)', uint16, tab

  ; A_CYCLE_NUMBER
  readu, fid_i, uint16
  uint16 = swap_endian(uint16, /SWAP_IF_LITTLE_ENDIAN)
  printf, fid_o, FORMAT = '(I0,A1,$)', uint16, tab

  ; TIME
  readu, fid_i, d
  d      = swap_endian(     d, /SWAP_IF_LITTLE_ENDIAN)
  printf, fid_o, FORMAT = '(D0,A1,$)', d, tab

  ; TELEMETRY_MODE
  readu, fid_i, uint8
  ; no need to swap endianity for one byte
  printf, fid_o, FORMAT = '(I0,A1,$)', uint8, tab

  ; SPARE if SNG, COLLAPSE_FLAG if ELS
  readu, fid_i, uint8
  ; no need to swap endianity for one byte
  printf, fid_o, FORMAT = '(I0,A1,$)', uint8, tab

  ; OFFSET_TIME
  readu, fid_i, uint16
  uint16 = swap_endian(uint16, /SWAP_IF_LITTLE_ENDIAN)
  printf, fid_o, FORMAT = '(I0,A1,$)', uint16, tab

  ; FIRST_ENERGY_STEP
  readu, fid_i, uint16
  uint16 = swap_endian(uint16, /SWAP_IF_LITTLE_ENDIAN)
  printf, fid_o, FORMAT = '(I0,A1,$)', uint16, tab

  ; LAST_ENERGY_STEP
  readu, fid_i, uint16
  uint16 = swap_endian(uint16, /SWAP_IF_LITTLE_ENDIAN)
  printf, fid_o, FORMAT = '(I0,A1,$)', uint16, tab

  ; FIRST_AZIMUTH_VALUE
  readu, fid_i, uint16
  uint16 = swap_endian(uint16, /SWAP_IF_LITTLE_ENDIAN)
  printf, fid_o, FORMAT = '(I0,A1,$)', uint16, tab
```

```
  ; LAST_AZIMUTH_VALUE
  readu, fid_i, uint16
  uint16 = swap_endian(uint16, /SWAP_IF_LITTLE_ENDIAN)
  printf, fid_o, FORMAT = '(I0,A1,$)', uint16, tab

  ; DATA
  for j=0, 7 do begin
    readu, fid_i, uint16
    uint16 = swap_endian(uint16, /SWAP_IF_LITTLE_ENDIAN)
    printf, fid_o, FORMAT = '(I0,A1,$)', uint16, tab
  endfor

  ; Now add end of line
  printf, fid_o, "" ; new line
endfor

; close files
close, fid_i
close, fid_o

end
```

Alternatively, you can replace the file open line with:

```
openr, fid_i,filename , /get_lun ,/SWAP_IF_LITTLE_ENDIAN
```

and then delete all (9) the `swap_endian` lines.
(You MUST delete them or else it will swap the enditanity twice (once via the openr and then again) meaning you've not swapped them at all.)
The code is now shorter and looks like:

```
pro SNG_example_idl

filename =  'SNG_200528400_U3.DAT' ; Could be ELS or SNG file

; Set up different data types expected in the file
uint8  = byte(0)
uint16 = uint(0)
d = double(0);

tab = string(9B);

; find file size
File_Info_Struct = file_info(filename)
; SNG and ELS have 40 bytes per record, make it a long integer
n_records = File_Info_Struct.size/40L


; open files
openr, fid_i,filename , /get_lun, /SWAP_IF_LITTLE_ENDIAN
openw, fid_o,"output_idl.txt" , /get_lun

; loop through binary writing out text file
for i= 1L, n_records do begin ; Ensure i is a long integer
  ; B_CYCLE_NUMBER
  readu, fid_i, uint16
  printf, fid_o, FORMAT = '(I0,A1,$)', uint16,tab

  ; A_CYCLE_NUMBER
  readu, fid_i, uint16
  printf, fid_o, FORMAT = '(I0,A1,$)', uint16,tab

  ; TIME
  readu, fid_i, d
  printf, fid_o, FORMAT = '(D0,A1,$)', d, tab

  ; TELEMETRY_MODE
  readu, fid_i, uint8
```

```
  printf, fid_o, FORMAT = '(I0,A1,$)', uint8, tab

  ; SPARE if SNG, COLLAPSE_FLAG if ELS
  readu, fid_i, uint8
  printf, fid_o, FORMAT = '(I0,A1,$)', uint8, tab

  ; OFFSET_TIME
  readu, fid_i, uint16
  printf, fid_o, FORMAT = '(I0,A1,$)', uint16,tab

  ; FIRST_ENERGY_STEP
  readu, fid_i, uint16
  printf, fid_o, FORMAT = '(I0,A1,$)', uint16,tab

  ; LAST_ENERGY_STEP
  readu, fid_i, uint16
  printf, fid_o, FORMAT = '(I0,A1,$)', uint16,tab

  ; FIRST_AZIMUTH_VALUE
  readu, fid_i, uint16
  printf, fid_o, FORMAT = '(I0,A1,$)', uint16,tab

  ; LAST_AZIMUTH_VALUE
  readu, fid_i, uint16
  printf, fid_o, FORMAT = '(I0,A1,$)', uint16,tab

  ; DATA
  for j=0, 7 do begin
    readu, fid_i, uint16
    printf, fid_o, FORMAT = '(I0,A1,$)', uint16,tab
  endfor

  ; Now add end of line
  printf, fid_o, "" ; new line
endfor

; close files
close, fid_i
close, fid_o

end
```

4.7.4 Sample code: c

```c
/* This example c code will convert a SNG or ELS binary file into a text file.       */
/* Each line is one record, and the columns represent the different Objects.         */
/* To compile under linux: gcc -O3 -lm -Wall -W -o SNG_example_c.exe SNG_example_c.c */

#include <stdio.h>
#include <string.h>

void SwapUnsignedInt16( unsigned short int *a );
void SwapDouble( double *a );

/* Add routines that will swap bytes in case we need to correct for Endianity */
void SwapUnsignedInt16( unsigned short int *a ){
    unsigned char   b[2], c[2];
    memcpy( b, a, 2);
    c[0] = b[1];
    c[1] = b[0];
    memcpy( a, c, 2);
}
void SwapDouble( double *a ){
    unsigned char   b[8], c[8];
    memcpy( b, a, 8);
    c[0] = b[7];
    c[1] = b[6];
    c[2] = b[5];
    c[3] = b[4];
    c[4] = b[3];
    c[5] = b[2];
    c[6] = b[1];
    c[7] = b[0];
    memcpy( a, c, 8);
}

/* Main Routine */
int main(void) {
/* Declarations for Endiatity checking */
      unsigned char EndianTest[2] = { 1, 0 };
      short int  iLE; /* iLE = Is Little Endian */
/* Declarations for file reading and file sizes */
    FILE          *fi, *fo;
    unsigned long rec, fileLen, recLen;
    int           bytes_per_rec;
/* Declarations for objects in FMT file */
    unsigned char       ui8;
    unsigned short int  ui16;
    double d;
    int    j; /* Just needed for the loop for the DATA object */

/* Open Binary for reading - put filename here */
      fi = fopen( "SNG_200528400_U3.DAT", "rb" );
/* SNG and ELS have 40 bytes per record, so get number of records */
      bytes_per_rec = 40;
/* Open output files for writing */
      fo = fopen( "output_c.txt", "wt" );


/* Get file length in bytes, then number of records */
      fseek(fi, 0L, SEEK_END);
      fileLen = ftell(fi);
      fseek(fi, 0L, SEEK_SET); /* return to start of file */
      recLen = fileLen/bytes_per_rec;

/* Check what endianity this computer is */
      iLE = *(short *) EndianTest;   /* 1 = little endian, 256 = big endian */
```

```
/* Read each record in the file */
      for (rec = 0; rec < recLen; rec++) {
      /* B_CYCLE_NUMBER */
        fread(&ui16, 2, 1, fi);                    /* read value from file          */
        if (iLE == 1) SwapUnsignedInt16( &ui16 ); /* Convert Endianity if required */
        fprintf(fo, "%u\t", (unsigned int)ui16);  /* Write out text to file        */
      /* A_CYCLE_NUMBER */
        fread(&ui16, 2, 1, fi);
        if (iLE == 1) SwapUnsignedInt16( &ui16 );
        fprintf(fo, "%u\t", (unsigned int)ui16);
      /* TIME */
        fread(&d   , 8, 1, fi);
        if (iLE == 1) SwapDouble( &d );
        fprintf(fo, "%f\t", d   );
      /* TELEMETRY_MODE */
        fread(&ui8, 1, 1, fi);
        /* 1-byte has no endianity issues */
        fprintf(fo, "%u\t", (unsigned int)ui8);
      /* SPARE if SNG, COLLAPSE_FLAG if ELS */
        fread(&ui8, 1, 1, fi);
        /* 1-byte has no endianity issues */
        fprintf(fo, "%u\t", (unsigned int)ui8);
      /* OFFSET_TIME */
        fread(&ui16, 2, 1, fi);
        if (iLE == 1) SwapUnsignedInt16( &ui16 );
        fprintf(fo, "%u\t", (unsigned int)ui16);
      /* FIRST_ENERGY_STEP */
        fread(&ui16, 2, 1, fi);
        if (iLE == 1) SwapUnsignedInt16( &ui16 );
        fprintf(fo, "%u\t", (unsigned int)ui16);
      /* LAST_ENERGY_STEP */
        fread(&ui16, 2, 1, fi);
        if (iLE == 1) SwapUnsignedInt16( &ui16 );
        fprintf(fo, "%u\t", (unsigned int)ui16);
      /* FIRST_AZIMUTH_VALUE */
        fread(&ui16, 2, 1, fi);
        if (iLE == 1) SwapUnsignedInt16( &ui16 );
        fprintf(fo, "%u\t", (unsigned int)ui16);
      /* LAST_AZIMUTH_VALUE */
        fread(&ui16, 2, 1, fi);
        if (iLE == 1) SwapUnsignedInt16( &ui16 );
        fprintf(fo, "%u\t", (unsigned int)ui16);
      /* DATA */
        for (j = 0; j < 8; j++) {
            fread(&ui16, 2, 1, fi);
            if (iLE == 1) SwapUnsignedInt16( &ui16 );
            fprintf(fo, "%u\t", (unsigned int)ui16);
        }
        fprintf(fo,"\n"); /* end line */
      }

/* Close files and exit */
      fclose(fi);
      fclose(fo);
      return 0;
}
```

4.7.5 Sample output

The following gives the first 130 rows of the 45045 lines output from the above codes for file SNG_200528400_U3.DAT, which can be used to check your code is working correctly. There are two full sweeps of 63 individual energy step records that are summed over 8 Azimuths (see section 7.2), then the next 4 of the next sweep.

```
65535   1   182260883.645872 2   0   0   1   1   1   8   0   0   0   0   0   0   1   0
65535   1   182260883.645872 2   0   62  2   2   1   8   0   0   0   0   0   0   0   0
65535   1   182260883.645872 2   0   125 3   3   1   8   11  13  18  18  7   9   17  11
65535   1   182260883.645872 2   0   187 4   4   1   8   16  19  22  19  16  22  24  11
65535   1   182260883.645872 2   0   250 5   5   1   8   18  18  15  14  16  22  13  20
65535   1   182260883.645872 2   0   312 6   6   1   8   13  18  17  17  17  17  19  17
65535   1   182260883.645872 2   0   375 7   7   1   8   11  18  15  14  24  15  18  19
65535   1   182260883.645872 2   0   437 8   8   1   8   20  14  18  20  21  20  14  12
65535   1   182260883.645872 2   0   500 9   9   1   8   16  10  18  26  15  23  24  19
65535   1   182260883.645872 2   0   562 10  10  1   8   17  14  17  12  21  17  18  19
65535   1   182260883.645872 2   0   625 11  11  1   8   23  8   17  13  21  25  22  17
65535   1   182260883.645872 2   0   687 12  12  1   8   11  17  20  24  20  28  26  25
65535   1   182260883.645872 2   0   750 13  13  1   8   17  14  20  22  20  26  31  27
65535   1   182260883.645872 2   0   812 14  14  1   8   18  19  19  26  21  19  32  25
65535   1   182260883.645872 2   0   875 15  15  1   8   11  12  18  21  20  25  39  23
65535   1   182260883.645872 2   0   937 16  16  1   8   18  21  24  26  33  48  54  36
65535   1   182260883.645872 2   0   1000 17 17  1   8   19  24  22  22  36  35  55  34
65535   1   182260883.645872 2   0   1062 18 18  1   8   21  29  14  26  25  39  49  40
65535   1   182260883.645872 2   0   1125 19 19  1   8   18  17  20  32  30  43  63  34
65535   1   182260883.645872 2   0   1187 20 20  1   8   26  23  16  22  32  46  40  43
65535   1   182260883.645872 2   0   1250 21 21  1   8   16  19  14  17  24  37  35  30
65535   1   182260883.645872 2   0   1312 22 22  1   8   19  10  15  21  27  25  34  26
65535   1   182260883.645872 2   0   1375 23 23  1   8   11  14  20  30  37  25  25  26
65535   1   182260883.645872 2   0   1437 24 24  1   8   22  17  16  18  19  24  29  13
65535   1   182260883.645872 2   0   1500 25 25  1   8   14  13  15  22  25  26  20  21
65535   1   182260883.645872 2   0   1562 26 26  1   8   17  18  13  12  16  19  28  27
65535   1   182260883.645872 2   0   1625 27 27  1   8   15  14  15  13  27  28  27  22
65535   1   182260883.645872 2   0   1687 28 28  1   8   21  18  15  17  21  36  39  28
65535   1   182260883.645872 2   0   1750 29 29  1   8   21  22  23  21  28  37  35  25
65535   1   182260883.645872 2   0   1812 30 30  1   8   14  18  10  14  20  36  44  21
65535   1   182260883.645872 2   0   1875 31 31  1   8   12  15  16  21  31  28  32  21
65535   1   182260883.645872 2   0   1937 32 32  1   8   20  22  22  14  22  23  25  24
65535   1   182260883.645872 2   0   2000 33 33  1   8   12  20  17  22  21  19  28  20
65535   1   182260883.645872 2   0   2062 34 34  1   8   18  11  8   17  30  22  30  24
65535   1   182260883.645872 2   0   2125 35 35  1   8   21  21  16  12  21  26  30  22
65535   1   182260883.645872 2   0   2187 36 36  1   8   14  11  18  18  20  22  25  18
65535   1   182260883.645872 2   0   2250 37 37  1   8   15  13  12  20  11  15  15  18
65535   1   182260883.645872 2   0   2312 38 38  1   8   16  20  15  18  19  24  20  18
65535   1   182260883.645872 2   0   2375 39 39  1   8   9   12  17  12  13  11  24  22
65535   1   182260883.645872 2   0   2437 40 40  1   8   21  19  17  13  10  9   18  22
65535   1   182260883.645872 2   0   2500 41 41  1   8   16  16  13  20  11  20  22  11
65535   1   182260883.645872 2   0   2562 42 42  1   8   22  14  10  13  15  21  13  14
65535   1   182260883.645872 2   0   2625 43 43  1   8   15  15  15  18  14  13  11  7
65535   1   182260883.645872 2   0   2687 44 44  1   8   19  17  29  17  16  17  12  11
65535   1   182260883.645872 2   0   2750 45 45  1   8   17  10  17  16  10  12  8   14
65535   1   182260883.645872 2   0   2812 46 46  1   8   17  18  11  21  18  18  13  17
65535   1   182260883.645872 2   0   2875 47 47  1   8   14  17  9   9   12  10  12  10
65535   1   182260883.645872 2   0   2937 48 48  1   8   18  19  10  11  8   11  15  19
65535   1   182260883.645872 2   0   3000 49 49  1   8   15  17  13  17  14  10  14  13
65535   1   182260883.645872 2   0   3062 50 50  1   8   14  17  14  11  15  16  12  11
65535   1   182260883.645872 2   0   3125 51 51  1   8   12  12  17  15  13  16  16  8
65535   1   182260883.645872 2   0   3187 52 52  1   8   12  13  10  14  11  15  10  16
65535   1   182260883.645872 2   0   3250 53 53  1   8   14  18  13  16  15  15  17  14
65535   1   182260883.645872 2   0   3312 54 54  1   8   15  16  13  15  11  15  16  16
65535   1   182260883.645872 2   0   3375 55 55  1   8   16  14  12  18  14  8   10  16
65535   1   182260883.645872 2   0   3437 56 56  1   8   15  8   6   14  6   12  14  10
65535   1   182260883.645872 2   0   3500 57 57  1   8   17  15  10  15  14  14  15  13
65535   1   182260883.645872 2   0   3562 58 58  1   8   17  18  14  11  10  14  13  20
65535   1   182260883.645872 2   0   3625 59 59  1   8   16  14  10  16  13  10  17  17
65535   1   182260883.645872 2   0   3687 60 60  1   8   16  21  11  10  11  14  15  14
65535   1   182260883.645872 2   0   3750 61 61  1   8   22  15  13  13  11  12  7   10
65535   1   182260883.645872 2   0   3812 62 62  1   8   13  16  4   19  14  12  15  12
65535   1   182260883.645872 2   0   3875 63 63  1   8   19  18  6   14  11  11  16  19
65535   2   182260915.645667 2   0   0   1   1   1   8   0   0   0   0   0   0   0   1
65535   2   182260915.645667 2   0   62  2   2   1   8   0   0   0   0   0   0   0   0
65535   2   182260915.645667 2   0   125 3   3   1   8   15  21  14  8   16  10  16  10
```

```
65535   2   182260915.645667 2   0   187 4   4   1   8   19  15  19  15  16  16  13  15
65535   2   182260915.645667 2   0   250 5   5   1   8   26  18  15  16  14  26  21  18
65535   2   182260915.645667 2   0   312 6   6   1   8   13  15  11  16  21  31  16  23
65535   2   182260915.645667 2   0   375 7   7   1   8   18  21  12  14  13  17  21  25
65535   2   182260915.645667 2   0   437 8   8   1   8   22  21  20  15  15  22  23  24
65535   2   182260915.645667 2   0   500 9   9   1   8   15  18  20  17  23  13  18  28
65535   2   182260915.645667 2   0   562 10  10  1   8   19  25  20  12  14  27  30  31
65535   2   182260915.645667 2   0   625 11  11  1   8   17  12  15  15  36  27  24  34
65535   2   182260915.645667 2   0   687 12  12  1   8   27  15  19  8   19  20  25  31
65535   2   182260915.645667 2   0   750 13  13  1   8   17  16  13  26  23  34  41  49
65535   2   182260915.645667 2   0   812 14  14  1   8   27  21  18  19  21  41  63  83
65535   2   182260915.645667 2   0   875 15  15  1   8   18  24  31  26  37  69  104 114
65535   2   182260915.645667 2   0   937 16  16  1   8   18  23  31  36  81  114 148 157
65535   2   182260915.645667 2   0   1000 17 17  1   8   25  29  41  54  92  157 245 223
65535   2   182260915.645667 2   0   1062 18 18  1   8   22  36  36  41  130 245 421 359
65535   2   182260915.645667 2   0   1125 19 19  1   8   44  50  47  63  126 245 434 337
65535   2   182260915.645667 2   0   1187 20 20  1   8   49  50  49  63  139 261 359 348
65535   2   182260915.645667 2   0   1250 21 21  1   8   24  41  36  42  92  196 261 306
65535   2   182260915.645667 2   0   1312 22 22  1   8   27  30  26  39  73  168 245 230
65535   2   182260915.645667 2   0   1375 23 23  1   8   20  23  16  28  58  122 157 173
65535   2   182260915.645667 2   0   1437 24 24  1   8   23  23  24  34  45  71  104 107
65535   2   182260915.645667 2   0   1500 25 25  1   8   25  24  23  38  40  59  69  86
65535   2   182260915.645667 2   0   1562 26 26  1   8   15  17  19  33  47  53  71  95
65535   2   182260915.645667 2   0   1625 27 27  1   8   11  15  17  31  40  40  62  81
65535   2   182260915.645667 2   0   1687 28 28  1   8   23  20  15  24  38  58  89  95
65535   2   182260915.645667 2   0   1750 29 29  1   8   23  23  19  36  44  59  86  122
65535   2   182260915.645667 2   0   1812 30 30  1   8   23  15  18  18  33  71  107 139
65535   2   182260915.645667 2   0   1875 31 31  1   8   15  23  28  23  47  71  122 152
65535   2   182260915.645667 2   0   1937 32 32  1   8   23  23  17  20  38  69  111 126
65535   2   182260915.645667 2   0   2000 33 33  1   8   16  22  19  23  33  55  92  122
65535   2   182260915.645667 2   0   2062 34 34  1   8   14  17  18  16  43  54  76  107
65535   2   182260915.645667 2   0   2125 35 35  1   8   23  24  24  26  35  56  107 111
65535   2   182260915.645667 2   0   2187 36 36  1   8   18  14  19  17  31  56  58  67
65535   2   182260915.645667 2   0   2250 37 37  1   8   14  12  18  30  20  37  61  44
65535   2   182260915.645667 2   0   2312 38 38  1   8   18  15  10  16  30  37  46  56
65535   2   182260915.645667 2   0   2375 39 39  1   8   16  16  16  16  25  30  41  34
65535   2   182260915.645667 2   0   2437 40 40  1   8   14  9   11  22  24  28  36  27
65535   2   182260915.645667 2   0   2500 41 41  1   8   12  19  20  19  23  19  26  19
65535   2   182260915.645667 2   0   2562 42 42  1   8   14  22  22  19  17  11  17  21
65535   2   182260915.645667 2   0   2625 43 43  1   8   15  10  10  14  15  13  17  18
65535   2   182260915.645667 2   0   2687 44 44  1   8   11  14  7   15  18  14  21  13
65535   2   182260915.645667 2   0   2750 45 45  1   8   15  16  15  16  13  11  13  14
65535   2   182260915.645667 2   0   2812 46 46  1   8   14  13  14  21  17  14  13  14
65535   2   182260915.645667 2   0   2875 47 47  1   8   13  12  21  13  13  17  15  14
65535   2   182260915.645667 2   0   2937 48 48  1   8   10  18  15  14  14  19  12  13
65535   2   182260915.645667 2   0   3000 49 49  1   8   17  16  14  11  11  13  11  8
65535   2   182260915.645667 2   0   3062 50 50  1   8   14  14  14  10  10  13  10  9
65535   2   182260915.645667 2   0   3125 51 51  1   8   16  11  12  18  13  10  21  16
65535   2   182260915.645667 2   0   3187 52 52  1   8   10  12  9   11  10  11  17  14
65535   2   182260915.645667 2   0   3250 53 53  1   8   14  12  13  11  7   17  11  9
65535   2   182260915.645667 2   0   3312 54 54  1   8   15  11  4   10  11  12  12  14
65535   2   182260915.645667 2   0   3375 55 55  1   8   9   14  18  19  10  17  10  10
65535   2   182260915.645667 2   0   3437 56 56  1   8   19  20  13  9   11  8   16  10
65535   2   182260915.645667 2   0   3500 57 57  1   8   14  13  15  13  12  19  11  9
65535   2   182260915.645667 2   0   3562 58 58  1   8   11  15  17  11  11  13  16  8
65535   2   182260915.645667 2   0   3625 59 59  1   8   14  12  15  11  14  19  16  13
65535   2   182260915.645667 2   0   3687 60 60  1   8   16  13  9   17  13  11  19  11
65535   2   182260915.645667 2   0   3750 61 61  1   8   10  11  17  12  16  15  17  16
65535   2   182260915.645667 2   0   3812 62 62  1   8   12  17  12  9   12  9   18  13
65535   2   182260915.645667 2   0   3875 63 63  1   8   13  16  13  10  10  21  18  15
65535   3   182260947.645463 2   0   0   1   1   1   8   0   1   0   0   0   0   0   0
65535   3   182260947.645463 2   0   62  2   2   1   8   0   0   0   0   0   0   0   0
65535   3   182260947.645463 2   0   125 3   3   1   8   17  10  11  15  12  12  16  15
65535   3   182260947.645463 2   0   187 4   4   1   8   15  15  18  19  19  20  17  19
```

4.8 How to Read in the Text files .TAB

Binary files can be confusing to read in, hence the lengthy explanations above. Text files such as .TAB are not, these are merely columns of numbers and should not pose any problem.

4.9 How to Plot Line Slices (SNG, ELS, IBS, ION)

First you must extract the azimuth/azimuth-sum of data you wish to plot. For simplicity in this section azimuth-sum can also mean an azimuth (without summing). The following example is for SNG data, and uses the sample output listed above.

Every DAYQ file begins with a record that has FIRST_ENERGY_STEP = 1, which marks the start of an azimuth-sum. Subsequent records increase in energy step, and the azimuth-sum is over when LAST_ENERGY_STEP = 63. That is the first azimuth-sum of the file. The next record should have FIRST_ENERGY_STEP = 1 again, marking the beginning of the next azimuth sum, and the pattern follows.

The above sample output has FIRST_ENERGY_STEP and LAST_ENERGY_STEP as columns 7 and 8 respectively, and if you wish to plot the counts per accumulation data from anode 7, that is column 17.

Let's take the second azimuth sum of that DAYQ, which using the above description will fall on lines 64 to 126 of the sample output, and just take those 3 columns to get a new table:

Column A FIRST ENERGY STEP	Column B LAST ENERGY STEP	Column C Anode 7 Counts/Accumulation
1	1	0
2	2	0
3	3	16
4	4	13
5	5	21
6	6	16
7	7	21
8	8	23
9	9	18
10	10	30
11	11	24
12	12	25
13	13	41
14	14	63
15	15	104
16	16	148
17	17	245
18	18	421
19	19	434
20	20	359
21	21	261
22	22	245
23	23	157
24	24	104
25	25	69
26	26	71
27	27	62
28	28	89
29	29	86
30	30	107
31	31	122
32	32	111
33	33	92
34	34	76
35	35	107
36	36	58
37	37	61
38	38	46
39	39	41
40	40	36
41	41	26
42	42	17
43	43	17
44	44	21
45	45	13
46	46	13
47	47	15
48	48	12
49	49	11
50	50	10
51	51	21
52	52	17
53	53	11
54	54	12
55	55	10
56	56	16
57	57	11
58	58	16
59	59	16
60	60	19
61	61	17
62	62	18
63	63	18

In this case it is clear that LAST_ENERGY_STEP = FIRST_ENERGY_STEP for every row, so the line plot is simply "column C of this table vs. column A (or B)" of this table, shown in **Error! Reference source not found.**Figure 4.1.
Error! Reference source not found.Error! Reference source not found.

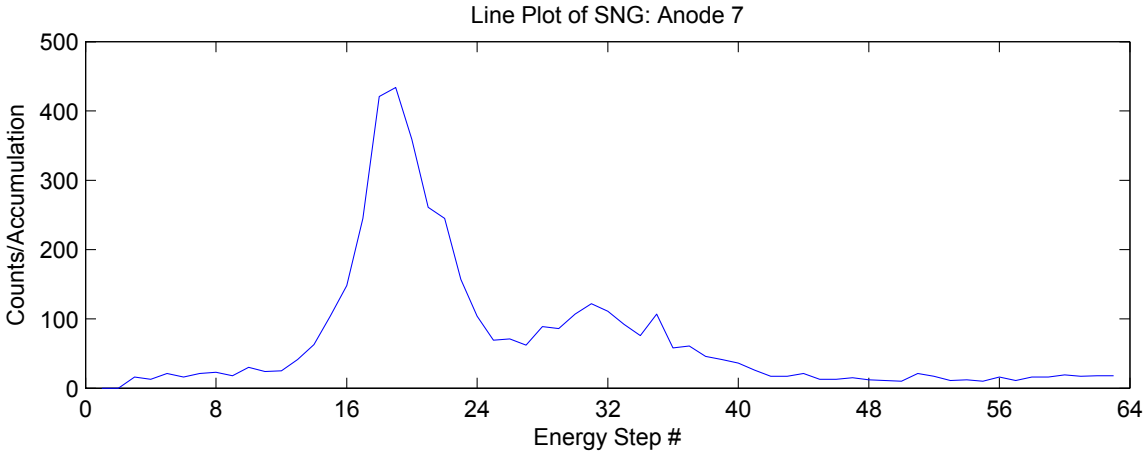

Figure 4.1: Example line plot of column C vs. column A.

Figure 4.1If FIRST_ENERGY_STEP is not equal to LAST_ENERGY_STEP for every row, the line plot would use the mean of columns A and B for the x-axis, "column C of this table vs. (column A + column B)/2".

The above description is for SNG data, however it is identical for ELS data.

ION data would be much the same the same, except when pulling out your 63 records (or 32 if energy summing) for the azimuth-sum you must ensure that the SAM_ION_NUMBER is the one you desire to plot.

IBS data are much the same again, however the energy steps for an azimuth sum do not go from 1 to 63. They do still start the DAYQ file at the lowest value for that mode and increase in subsequent records though, so whenever the FIRST_ENERGY_STEP of the current record is less than the LAST_ENERGY_STEP of the previous record you know you are starting a new azimuth-sum.

If you wish to continue and turn the x-axis into units of eV/q you must convert energy step using the voltage sweep tables described in section 7.8. Likewise if you wish to turn the counts into other units you can do that too, but the essence of generating the line plot remains the same.

4.10 How to Plot Spectrograms (SNG, ELS, IBS, ION)

Spectrograms are generated by plotting one record at a time on the figure, and just keep adding as many records as you want. Again SNG data set will be used for the example.

From your text output (i.e. sample output above) extract the subset of records you wish to include in the spectrogram. This will usually be a whole number of azimuths/azimuth-sums (i.e. first record has FIRST_ENERGY_STEP = 1 and last record has LAST_ENERGY_STEP = 63), but does not have to be. Each record will form one pixel of the spectrogram, where a pixel is a rectangle of color representing the quantity you wish to show. For instance, if it were data from anode 8 you would take the value from column 18 to use as your data, and assign that a color based on your colormap (colorbar) of choice. The question is where to put your pixel.

Each record has an A_CYCLE_NUMBER, FIRST_AZIMUTH_VALUE, LAST_AZIMUTH_VALUE, FIRST_ENERGY_STEP and LAST_ENERGY_STEP, and these 5 scalar values provide the location, height and width of the pixel for that record.

To begin with you must create a 2-D map. On the vertical axis will be the energy steps. Since CAPS has highest voltage at lowest energy step it's normal to reverse this axis so lower energy steps are at the top. The horizontal axis is made up of number of azimuths, but notated by {A CYCLE NUMBER}:{AZIMUTH NUMBER}.

From this we begin to step through all our records and plot the pixels. Each pixel must have a height, a width, and be centered appropriately.

For instance, if FIRST_ENERGY_STEP = LAST_ENERGY_STEP = 8, then the height of the pixel must be 1, starting at 7.5 and extending to 8.5 on the horizontal axis. Or if FIRST_ENERGY_STEP = 5 and LAST_ENERGY_STEP = 6, then the height of the pixel must be 2, starting at 4.5 and extending to 6.5 on the horizontal axis.

The horizontal axis is similar, if A_CYCLE_NUMBER = 3, FIRST_AZIMUTH_VALUE = 5 and LAST_AZIMUTH VALUE =5, the width is 1 azimuth, from 3:4.5 to 3:5.5. Or if A_CYCLE_NUMBER = 2, FIRST_AZIMUTH_VALUE = 1 and LAST_AZIMUTH VALUE =8, the width is 1 azimuth, from 2:0.5 to 2:8.5.
However A Cycles bound each other, so 1:8.5 = 2:0.5, etc.

The following diagram (Figure 4.2) maps this out, and shows examples of different records placed in their correct position. Note that no colored pixels should ever cross an A-cycle boundary.

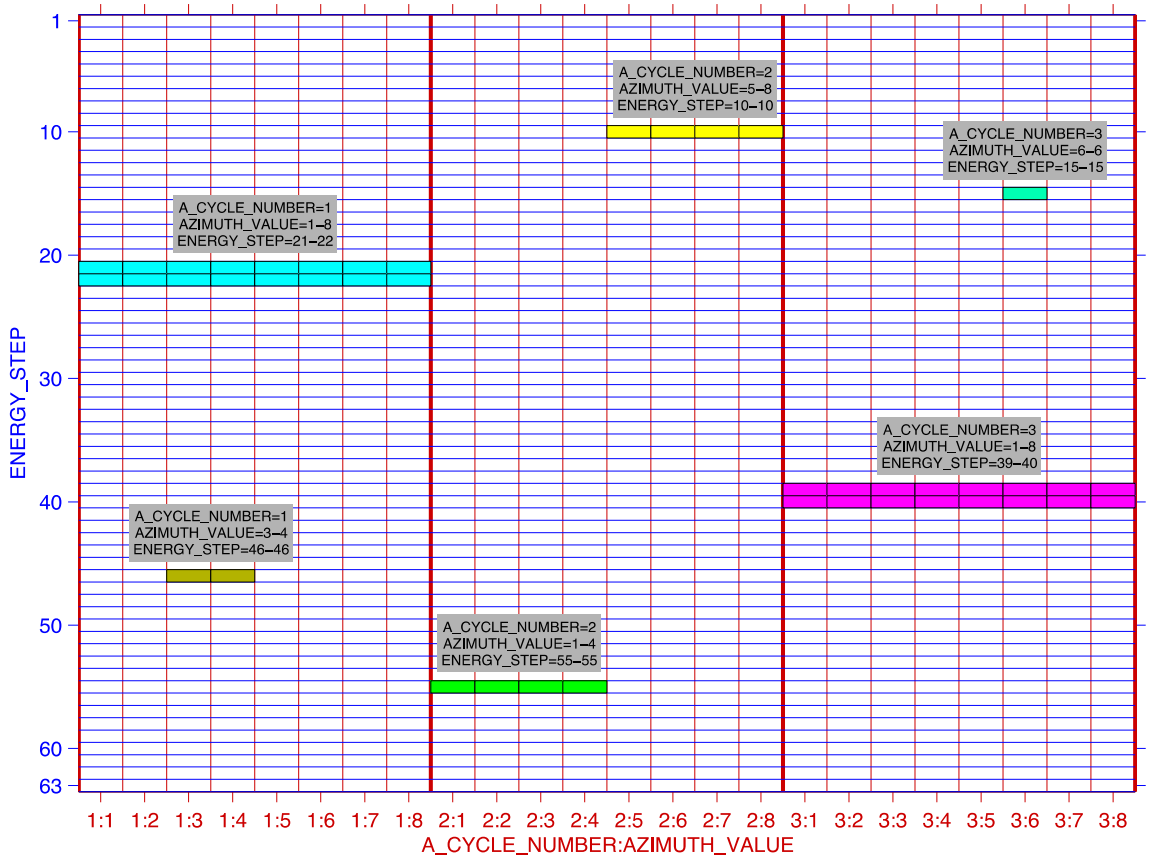

Figure 4.2: Anatomy of a spectrogram (IBS/ELS/ION/SNG).

This is the basics of spectrograms. If you wish to turn the vertical axis into units of eV/q you may (see later sections for conversions), or the horizontal axis into units of time or distance from Saturn you may. However this will change the widths and heights of the pixels to variable numbers (i.e. widths are no longer whole numbers). In particular if you change energy step to units of eV/q you'll likely want to plot the y-axis on a logarithmic scale, where the absolute height of each pixel changes depending on energy step.

Instead of having an x-axis in units of 'A_CYCLE_NUMBER:AZIMUTH_VALUE' you may prefer cumulative azimuths number, CA, where:

CA_{SNG} = (A_CYCLE_NUMBER – 1)*8 + AZIMUTH_VALUE

In the above figure that'd label the x-axis pixels from 1 to 24 and may be an easier unit to work in.

For ELS data the spectrograms are done in the same way, except that AZIMUTH_VALUES go from 1 to 16 (rather than 1 to 8 for SNG). Likewise:

CA_{ELS} = (A_CYCLE_NUMBER – 1)*16 + AZIMUTH_VALUE

4.11 Example Plots

The top panel of Figure 4.3 shows a spectrogram of SNG anode 8 for the sample output for the first hour of the DAYQ (SNG_200528400_U3.DAT), counts per accumulation (i.e. the value in the DATA object) plotted as energy step against time (HH:MM). The top panel shows the data on a linear scaled colorbar and the middle panel the same data on a log10 colorbar – clearly the logged data are more useful at showing the features of the data and is generally the preferred way. The 3rd panel shows a zoomed portion of the second panel to high-light the change in pixel size. To begin with the SNG data records had azimuth-sums of 8 Azimuths, however around 00:47 the azimuth-sums are just 4 Azimuths for 8 A-cycles (A_CYCLE_NUMBERS 88 to 95).

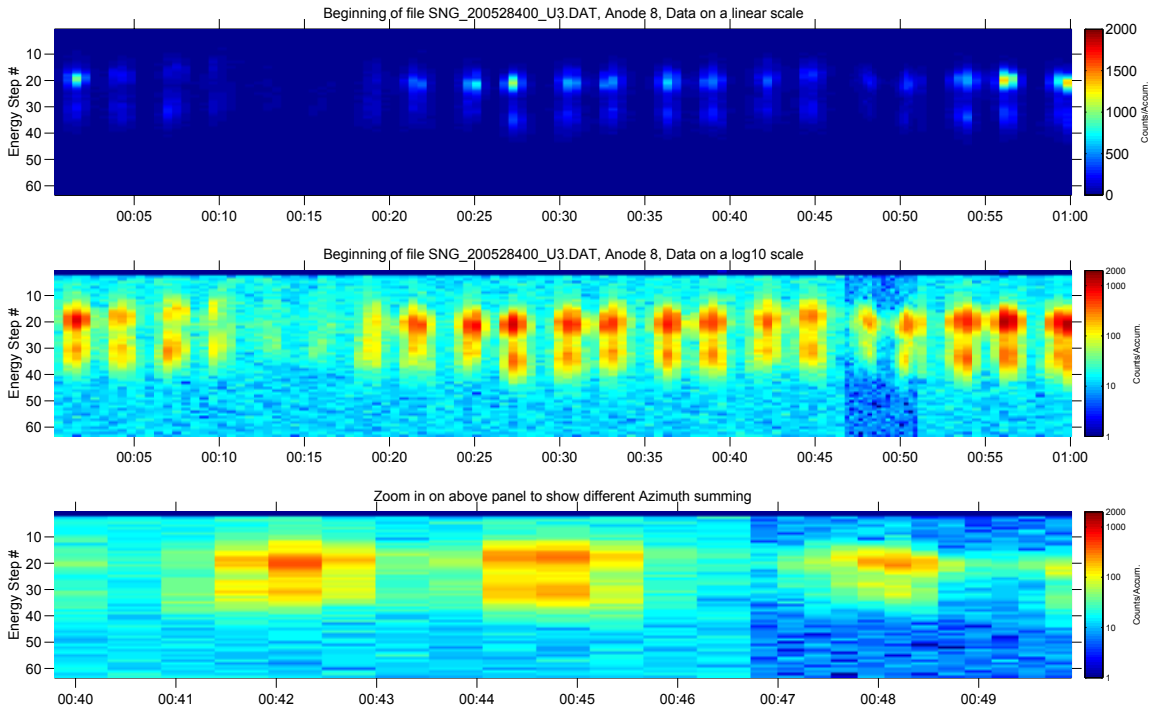

Figure 4.3: Example SNG spectrograms.

The equivalent spectrograms for ELS (anode 8 again) for the ELS data of the same DAYQ (ELS_200528400_U3.DAT), are shown in Figure 4.4. Again the top panel is with a linear color scale, the middle a log10 color scale and the bottom a zoom in of the middle to show the when the azimuth-sums value changes:

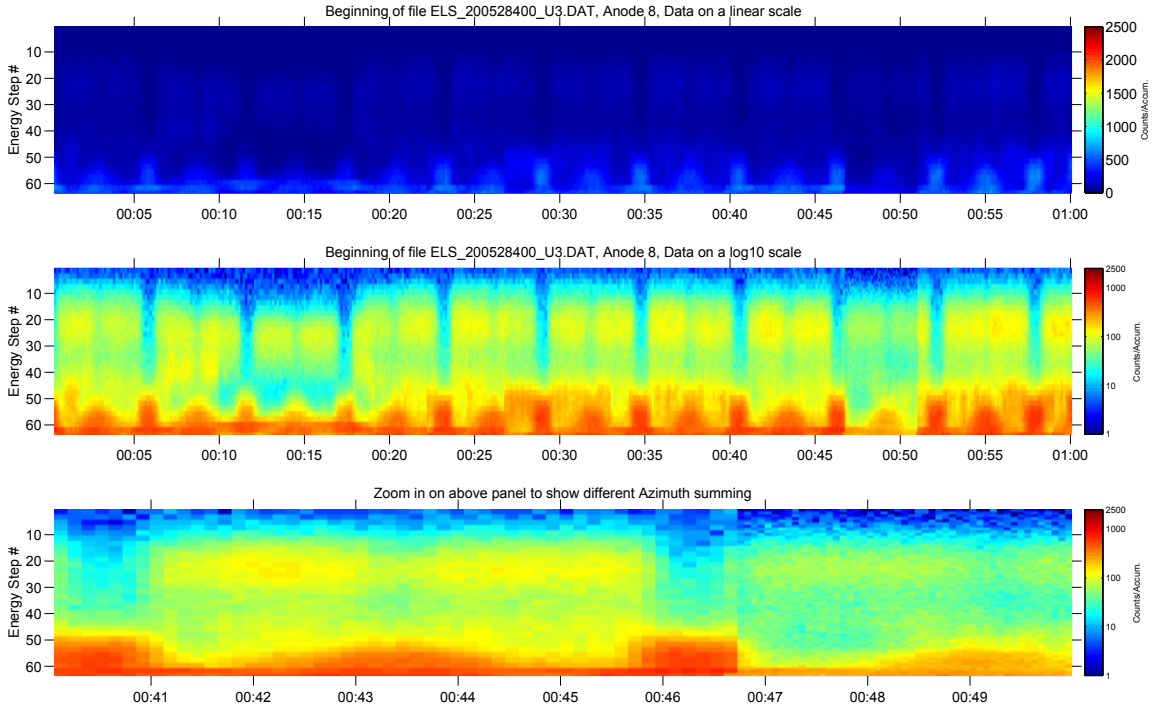

Figure 4.4: Example ELS spectrograms.

The next example (Figure 4.5) is of the ION data for the first hour of this DAYQ, and again anode 8 as anode 8 had the most counts. The top, middle and bottom panels are for SAM_ION_NUMBERS (explained in section 11.3) 01116, 13116 and 17116 respectively. All are shown on a linear colorbar as the counts are so low (typical to expect far fewer counts in ION data than SNG). ION data only comes down in higher telemetry modes, that is to say in the first hour of this day there are only the 8 A-cycles starting around 00:46:51.

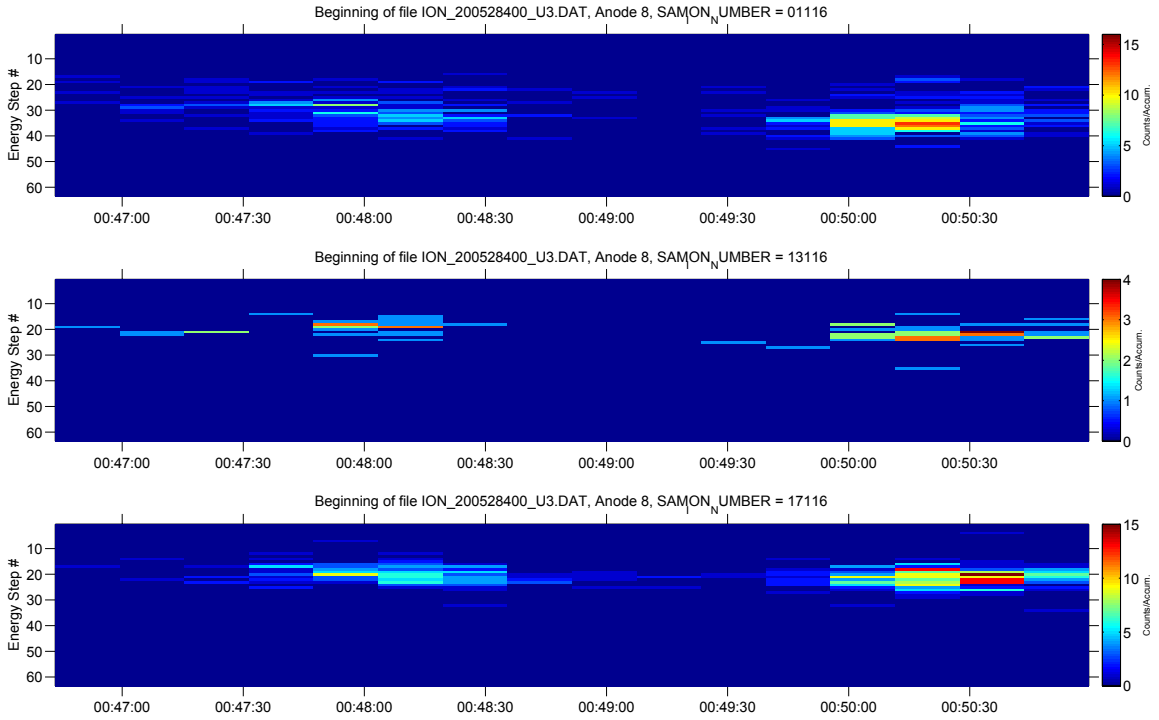

Figure 4.5: Example ION spectrograms.

To complete the examples of the first hour our sample day, Figure 4.6 shows example IBS spectrograms for anode 1, 2 and 3 respectively from file IBS_200528400_US.DAT, shown on a linear colors bar. The 'city-scape' effect truncating the lower parts of each column are due to the run-length encoding IBS uses and in the DATA object are MISSING_CONSTANT values.

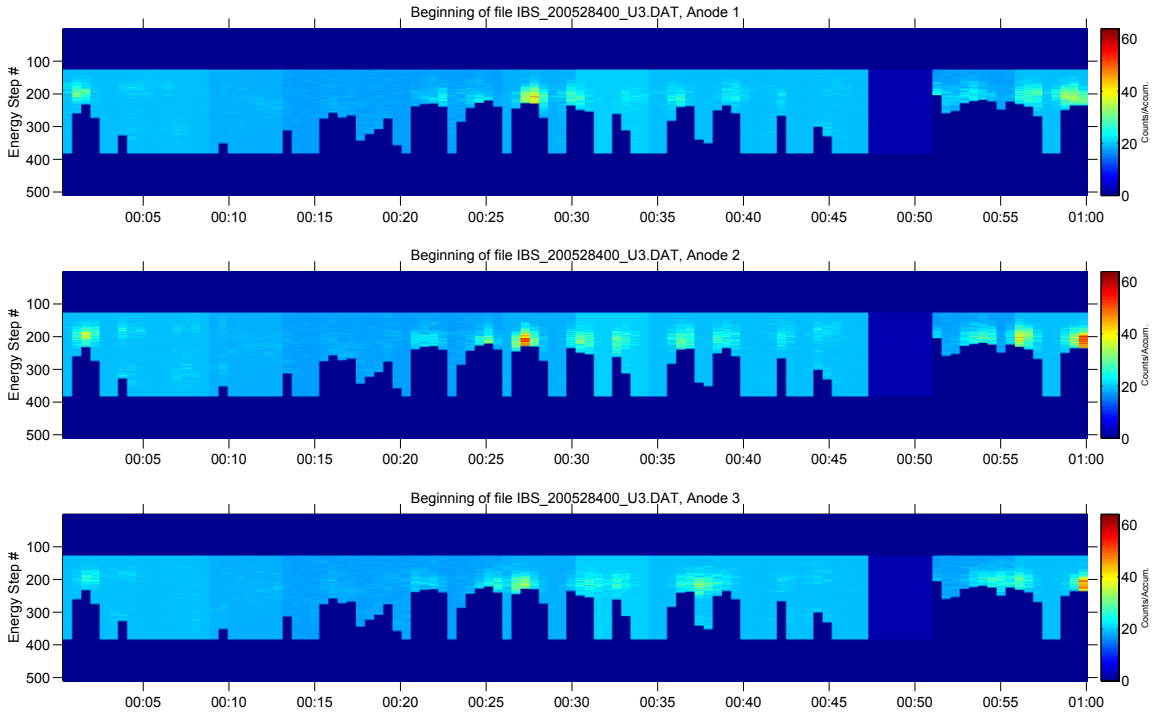

Figure 4.6: Example IBS spectrograms.

For information on converting energy step number to units of eV/q, see section 7.8.

These spectrograms were all done with the following RGB colormap of 64 different colors, where each R, G or B value is between 0 and 1:

Index	R	G	B
1	0	0	0.5625
2	0	0	0.625
3	0	0	0.6875
4	0	0	0.75
5	0	0	0.8125
6	0	0	0.875
7	0	0	0.9375
8	0	0	1
9	0	0.0625	1
10	0	0.125	1
11	0	0.1875	1
12	0	0.25	1
13	0	0.3125	1
14	0	0.375	1
15	0	0.4375	1
16	0	0.5	1
17	0	0.5625	1
18	0	0.625	1
19	0	0.6875	1
20	0	0.75	1
21	0	0.8125	1
22	0	0.875	1
23	0	0.9375	1
24	0	1	1
25	0.0625	1	0.9375
26	0.125	1	0.875
27	0.1875	1	0.8125
28	0.25	1	0.75
29	0.3125	1	0.6875
30	0.375	1	0.625
31	0.4375	1	0.5625
32	0.5	1	0.5

Index	R	G	B
33	0.5625	1	0.4375
34	0.625	1	0.375
35	0.6875	1	0.3125
36	0.75	1	0.25
37	0.8125	1	0.1875
38	0.875	1	0.125
39	0.9375	1	0.0625
40	1	1	0
41	1	0.9375	0
42	1	0.875	0
43	1	0.8125	0
44	1	0.75	0
45	1	0.6875	0
46	1	0.625	0
47	1	0.5625	0
48	1	0.5	0
49	1	0.4375	0
50	1	0.375	0
51	1	0.3125	0
52	1	0.25	0
53	1	0.1875	0
54	1	0.125	0
55	1	0.0625	0
56	1	0	0
57	0.9375	0	0
58	0.875	0	0
59	0.8125	0	0
60	0.75	0	0
61	0.6875	0	0
62	0.625	0	0
63	0.5625	0	0
64	0.5	0	0

4.12 How to plot TOF

The TOF data is substantially different in format to ELS, SNG or ION data, hence will be covered separately in the TOF chapter (10).

5 Using the Data – General Descriptions

Let's begin with the format (FMT) files as they describe the binary files that contain the CAPS data. Text in `courier new` font are direct copies from the respective PDS FMT files, and objects here are listed in the same order as they are found in those FMT files. The file formats for ELS, SNG, ION and IBS are very similar – deliberately so – to make it easy to move between datasets. Where an example description is given the SNG one is shown, unless it's different for all datasets in which case they are all shown.

TOF data are not described in this section as they have a very different format (see section 10), yet some of the values (like TIME, B_CYCLE_NUMBER) are identical.

5.1 What is a Record of Data

A record of data is merely how the data are packaged in the binary files. In the case of SNG, ION, ELS and IBS a record contains the counts for each anode at a particular energy step (or step-sum) and for a particular azimuth (or azimuth-sum), all listed as different 'objects' in the record.
For instance, if you used the above example codes to turn the binary data into a text file, each row is a different record.

The records are usually grouped and assembled into sweeps or azimuths before being plotted by the user.

5.2 Sweep versus Record versus Azimuth

A sweep of data is when the sensor goes from the highest energy (step 1) all the way to the lowest (step 63). This is known as 1 azimuth; for SNG data this takes 4-seconds, for ELS it takes 2-seconds. The sensor always works this way, however for telemetry saving reasons may group several sweeps together and download their sum, see Azimuth summing in section 7.3.1 for more info.

However the end user will have to group the records into sweeps and azimuths (or azimuth sums) themselves. For SNG, ION and ELS data, the start of the sweep begins with a record that has FIRST_ENERGY_STEP = 1, then contains the following records until one where LAST_ENERGY_STEP = 63 is met, which is included in the same sweep. Alternatively if you start each new sweep/azimuth with a record where FIRST_ENERGY_STEP = 1 and continue until the end of the file you will have the same result. The CAPS binary files order records with increasing energy step until it reaches the end and returns to 1. IBS may give subsets of energy steps, but always begins each sweep at the lowest energy step, and ends at the highest, as such this is

still easy to identify (if current record's energy step is lower that previous record's energy step a new sweep has begun).

It would be nice if we read in the data from a **SNG** binary file and out came a 3-D array of data of size (t, 8, 63) which would be (time, anode, energy step).
However, in the SNG, ELS and ION data each record contains data in a 1-D array of 8 values (for IBS this is a 1-D array of 3 values). That is each record corresponds to a particular energy step (or step sum) for a particular azimuth (or azimuth sum) and lists the counts in all the sensor's anodes. It is the job of the user to re-order these into sweeps in order to plot spectra.

5.3 Layout of a Binary File

The format of the ELS and SNG files are essentially identical – just one byte has a different meaning (ELS.COLLAPSE_FLAG and SNG.SPARE).

The IBS format is similar to that of SNG, with two exceptions – IBS.DATA has 3 values (1 per anode) whereas SNG.DATA has 8 values, and the OFFSET_TIME has the same meaning, but is 4-bytes for IBS instead of 2-bytes for SNG.

The ION format is very similar to SNG format, again with two exceptions – there is an extra object before the DATA object for ION data (SAM_ION_NUMBER) and the DATA values are signed integers whereas for SNG they are unsigned integers.

The objects are now described in detail, in the order they appear in the format files.

5.3.1 B_CYCLE_NUMBER

```
B cycle number from the start of the day, a value of 65535 indicates no
B-cycle data is available
```

The first B-cycle starting on a new day will be number 1, however the first records of the day may be fill values.
Do not use B_CYCLE_NUMBER to test for endianity issues, as 65535 a very common value. In binary that is 1111111111111111 and both little and big endian representations are the same – as such it's a poor test for endianity.

5.3.2 A_CYCLE_NUMBER

```
A cycle number from the start of day, a value of 65535 indicates that
no A-cycle header information is available
```

The first A-cycle starting on a new day will be number 1. A-cycle numbers are never fill values. If a data set has no use for A-cycle numbers (i.e. TOF) then they are simply not in the binary record at all (i.e. TOF binaries only list B-cycle numbers).

5.3.3 TIME

```
Start time of the A cycle, seconds from J2000 (barycentric dynamic
time).  An A-cycle is the 32 second instrument collection cycle.
```

This does not give you the time that the record is from, but the time of the start of the A-cycle the record falls within. See chapter 6 for how to turn this into UT, and also how to work out the time for a particular azimuth or record within the A-cycle. Note that these TIME values will increase or stay the same as you read down through a binary file. This provides a sanity check that you can include in your binary reading codes; the next TIME is never less than the previous TIME.

5.3.4 TELEMETRY_MODE

```
Logical telemetry rate and mode:
1   = 250 bps
2   = 500 bps
4   = 1 kbps
8   = 2 kbps
16  = 4 kbps
32  = 8 kbps
64  = 16 kbps
130 = 500 bps solar wind
132 = 1 kbps solar wind
136 = 2 kbps solar wind
```

The Telemetry Mode seems like a useful value, however it is of no help with using the data. This is because onboard software changes over the years mean that the same telemetry mode may sum azimuths or steps in different ways. In addition, even for the same telemetry mode and same onboard software version, SNG and ION data, which both come from IMS, can sum differently (see section 11.4.3.2).

It is better to read in the *FIRST_ENERGY_STEP*, *LAST_ENERGY_STEP*, *FIRST_AZIMUTH_VALUE* and *LAST_AZIMUTH_VALUE* objects for the given dataset to work out what averaging went on. See sections on accumulation time for SNG (8.4.1) and ELS (9.3.1), and also section 11.4.3.2 for differences for ION vs. SNG.

5.3.5 SPARE, COLLAPSE_FLAG or IBS_MODE_SUBMODE_STRATEGY

A 1-byte unsigned integer value follow the telemetry byte, but has different names and meanings in the different data sets as follows:

5.3.5.1 SPARE (SNG and ION data)

```
Contains zeroes
```

Does as it says, and is useless to the user. This is a one-byte spacer (each record is a single zero) so that the basic structure and size of an ELS record is the same as an

SNG record. It's worth adding a check in your code to make sure this value is zero, if not your binary reading script has a bug or the binary file has been corrupted.

The SPARE is a necessity of a PDS requirement that multi-byte values must start on an even byte in the file. As such the single byte TELEMETRY_MODE must be followed by a SPARE (another single byte) so that the two-byte OFFSET_TIME can follow on an even byte number in the file. (PDS also request that the whole record's length be an even number of bytes.)

5.3.5.2 COLLAPSE_FLAG (ELS data)

```
Flag indicating how data is collapsed:
0: average
1: sum
2: average with in-flight deadtime correction
3: sum with in-flight deadtime correction
4: snapshot portion
NOTE:  For snapshot, full collapse information is
gained by adding 4 (so snapshot portion can be
4, 5, 6, or 7 depending upon the collapse).
The upper bit will be set to 1 when
housekeeping is missing.
```

During the Saturn data this is usually 3, such that any dead time corrections have already been done for you. However if you are examining the cruise part of the mission you may need to add dead time corrections yourself depending on this flag.

5.3.5.3 IBS_MODE_SUBMODE_STRATEGY (IBS data)

```
IBS mode and submode flag:
0 - 6 Are DPU Acquire Normal
0 = Standard Sweep Collapse
1 = Standard Sweep Snapshot
2 = Solar Wind Search
3 = Solar Wind Track
4 = Magnetosphere Search
5 = Magnetosphere Survey
6 = Calibration Mode
16 - 22 Are DPU Acquire Solar Wind
16 = Standard Sweep Collapse
17 = Standard Sweep Snapshot
18 = Solar Wind Search
19 = Solar Wind Track
20 = Magnetosphere Search
21 = Magnetosphere Survey
22 = Calibration Mode
7,23-254 = spare
In the latest version of flight software, there are
small differences in the compression of these products
255 = Fill"
```

5.3.6 OFFSET_TIME

```
Milliseconds from start of A cycle
```

A potentially misleading variable that is not required. Since the *TIME* variable only gives you the time of the start of the current A-cycle, this is to aid you work out how many milliseconds into that A cycle that particular energy step started at, and at what time the record began, but gives you no information on how long the record lasts. The same information can be calculated from the *FIRST_AZIMUTH_VALUE* and *FIRST_ENERGY_VALUE*

For SNG and ION data this may be recalculated as follows:
OFFSET_TIME = (int)((32000/8) * (*FIRST_AZIMUTH_VALUE* – 1) +
(4000/64) * (*FIRST_ENERGY_VALUE* – 1))
[The (int) is to round the value down to the nearest whole number.]

The reason it is misleading follows. If there is no Azimuth summing then assigning this record a time in seconds of "*TIME* + *OFFSET_TIME*/1000" is fair enough, and has a duration of 62.5 ms (=4/64 s). But if all the azimuths in the A-cycle are summed then the start time may be the same, but the last measurement that was included in this record ended 28.0625 s later (i.e. the last Azimuth). If you consider a representative time of the record being the mean of the start of the first measurement and the end of the last, then in these two extremes they are 14 seconds apart – but you would have no idea of that if you were to just use the *OFFSET_TIME* variable.

For ELS the equation is:
OFFSET_TIME = (int)((32000/16) * (*FIRST_AZIMUTH_VALUE* – 1) +
(2000/64) * (*FIRST_ENERGY_VALUE* – 1))
[There is one rare exception to this, see section 9.1.6.]

IBS is a different case, in that it's not time since the current A-cycle started:

```
Milliseconds from start of the IBS collection cycle.
An IBS data product is constructed from 16 to 128
azimuths of data, with each azimuth representing 2
seconds of instrument data collection.
```

Due to this OFFSET_TIME in IBS is a four-byte unsigned integer, whereas for ELS/SNG/ION data its a two-byte unsigned integer so make sure your IBS code has this correctly as a four-byte value.

For IBS the equation is:
OFFSET_TIME = (int)((32000/16) * (*FIRST_AZIMUTH_VALUE* – 1) +
(2000/256) * (*FIRST_ENERGY_VALUE* – 1))

5.3.7 FIRST_ENERGY_STEP

`Minimum energy step in collapsed data`

The start energy step of the record.
May be a value from 1 to 63 only for SNG, ELS and ION
IBS may have value from 1 to 852.

If *FIRST_ENERGY_STEP* = *LAST_ENERGY_STEP* there is no bin summing.
These numbers may be the same, or "*LAST_ENERGY_STEP - FIRST_ENERGY_STEP"* may be equal to 1, 2 or 3 only (or 0 if they are the same).

5.3.8 LAST_ENERGY_STEP

`Maximum energy step in collapsed data`

The end energy step of the record. If any energy summing has occurred this will be different to the start energy.
See *FIRST_ENERGY_STEP* for conditions.
LAST_ENERGY_STEP must be the same or greater than *FIRST_ENERGY_STEP*.

5.3.9 FIRST_AZIMUTH_VALUE

`Minimum azimuth value in collapsed data`

The start azimuth of the record.
May be a value from 1 to 8 only for SNG and ION.
May be a value from 1 to 16 only for ELS.
May be a value from 1 to 128 only for IBS.

If *FIRST_AZIMUTH_VALUE* = *LAST_AZIMUTH_VALUE* there is no azimuth summing.
These numbers may be the same, or *LAST_AZIMUTH_VALUE* minus *FIRST_AZIMUTH_VALUE* may be equal to 0, 1, 3 or 7 only.
i.e. only these combinations are allowed for SNG or ION data:

FIRST_AZIMUTH_VALUE	*LAST_AZIMUTH_VALUE*
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
1	*2*
3	*4*
5	*6*
7	*8*
1	*4*
5	*8*
1	*8*

5.3.10 LAST_AZIMUTH_VALUE

```
Maximum azimuth value in collapsed data
```

The end azimuth of the record. If any azimuth summing has occurred this will be different to the start azimuth.
See *FIRST_AZIMUTH_VALUE* for conditions.
LAST_AZIMUTH_VALUE must be the same or greater than FIRST_AZIMUTH_VALUE.

5.3.11 SAM_ION_NUMBER – ION Data only!

This object breaks the symmetry of the formats and is only found in the ION data. SNG, ELS, and IBS data do not require it and therefore skip straight to the next object, Data.

```
SAM ion number (identifies ion and group table)
```

The meaning of this number is explained in detail in the ION section 11.3.

5.3.12 DATA

The previous objects all contained just one value, this one contains several – one count value for each anode during that record.

For SNG, ELS and ION data:

```
Counts in elevations 1 through 8
```

Which is the counts in anodes 1 to 8 respectively.

For IBS its similar:

```
Counts in fans 1 through 3
```

This time it's the counts in anodes 1, 2 and 3 respectively.

For SNG, ELS and IBS the counts are two-byte unsigned integers, i.e. 0 to 65535. However for ION data the counts are two-byte signed integers, with a minimum valid count of -32. If you copy code for SNG to ION, be sure to check the data type is now signed (not unsigned). The explanation for negative counts is given in the ION section 11.4.6.

5.4 Missing information

The binary files tell you almost everything you need to know in order to plot and use the raw data. However, there is no mention in the file about which Voltage Sweep table you should use to decode the data (see section 7.8.1).

For ELS data there is only 1 sweep table (*ELS_SWEEP_TABLE_ALL_VER*) so there is no question of which table to use.
However SNG (and therefore ION) and IBS both have several different sweep tables that could be used at any given time. This information is in the **ANC** binary file and the objects *IMS_SWEEP_TABLE_NUMBER* (SNG and ION) or *IBS_SWEEP_TABLE_NUMBER* (IBS), as such you should always read in the **ANC** binary file alongside the data file to check the data are being plotted at the correct energy/charge values. See the section on Sweep (Voltage) tables, section 7.8**.**

6 TIME, Barycentric to Universal Time (UT)

The PDS binary files all have times specified in seconds from epoch J2000 (defined as January 1, 2000, 12:00:00 Barycentric Dynamical Time). In each binary file it is listed as the TIME variable and refers to the start time of the A-cycle of that record. Adjacent records should occur at some multiple of 32 seconds apart, depending on data availability and data product.
(Possible exception if CAPS is turned off then on again, then A-cycles may not be a whole number of 32s apart. Also TOF data are at B-cycle resolution, so TIME is the start time of that B-cycle.)

In converting to UT time from seconds from J2000 (Barycentric dynamic time) leap seconds must be accounted for, of which there have been 5 (at the time of writing this document) since 1997 to 2012 inclusive. These occurred at 1997-Jun-30 23:50:60 (before Cassini launch), 1998-Dec-31 23:59:60, 2005-Dec-31 23:59:60, 2008-Dec-31 23:59:60 and 2012-Jun-30 23:59:60. (See http://en.wikipedia.org/wiki/Leap_seconds for more details.)

It is not immediately obvious how to put this into UT time and requires many small corrections. NAIF SPICE should be used for accuracy (see section 6.6), but a 'good enough' accuracy can be obtained more easily if you know when leap seconds have occurred. Both ways require you to keep up to date on knowledge of leap seconds to see if any new ones have been introduced within the time range of your data.

The following table gives the correct UT for specific dates and their seconds from J2000, corrected for leap seconds, and good to within 2 ms.

UT	Seconds from J2000 (Barycentric dynamic time)
1997-07-01 00:00:00.000	-79012736.816
1998-01-01 00:00:00.000	-63115136.816
1999-01-01 00:00:00.000	-31579135.816
2000-01-01 00:00:00.000	-43135.816
2000-01-01 11:58:55.816	0.000
2001-01-01 00:00:00.000	31579264.184
2002-01-01 00:00:00.000	63115264.184
2003-01-01 00:00:00.000	94651264.184
2004-01-01 00:00:00.000	126187264.184
2005-01-01 00:00:00.000	157809664.184
2006-01-01 00:00:00.000	189345665.184
2007-01-01 00:00:00.000	220881665.184
2008-01-01 00:00:00.000	252417665.184
2009-01-01 00:00:00.000	284040066.184
2010-01-01 00:00:00.000	315576066.184
2011-01-01 00:00:00.000	347112066.184
2012-01-01 00:00:00.000	378648066.184
2012-07-01 00:00:00.000	394372867.184

A quick coding method to convert seconds to UT in the correct date format for IDL/Matlab/Excel/OpenOffice/c/python is as follows. This assumes you know what year and DOY the binary file you are using is from, AND that a data record does not start on a leap second (nor midnight) itself.

Firstly use this pseudo-code to work out constant c:

```
if      (1999 <= YEAR & YEAR < 2006) { c =  0; }
else if (2006 <= YEAR & YEAR < 2009) { c = -1; }
else if (2009 <= YEAR & YEAR < 2012) { c = -2; }
else if (YEAR == 2012 & DOY  <  183) { c = -2; } # i.e. Jan-Jun
          # DOY 183 in 2012 (leap year) is 01 July, after leap second
else if (YEAR == 2012 & DOY  >= 183) { c = -3; } # i.e. Jul-Dec
          # <2013 as only know leap seconds up to end of 2012
          # currently (there could be another one 2012-Dec-31T23:59:60)
else if (               YEAR < 1999) { c = +1; }
          # Only valid for >= 1997-Jul-01, OK as Launch was 1997-Oct
else { print "outside range of known leap-seconds"; exit(1); }
          # Bail out if not in date range
```

Equivalently, you may use the above table to substitute in barycentric seconds for YEAR, that way you only use the contents of the binary files rather than relying on them being named correctly and/or having to read in the file name to get the year. This is also the safe way that would work if a leap second was introduced mid-year.

Let {Barycentric time in secs} = {secs} then the above loop becomes:

```
if      (-31579135.816  <= {secs} & {secs} < 189345665.184) { c =  0; }
else if (189345665.184  <= {secs} & {secs} < 284040066.184) { c = -1; }
else if (284040066.184  <= {secs} & {secs} < 394372867.184) { c = -2; }
else if (394372867.184  <= {secs} & {secs} < 410270467.184) { c = -3; }
else if (                           {secs} < -31579135.816) { c = +1; }
else { print "outside range of known leap-seconds"; exit(1); }
          # The long numbers are taken from the table above
```

The next line is then slightly different depending on if you are using IDL/Matlab/Excel/OpenOffice/c/python/etc., and will be broken down below:

6.1 IDL time formats

```
IDL Julian day     = 2451544.5D + ( {Barycentric time in secs} + ,$
                                          43135.816D + c )/86400D

Where 2451544.5D = julday(1,1,2000,0,0,0) & 43135.816 = 11h58m55.816s
```

Useful IDL time commands:

- caldat
- julday

Warnings:

- When using julday always include hour, min and sec!
  julday(1,1,2000,0,0,0) NE julday(1,1,2000)
- caldat keeps fractions of seconds (good!) but don't name your month & minute variables the same name!
- Ensure Julian day Number in caldat command is double precision.
- Julday has integer days starting at noon rather than midnight, hence julday of 2000-Jan-1 00:00:00 ends with .5

6.2 Matlab time formats

```
Matlab serial date =  730486    + ( {Barycentric time in secs} + ...
                                          43135.816  + c )/86400 ;

Where 730486 = datenum('01-Jan-2000') & 43135.816 = 11h58m55.816s
```

Useful Matlab time commands:

- datenum
- datestr
- datetick

Warnings:

- datestr rounds seconds down
- However datestr now has ability to show milliseconds (the FFF in the format)
  datestr(t,'yyyy-mm-ddTHH:MM:SS.FFF')
- Else fraction of second left = 86400 * (t -datenum(datestr(t)));
  where t = barycentric time

6.3 Spreadsheet time formats: Excel (Windows), Excel (Macs), OpenOffice (Windows/Macs/Linux):

```
Spreadsheet serial date, type in the cell (on the same line):
= datevalue("01-Jan-2000") + ({Barycentric time in secs} + 43135.816 +
c)/86400

Where 43135.816 = 11h58m55.816s
```

Useful Spreadsheet time commands:

- Change cell format between date/time & decimal as needed
- Useful Custom cell formats
  - yyyy-mmm-dd HH:MM:SS
  - yyyy-mm-dd HH:MM:SS
  - yyyy-mm-dd HH:MM:SS.000
    The .000 give decimal secs

Warnings:

- Date & Time cell formats round to nearest value of last unit in cell format
  i.e. if cell format ends SS then it rounds to nearest second, if MM then nearest minute, etc.
- Beware that spreadsheets calculate time in decimal days, but all tend to use a different epoch. To get around this use the **datevalue** command as above and you never have to worry. For reference:
  - Excel on Windows has a serial day of 1 = 1900-Jan-01 00:00:00
  - Excel on Macs has a serial day of 1 = 1904-Jan-02 00:00:00
  - Open Office has a serial day of 1 = 1899-Dec-31 00:00:00

6.4 Python time formats

Like c, Python also works in serial seconds (rather than days) and has a habit of trying to adjust the time for your local time zone that can lead to some confusion (or result in your times being several hours out). As such you should use 'naïve' date and time python objects (of which there are many), rather than 'aware' objects. See Python datetime documentation for more info and many more commands: http://docs.python.org/library/datetime.html

```
t = datetime(2000,1,1,11,58,55,816000) +
                  timedelta(seconds=({Barycentric time in secs} + c))
```

Useful Python date/time commands:

- datetime give it (year, month, day, hour, min, sec, microsecond)
- timedelta
- print(t).strftime('%Y-%m-%d %H:%M:%S.%f ') prints yyyy-mm-dd HH:MM:SS.ssssss

Warnings:

- Avoid daylight savings and time zone adjustments in your codes!

6.5 C code time formats

For c code, things are a bit different as it uses Unix (POSIX) time, which is a measure of seconds (rather than days) since 1970-Jan-01. Coding is also complicated by many c commands trying to adjust for daylight savings time or the time zone of your home country, neither of which we want.

For the c code to work, you must include the time header file:

```
#include <time.h>
```

Let BT= and c as before, with both BT and c being doubles, then:

```
char     str_time[20]; /* 19 chars in string yyyy-mm-ddTHH:MM:SS */
                       /* +1 for the terminating null            */
struct tm   *tmp_time;
time_t   tmp_t, Pt;

Pt = (time_t)(10957*86400 + 43135.816 + Bt  + c);
/* Pt = Seconds since Unix epoch time
   10957 days from 1970-Jan-01 (epoch) to 2000-Jan-01
   then 43135.816 is HH:MM:SS.sss
 */

/* Now to convert to a human readable string */
tmp_time =gmtime( &Pt );
strftime( str_time, 20, "%Y-%m-%dT%H:%M:%S", tmp_time);
/* Note, this rounds DOWN to whole seconds only. */
printf("%s\n",str_time); /* And print to screen… */
```

Useful c time commands:

- mktime
- gmtime
- strftime

Warnings:

- Avoid daylight savings adjustments (keep structure tm_isdst = 0) and routines that convert to your local time zone.

6.6 NAIF SPICE time conversions (accurate!)

To do this you need to download the SPICE toolkit for your programming platform of choice, and also the relevant SPICE kernels. This is a whole other data product and language in its own right – and most people find it confusing to begin with. It is beyond the scope of this document but we'll point you in the right direction.

For info on SPICE, the toolkits for different programming languages and to download the kernels go to: http://naif.jpl.nasa.gov/

ALWAYS ensure you have the latest kernels and especially the latest leap seconds file. At time of writing this is *naif0010.tls* (or *naif0010.tls.pc* if you use Windows), but that number may increase. SPICE will not auto-update these for you, so it is very easy to use old (and therefore incorrect) kernels accidentally.

The following is a snippet of basic SPICE code using MICE (the MATLAB SPICE toolkit), however the commands have the same names and inputs/outputs across the different computer languages, just in a slightly different order.

First set up SPICE and load in the leap second kernel:

```
% Clear Kernels to avoid accidentally using the wrong ones later
cspice_kclear
% Load in Leap Second kernel file - remember to update this!!
cspice_furnsh('path/to/kernel/naif0010.tls')
```

Then to do the conversion

```
% Convert Barycentric time to UTC (example is 2010-06-30T06:40:00.000)
Bary_time = 331152066.1839; % Barycentric seconds (TIME variable)
UTC_time_str = cspice_et2utc(Bary_time, 'ISOC', 3);
% ISOC is ISO Calendar format, UTC,
% the 3 is precision, so 3 = millisecond
fprintf('%s\n', UTC_time_str)
```

This will write out "2010-06-30T06:40:00.000".
If you have many times to convert, then in Matlab and IDL you can greatly increase the speed by vectorizing your code – see the SPICE help file for the et2utc command.

If you wish to go in the opposite direction:

```
% Convert UTC to Barycentric time
UTC_time_str = '2010-06-30T06:40:00.000';
et = cspice_str2et( UTC_time_str );
fprintf('%9.3f\n', et); % print to screen to millisecond precision.
```

This will print out "331152066.184". As before if you have many times to convert vectorize your Matlab/IDL code, see the str2et help file.

6.7 Finding a record's start and end time

This chapter has dealt with converting the TIME object from the binary files into something more useful for coding or UTC. But generally the TIME value was just the time at the start of that A-cycle, and not the time of that record.

Assuming we have already converted that TIME object from Barycentric time to a serial number of days, t, then the start time of a record is given by:

For SNG and ION:
T_{Start} = t + (*FIRST_AZIMUTH_VALUE* – 1 + (*FIRST_ENERGY_VALUE* – 1)/64)*4/86400

For ELS:
T_{Start} = t + (*FIRST_AZIMUTH_VALUE* – 1 + (*FIRST_ENERGY_VALUE* – 1)/64)*2/86400

The end time of the record can be calculated similarly:
For SNG and ION:

T_{End} = t + (*LAST_AZIMUTH_VALUE* – 1 + (*LAST_ENERGY_VALUE*)/64)*4/86400

For ELS:
T_{End} = t + (*LAST_AZIMUTH_VALUE* – 1 + (*LAST_ENERGY_VALUE*)/64)*2/86400

The duration of that record (in days) is simple if there has been no azimuth summing (i.e. *FIRST_AZIMUTH_VALUE* = *LAST_AZIMUTH_VALUE*), and is merely T_{End} - T_{Start}.
For SNG or ION:
T_{Dur} = (*LAST_AZIMUTH_VALUE* + (*LAST_ENERGY_VALUE*)/64)*4/86400 –
(*FIRST_AZIMUTH_VALUE* + (*FIRST_ENERGY_VALUE* – 1)/64)*4/86400
For ELS:
T_{Dur} = (*LAST_AZIMUTH_VALUE* + (*LAST_ENERGY_VALUE*)/64)*2/86400 –
(*FIRST_AZIMUTH_VALUE* + (*FIRST_ENERGY_VALUE* – 1)/64)*2/86400

However this duration is not the same as accumulation time and care must be taken. The duration is really from the start time of the first energy step to the last energy step, then waiting (but still counting) for the sensor to finish the sweep and start the next sweep, to return to the first energy step again and last energy step. As such the duration here includes the time at all other energy steps too, as such this duration should not be used to calculate accumulation times.
This is best explained with an example, consider an SNG A-cycle with no summing at all, first and last azimuths and energy steps are the same (all set to 1). Then in units of days:
T_{Dur} = (1 + 1/64)*4/86400 – (1)*4/86400 = (4/64)/86400
Or (4/64) seconds.

Now consider an A-cycle where the energy steps stay the same (first and last at 1) but last azimuth is 8 and first azimuth is 1. This is just 8 azimuths summed, so will T_{Dur} be 8 times larger at 0.5 seconds??

The answer is no:
T_{Dur} = (8 + 1/64)*4/86400 – (1)*4/86400 = (28+4/64)/86400
Rather than half a second it's now just over 28 seconds.

As such this method of working out duration of a record should NEVER be used to work out accumulation times. See the individual chapters (8-12) for each sensor for more information on accumulation times.

7 Instrument Dataset Knowledge

7.1 "Top Hat" Selects The Energy/Charge Of Particles To Be Analyzed.

The CAPS ELS and IMS sensors are electrostatic analyzers (ESA) at the entrance to their particular instrument to select a particular energy per charge of ion (positive or negative, including electrons) to continue to the sensor to count them. The symmetric shape of the ESA resembles that of a top hat, hence the name. For more detail on the instrument design see the CAPS instrument paper, although a very brief description follows. The instrument paper reference is:

Young, D.T. *et al.* (2004), Cassini Plasma Spectrometer Investigation, *Space Science Reviews*, 114, 1-112. doi: 10.1007/s11214-004-1406-4

Plasma enters the instrument through the parallel section at the top – the dimensions of which dictate the acceptance field of view at a given time, i.e. the longer the parallel section the narrower incident angle of incoming ions has to be in order to reach the center without being lost to the sides. At the middle of the top hat the ion enters two concentric hemispheres at different voltages. This sets up an electric field between the hemispheres and bends the trajectory of the ion. If the voltage on the hemispheres is just right it'll bend the ion's (or electron's) path so it can curve the right amount to exit the bottom of the analyzer without hitting the side and being lost. As such, by varying the voltage you can select out the ions (or electrons) with the energy/charge you desire and reject the rest (they hit the sides of the hemisphere and are lost).

For CAPS, the voltages start high and step down logarithmically in value, pausing at each step to accumulate data. More information on that is below, including how to account for settling time of this voltage.

7.2 Azimuth and Azimuth Sums

An azimuth is usually a term reserved for a single sweep through the voltages on a spinning spacecraft. It has a field of view that is defined in the spacecraft polar coordinate (0 to 180°) by the anode position, and the spacecraft azimuth coordinate (0 to 360°) by the angle the spacecraft has spun in the time interval of that sweep.

Cassini is a 3-Axis stabilized spacecraft that does not spin; hence 'azimuth' is not immediately an obvious term. In the case of CAPS, an azimuth refers to a particular voltage sweep of the top hat analyzer from high to low voltages. The term azimuth for Cassini CAPS does not imply the spacecraft is rolling, or the actuator is on – it is just a term for one sweep of data.

For instance, one SNG azimuth occurs every 4 seconds, one ELS azimuth is every 2 seconds, and one IBS azimuth is every 2 seconds. However Cassini can only downlink so much data to Earth, dependent on telemetry mode, so in lower telemetry modes it is common to sum azimuths together onboard Cassini, then downlink the sum. When azimuths are summed it is by a power of 2, i.e. for SNG data it is 1 (no summing), 2 (pairs), 4 (quads), or 8 (octs). For ELS* or IBS it could be 1, 2, 4, 8 or 16. In both cases the maximum summation is for 1 A-cycle (since a single ELS or IBS azimuth is half the duration of a single SNG azimuth).

[* ELS prior to arrival at Saturn (pre-2004, flight software 1 & 2 only) has a mode that groups an A-cycle into 3 uneven sums of 5, 5 and 6 azimuths. See section 9.1.5.]

An azimuth number in a data record (generally FIRST_AZIMUTH_VALUE or LAST_AZIMUTH_VALUE) refers to which azimuth within an A-cycle is being discussed. i.e. if FIRST_AZIMUTH_VALUE = 3 and LAST_AZIMUTH_VALUE = 3 then no summing has occurred, if FIRST_AZIMUTH_VALUE = 5 and LAST_AZIMUTH_VALUE = 8 then 4 azimuths have been summed, etc.

7.3 Data Summing

Usually this refers to Azimuth summing, but could also be anode summing or energy summing and any combination thereof. The reasons for it are the usual… SNG measures an Azimuth every 4s, ELS & IBS every 2s, but there is not enough space in the telemetry stream to downlink all of the data obtained in all telemetry modes, therefore some compression is done. This reduces the number of values to be downlinked, but when used loses pointing information on exactly which the data came from for azimuth or anode (polar) summations, and loses E/q resolution for energy summing.

7.3.1 Azimuth Summing

Azimuth only summing is the most common method used. For instance with SNG data, rather than having 8 records of 1 azimuth each per A-cycle, you may find 4 records of summed azimuth-pairs (1-2, 3-4, 5-6, 7-8), 2 records of summed azimuth quads (1-4, 5-8) or 1 record of azimuths 1-8 summed per A-cycle.

7.3.2 Energy Summing

Energy summing usually sums neighboring pairs of energy steps. This is common in ELS data, but rare in SNG data. Note that SNG and ELS have 63 energy steps at full energy resolution, therefore 32 energy steps when energy summing occurs - meaning that 1 of those 32 energy steps is a singular (unpaired) bin. This is important as the accumulation time for that 1 unpaired bin is half that of the other 31 energy steps… care must be taken when converting to scientific units to account for this – as this means that the accumulation time per record can vary for records within the same sweep. i.e. for a data record, if FIRST_ENERGY_STEP = 34 and LAST_ENERGY_STEP = 34, then clearly no summing has happened, if FIRST_ENERGY_STEP = 21 and LAST_ENERGY_STEP = 22 then there has been data summing of pairs.

If the energies were summed, which is very common for ELS data, then there are 32 energy steps that are step-pairs over 63 original steps… therefore 31 steps really are step pairs (LAST_ENERGY_STEP - FIRST_ENERGY_STEP = 1), and one step 'pair' is really just the remaining bin (LAST_ENERGY_STEP - FIRST_ENERGY_STEP = 0). As such that 1 record in a sweep of 'pairs' has a different dt_{Summed} compared to the other 31 step-pairs in that sweep (which are obviously twice as long).

Energy summing of SNG data is very rare and has an added complication, see section 8.6 for details.

For this reason you should use the equation for dt_{Summed} given in the ELS, SNG, ION sections for absolutely each and every record separately – do not assume it will remain the same for a whole sweep or azimuth(-sum).

7.3.3 Anode Summing

Occasionally neighboring anodes are summed to anode-pairs, however this only occurs in SNG data, never in ELS nor IBS. ION data is not summed at Saturn (2004 and later), however some cruise data from 1999-2003 may have anode summing (see section 11.4.3.3). Even for SNG data it only occurs in TELEMETRY_MODE 132. See section 8.1.2 for further details of how it affects SNG data.

7.3.4 Data sanity checks on summing

Within your code you can run some simple checks that the values in your data make sense and are not giving errors. For instance these four rules must be obeyed:

FIRST_AZIMUTH_VALUE <= LAST_AZIMUTH_VALUE
FIRST_ENERGY_STEP <= LAST_ENERGY STEP

For SNG, ION, ELS data:

1 <= FIRST_ENERGY_STEP <= 63
1 <= LAST_ENERGY_STEP <= 63

For SNG and ION:

1 <= FIRST_AZIMUTH_VALUE <= 8
1 <= LAST_AZIMUTH_VALUE <= 8

For ELS and IBS:

1 <= FIRST_AZIMUTH_VALUE <= 16
1 <= LAST_AZIMUTH_VALUE <= 16

Recalling that azimuth summing only sums to powers of 2, (i.e. 1, 2, 4, 8…)
Then for SNG and ION data there are only 4 allowed values for difference between the azimuth values:

LAST_AZIMUTH_VALUE - FIRST_AZIMUTH_VALUE = 0, 1, 3, 7

Any other value must be wrong, which is what you'd expect if you were to consider all allowed variations of summing as shown in the following table.

FIRST_AZ	LAST_AZ	Diff.
1	1	0
2	2	0
3	3	0
4	4	0
5	5	0
6	6	0
7	7	0
8	8	0

FIRST_AZ	LAST_AZ	Diff.
1	2	1
3	4	1
5	6	1
7	8	1
1	4	3
5	8	3
1	8	7

For ELS we can also sum 16 azimuths, so these 5 values are allowed:

LAST_AZIMUTH_VALUE - FIRST_AZIMUTH_VALUE = 0, 1, 3, 7, 15

(However in pre-Saturn data (before 2004) with flight software 1 or 2 you may also have 4 or 5 as a valid difference – see the ELS section 9.1.5 for more details of this exception.)

Other simple checking rules like this can be constructed by looking at the VALID_MAXIMUM and VALID_MINIMUM values in the LBL files for each object.

7.4 A-cycles and B-cycles

An A-cycle is the basic interval of time for the CAPS instruments and lasts 32 seconds. For SNG a complete voltage sweep lasts 4 seconds, as such there are 8 voltage sweeps per A-cycle. ELS has a 2 second sweep, and therefore 16 voltage sweeps per A-cycle. Each A-cycle is numbered sequentially, with the counter reset at each day to begin again at 1. This does not necessarily happen at the first A-cycle after midnight, but rather when the last B-cycle that began on the previous day has finished. Alternatively if the instrument is off as we cross midnight, A-cycle number 1 is the first one when the instrument is turned on, which may be much later in the day. i.e. if the 3rd binary file of the day (the 12h-18h one) has an A-cycle equal to 1, then there can not be any data prior to that, and therefore the first two binary files of the day (0h-6h and 6h-12h) should not exist.

If there is a telemetry mode change, voltage table change, or mcp level change; it will only happen instantaneously at the start of an A-cycle, never within the A-cycle.

A-cycles are the base units for ACT, ANC, ELS, IBS, ION and SNG datasets, although each A-cycle contains multiple azimuths.

Every SNG and ELS data record must have a non-fill A-cycle number and can range from 1 to 2732. (2700 A-cycles in a 24-hour period, and if a C-cycle (a defunct CAPS unit for science that's still in the onboard flight code) just starts a fraction of a second before midnight it could continue for up to another 32 A-cycles.)

A B-cycle can have one of 3 different durations, usually 256 seconds but also 512 or 1024 seconds (8, 16 or 32 A-cycles respectively), dependent on telemetry modes. However only a subset of the daily A-cycles are also included in B-cycles, which is dependent on the telemetry mode of the time. B-cycles are also counted sequentially with the number resetting daily. B-cycle number 1 is the first B-cycle that day that started after midnight. B-cycles are used for Time-Of-Flight (TOF) measurements and are only useful with regard to SNG data for cross-referencing with TOF data, although the *TIME* variable is just as useful for that (and both may have a slight offset - see section 10.15.4 for more details on TOF and SNG/ION comparisons). There can never be more than 342 B-cycles in a day (2732/8) although it's extremely unlikely numbers will ever get this high. Also note that B-cycle duration can be confused when telemetry mode changes during a B-cycle (see section 10.15.3).

ION data are only returned near (and overlapping) a B-cycle. B-cycles happen simultaneously with A-cycles, but only occur in higher telemetry modes.

Since not every A-cycle is part of a B-cycle, you will often see the B-cycle number of a record being the fill value, 65535.

The DAYQ file will not end when time crosses 0 (midnight), 6, 12 or 18 hours, but only after the B-cycle it had started prior to then has finished.
i.e. the first record (A-cycle #1) of file SNG_200528400_U3.DAT is time stamped at 2005-284T00:00:19. If you really want the first 18 seconds of that day you need the end of the previous file SNG_200528318_U3.DAT, whose last A-cycle begins at 2005-283T23:59:47 and extends to 2005-284T00:00:19.
This overlap is true for all midnight boundaries when CAPS is on, and also for 06h, 12h and 18h boundaries.

7.5 Identifying the Flight Software Version

You may hear that some data features are only present in the flight software version 1, or a data issue was fixed by flight software version 4. They are referring to the software the CAPS instrument on Cassini was running, of which there have been 3 versions so far: 1, 2, and 4.

For any given A-cycle, the flight software version can be found from the ANC binaries that have 4 objects of use:
FSW_MAJOR_VERSION
FSW_SUBMAJOR_VERSION
FSW_MINOR_VERSION
FSW_SUBMINOR_VERSION

The description in the FMT files to these 4 objects is the same:

```
Flight software sub-minor version number.
To build the full flight software version:
Major.SubMajor.Minor.SubMinor
For example: 3.1.0.2
```

Generally people only speak of the major version (ANC.FSW_MAJOR_VERSION).

7.6 File Durations and Missing Files

Each DAYQ file can last up to 6 hours plus 32 A-cycles (see section 7.4) so will always have a duration of <= 06:17:04.
However they can contain far less, perhaps just a few A-cycles worth. Just because a DAYQ file exists does not mean there is 6 hours of data, nor does it mean there are 6 hours of continuous data. The best you can say without loading in the DAT file is by reading the LBL file which gives you the start time of the first record in the file, and the start time of the last record in the file.

For instance, file the LBL file SNG_200528400_U3.LBL has the lines:

```
START_TIME                    = 2005-284T00:00:19
STOP_TIME                     = 2005-284T05:59:47
```

As such the first record of the file started at 19s past midnight, and the last record of the file started at 05:59:47, however A-cycles are 32s long, so the last record of file ended at 06:00:19

While this DAYQ file may span 6 hours, it does not mean there are not data gaps within that time interval… the only way to be sure is to load in the DAT file and examine the time stamp of each record.

In this case the file does span exactly 6 hours of data, but it does not begin at midnight as the DAYQ number might have implied.

TOF data does not have A-cycle objects, hence the STOP_TIME in the LBL file is the start time of the final B-cycle. Since B-cycles can have different durations you require more information to figure out when the data actually stopped, see TOF chapter 10.

If a DAYQ has zero records then there is simply no data in that ~6 hour period. As such no binary files are produced. Therefore if you want SNG data for DAYQ 200518700 you would try to find file SNG_200518700_U3.DAT. If the file does not exist (nor SNG_200518700_U3.LBL) then either:

a) There was zero SNG data on that day so no files exist
   hence a file of that name is not in the PDS

Or

b) You've accidentally deleted your local copy of the file and need to re-download it from the PDS.

Unfortunately there is no easy way to distinguish the two possibilities. In this case of DAYQ 200518700, CAPS was turned off so there really is no data and no files (however by the end of this day CAPS was turned back on and data collection resumed).

7.7 Count quantization for SNG, ION, ELS and TOF

SNG Counts are measured during an accumulation period, then if any azimuth or energy summing occurs they are added together. This gives an integer number of counts from 0 to 65534 for each energy bin (or bin-sum) within a record, i.e. an unsigned two-byte word. Since downlinking bandwidth is limited this two-byte word is compressed to a one-byte word. In effect the data for the record (after any summing) is quantized to one of 256 values. For the SNG binary file on the ground

these numbers have been decompressed back to a two-byte word and range from 0 to 65535 again, however there are now only 256 unique values there. Whenever numbers were decompressed on the ground the bottom value of the range is given, not the center of the range.

The compression scheme used is linear at low counts, then logarithmic at higher counts. For (SNG) counts of 0 to 63 there is no change, these are still values 0 to 63 in the one-byte word. After that the logarithmic compression is applied, where numbers are rounded down, by a range of ~1.5%. The penultimate compressed value (254) is 26642 decompressed, but really represents any number from 26642 to 27498. The final compressed value (255) is 27499 decompressed – however represents all numbers larger than itself – although generally SNGs rarely measures such high counts in a record. This means that the uncertainty of a measurement is no longer the expected square root of the counts, but sqrt($N + a^2N^2$) where a is 0.0153 for SNG.

ELS has a similar two-byte to one-byte compression scheme that uses a look-up table, however the linear region is for counts 0 to 41, then logarithmic compression up to the final compressed value of 255 representing the values 65504 and above. Again the numbers are rounded down, by a range of ~1.7% (slightly larger than that for SNG due to the ELS measuring up to 65504). The same two-byte to one-byte compression scheme is also used by IBS.

The TOF data are compressed from 4-bytes to 2-bytes, however it is unlikely that there were enough counts measured outside of the 1:1 linear regime of compression, so it is often thought off as not having any quantization.

ION (and LOG) data are compressed to one-byte in a similar manner to SNG but with different parameters.

IBS data uses different compression methods (such as run length encoding), which also results in quantizing and truncating the data values (see IBS chapter 12). The IBS 2-byte to 1-byte table quantization table is identical to that used for ELS.

The following two pages list the 1-byte to 2-byte decompression tables used for ELS, IBS, SNG and ION, along with their respective fill values (giving 257 unique numbers). Note than the ION values do contain negative numbers (see 0), whereas the others are always positive (or zero).

1-Byte value	ELS & IBS	SNG	ION
Fill Value	65535	65535	28671
0	0	0	-32
1	1	1	-31
2	2	2	-30
3	3	3	-29
4	4	4	-28
5	5	5	-27
6	6	6	-26
7	7	7	-25
8	8	8	-24
9	9	9	-23
10	10	10	-22
11	11	11	-21
12	12	12	-20
13	13	13	-19
14	14	14	-18
15	15	15	-17
16	16	16	-16
17	17	17	-15
18	18	18	-14
19	19	19	-13
20	20	20	-12
21	21	21	-11
22	22	22	-10
23	23	23	-9
24	24	24	-8
25	25	25	-7
26	26	26	-6
27	27	27	-5
28	28	28	-4
29	29	29	-3
30	30	30	-2
31	31	31	-1
32	32	32	0
33	33	33	1
34	34	34	2
35	35	35	3
36	36	36	4
37	37	37	5
38	38	38	6
39	39	39	7
40	40	40	8
41	41	41	9
42	43	42	10
43	45	43	11
44	47	44	12
45	49	45	13
46	51	46	14
47	53	47	15
48	55	48	16
49	57	49	17
50	59	50	18
51	61	51	19
52	63	52	20
53	65	53	21
54	67	54	22
55	69	55	23
56	71	56	24
57	74	57	25
58	77	58	26
59	80	59	27
60	83	60	28
61	86	61	29
62	89	62	30
63	92	63	31
64	95	65	32
65	98	67	33
66	101	69	34
67	105	71	35
68	109	73	36
69	113	76	38
70	117	78	39
71	121	81	41
72	125	83	42
73	129	86	44
74	133	89	45
75	138	92	47
76	143	95	49
77	148	98	50
78	153	101	52
79	158	104	54
80	164	107	56
81	170	111	58
82	176	114	60
83	182	118	62
84	188	122	65
85	194	126	67
86	201	130	69
87	208	134	72
88	215	139	75
89	223	143	77
90	231	148	80
91	239	152	83
92	247	157	86
93	256	162	89
94	265	168	92
95	274	173	96
96	283	179	99
97	293	184	103
98	303	190	106
99	314	196	110
100	325	203	114
101	336	209	118
102	348	216	123
103	360	223	127
104	372	230	132
105	385	238	137
106	398	245	141
107	412	253	147
108	427	261	152
109	442	270	157
110	457	278	163
111	473	287	169
112	489	297	175
113	506	306	181
114	524	316	188
115	542	326	195
116	561	337	202
117	580	348	209
118	600	359	217
119	621	370	224
120	643	382	232
121	665	395	241
122	688	407	250
123	712	421	259
124	737	434	268
125	763	448	278
126	789	463	288
127	816	477	298

1-Byte value	ELS & IBS	SNG	ION
128	844	493	309
129	873	509	320
130	903	525	331
131	934	542	343
132	966	559	356
133	1000	577	368
134	1035	596	382
135	1071	615	396
136	1108	635	410
137	1147	655	425
138	1187	676	440
139	1228	698	456
140	1271	721	472
141	1315	744	489
142	1360	768	507
143	1407	793	525
144	1456	818	544
145	1507	844	564
146	1559	872	584
147	1613	900	605
148	1669	929	627
149	1727	959	649
150	1787	989	673
151	1849	1021	697
152	1913	1054	722
153	1979	1088	748
154	2048	1123	775
155	2112	1159	803
156	2176	1196	832
157	2240	1235	862
158	2304	1275	893
159	2400	1316	925
160	2496	1358	959
161	2592	1402	993
162	2688	1447	1029
163	2784	1493	1066
164	2880	1542	1104
165	2976	1591	1144
166	3072	1642	1186
167	3168	1695	1228
168	3296	1750	1272
169	3424	1806	1318
170	3552	1864	1366
171	3680	1924	1415
172	3808	1986	1466
173	3936	2050	1519
174	4064	2116	1574
175	4192	2184	1630
176	4352	2254	1689
177	4512	2327	1750
178	4672	2402	1813
179	4832	2479	1878
180	4992	2559	1946
181	5152	2641	2016
182	5344	2726	2089
183	5536	2814	2164
184	5728	2904	2242
185	5920	2998	2323
186	6112	3094	2406
187	6336	3193	2493
188	6560	3296	2583
189	6784	3402	2676
190	7008	3512	2772
191	7264	3625	2872

1-Byte value	ELS & IBS	SNG	ION
192	7520	3741	2975
193	7776	3862	3083
194	8032	3986	3194
195	8320	4114	3309
196	8608	4247	3428
197	8896	4383	3551
198	9216	4524	3679
199	9536	4670	3812
200	9856	4820	3949
201	10208	4975	4091
202	10560	5135	4239
203	10944	5300	4391
204	11328	5471	4550
205	11744	5647	4713
206	12160	5828	4883
207	12576	6016	5059
208	13024	6209	5241
209	13472	6409	5430
210	13952	6615	5626
211	14432	6828	5828
212	14944	7048	6038
213	15456	7275	6256
214	16000	7509	6481
215	16576	7750	6714
216	17152	7999	6956
217	17760	8257	7207
218	18400	8522	7466
219	19040	8796	7735
220	19712	9079	8014
221	20416	9372	8303
222	21120	9673	8602
223	21856	9984	8911
224	22624	10305	9232
225	23424	10637	9565
226	24224	10979	9909
227	25056	11332	10266
228	25920	11697	10636
229	26816	12073	11019
230	27744	12461	11416
231	28704	12862	11827
232	29696	13276	12253
233	30752	13703	12695
234	31840	14144	13152
235	32960	14599	13625
236	34112	15068	14116
237	35296	15553	14625
238	36544	16053	15151
239	37824	16570	15697
240	39136	17103	16262
241	40512	17653	16848
242	41920	18221	17455
243	43392	18807	18084
244	44896	19412	18735
245	46464	20036	19410
246	48096	20681	20109
247	49760	21346	20833
248	51488	22033	21583
249	53280	22742	22361
250	55136	23473	23166
251	57056	24228	24000
252	59040	25008	24865
253	61120	25812	25760
254	63264	26642	26688
255	65504	27499	27649

7.8 Voltage Sweeps and Energy tables

All the voltage sweep tables for all sensors are at a fixed voltage for each data accumulation period, start high and step down in value logarithmically.

7.8.1 SWEEP TABLES

These files can be downloaded from the PDS, under the CALIB directory of the PDS collection COCAPS_1SAT. The following table lists their filename and denotes which provide a voltage value, an energy value, or both:

File	Step Number	Voltage (V)	Energy (eV)
ELS_SWEEP_TABLE_ALL_VER	☑	☑	☑
IBS_SWEEP_V0_V1_V2	☑	☑	☒
IBS_SWEEP_V3	☑	☒	☑
IMS_SWEEP_TABLE_0_Std	☑	☑	☒
IMS_SWEEP_TABLE_0_Titans	☑	☑	☒
IMS_SWEEP_TABLE_1_Std	☑	☑	☒
IMS_SWEEP_TABLE_2_Std	☑	☑	☒
IMS_SWEEP_TABLE_15_Titans	☑	☑	☒
IMS_SWEEP_TABLE_16_Calib	☑	☑	☒
IMS_SWEEP_TABLE_17_Titans	☑	☑	☒
IMS_SWEEP_TABLE_255_Titans	☑	☑	☒

Clearly the sweep tables may have a column for just voltage, just eV, or both. If just voltage (the majority) then you'll need to know the k factor to convert to eV.

Note that sweep tables do not change during A-cycles, if they change they do so instantaneously at the start of the A-cycle.

Always use the ANC files to look up the correct sweep table you should be using for IMS and IBS, which are ANC objects IMS_SWEEP_TABLE_NUMBER and IBS_SWEEP_TABLE_NUMBER. There is no ELS table number information in the ANC files, as there is and will ever be just the one energy table.

There is some room for confusion when IMS_SWEEP_TABLE_NUMBER is equal to 0, which of the two tables listed in the table should be used (Std or Titans)?

If IMS Table is 0 and the year is < 2004 use IMS_SWEEP_TABLE_0_Std,
but if the year is 2005 or later then use IMS_SWEEP_TABLE_0_Titans.

The actual time ranges are given in the corresponding LBL files, and neither one covers 2004 at all. See section 8.8 for more detail.

Some basic notes on date ranges follow, but use the version given by IMS_SWEEP_TABLE_NUMBER.

The ELS table has never changed – it's the only one, valid for all times.

IMS Standard Tables
Table 0 was used from 1999-001T00:00:00 to 2003-236T00:00:00
Table 1 was used from 2003-236T00:00:00 to 2007-162T03:23:15 (2007-June-11)
Table 2 was used from 2007-162T03:23:15 (2007-June-11) to present

Difference between Table 1 to table 0 is that the 63rd bin is set to 0 in Table 1.
Difference between Table 2 to tables 1, 1st and 2nd bin are set to the value of the 3rd bin.

IMS Titan tables:
Table 0 was used for some Titan passes between 2005-022T10:00:00 and 2007-162T03:23:15
Table 15 was used for some Titan passes before 2007-162T03:23:15 (2007-June-11).
[These date ranges come from the PDS Label files for the tables, but only one of them is used per encounter, which IMS_SWEEP_TABLE_NUMBER tells you.]
Table 17 used for Titan passed after 2007-162T03:23:15.
Table 255 was only used once due to an error (the step number were offset, i.e. table 15 step 2 appeared at step 1 in table 255).

IMS table 16 is only used for calibration, and is radically different to all the others.

IBS_SWEEP_V0_V1_V2 is used up to 2003-236T00:00:00, afterwards IBS_SWEEP_V3

7.8.2 The k Factor: Voltage Step to Energy Step

These are essentially the same item with different units. Voltage Step 1 is the same as Energy Step 1, however the voltage will be measured in volts and refers to the voltage over the hemispheres in the top hat (see *Young et al.* (2004) instrument paper), while the energy is in eV of plasma ions/electrons selected by that voltage to pass through the top hat and be counted. The relation is simply

$$E = kV$$

where k is just a multiplier. Normally plots are shown with an axis either in eV or step number (voltage or energy).

For SNG, ION, TOF and LOG data, k = 6.25 eV/V.
(Reference: CAPS team meeting #41 by F. Crary)

For IBS – one file is in eV (so already ready for you) the other in V, and k = 19.0 eV/V (according to the IBS_SWEEP_V0_V1_V2 LBL file).

ELS k = about 6.2 eV/V.

Since the ELS sweep table contains both V and eV you can work out k, however it is not the exact same value throughout. As such it is recommended just to use the eV values provided. i.e. Lewis *et al.* [2010] assumes k = 6.2 but used the eV data.

In terms of plotting, it is usually easier and faster to plot against energy step number (1 = highest eV, 63 = lowest), as these are equally spaced numbers and as such equal sized pixels in the plot. Plotting in units of eV/q is more correct and more useful scientifically, but the logarithmically spaced values mean that each pixel is now a different height which makes the coding more complex, and also longer to execute.

7.8.3 Center vs. Upper/Lower Energies

The Sweep Tables listed above all provide the information needed to generate the center energy of each bin (either it's already in eV or you multiply the V by the k-factor). This is because the electrostatic analyzer sits at a fixed voltage during each accumulation period. For scientific unit conversions we tend to use these center energies to represent each energy bin.

Line plots tend to use the center energy value, 63 count values plotted on the vertical axis need 63 energy values plotted on the horizontal axis. But to plot a spectrogram each of those counts needs to be a pixel with a width and a height in order to color a rectangle appropriately. As such for the energy axis, center energies are of no use and you need an upper and lower energy value for each energy step.

The ΔE/E (units of percent) of the sensor then tells you how wide the range of energies of incoming ions that will be accepted and measured.
The Upper and Lower limits are then calculated from the center energy:

$$E(i)_{Upper} = \left(1 + \frac{[\Delta E/E]}{2}\right) E(i)_{Center}$$

$$E(i)_{Lower} = \left(1 - \frac{[\Delta E/E]}{2}\right) E(i)_{Center}$$

If the ΔE/E is 17% then that means that energies within 8.5% (half above and half below the center energy) would also successfully pass through the top hat to be counted. Therefore if the center energy of a given step is E, the upper energy of said step would be 1.085*E and the lower energy would be 0.915*E.

You might also need to know the upper and lower energies of each step if your analysis includes integration of phase space, for instance. However it is more commonly required to plot spectrograms with an accurate energy scale on the vertical axis (be it either a log or linear plot).

Requiring upper and lower energies (of particle transmission at full width half maximum) of each bin for plotting purposes poses a new issue. While the ΔE/E of a sensor is a fixed value for all energies and that the sweep tables are set up to separate the voltage steps on a log scale at about a distance of ΔE/E apart from each other, it's not perfect. If plotted as is strictly correct you would end up with horizontal blank gaps between some energy steps, and with other energy steps slightly overlapping each other. Ideally you want the lower edge of energy step 5, for example, to be identical to the upper edge of energy step 6, but while they will be close they will never be exactly the same value. However it is safe to fiddle the values for the purpose of plotting spectrograms.

The following example is for SNGs data (with 63 steps per sweep), but applies to plotting any sensor that should have continuous but non-overlapping energy bins. In general, if *i* is the step number (1 (highest) to 63 (lowest)), and we have the 63 center energies we can calculate E_{Upper_plot} and E_{Lower_plot} from the above E_{Upper} and E_{Lower}, via the following steps.

Step 1: The plotted upper energy of the highest energy bin (*i*=1) is the same as the actual upper energy based on ΔE/E. That is to say:

$$E(i=1)_{Upper\_plot} = E(i=1)_{Upper}$$

Step 2: Now fill in E_{Upper_plot} for *i*=2 to 63 (avoid *i*=1 that we set in step 1). This is merely the average of the lower energy of the energy step above, and the upper energy of the current energy step:

$$E(i)_{Upper\_plot} = \frac{E(i-1)_{Lower} + E(i)_{Upper}}{2}, \quad for \quad i > 1$$

Step 3: Now fill in the missing E_{Lower_plot} for *i*=1 to 62 (avoid *i*=63) by using E_{Upper_plot} from step 2.

$$E(i)_{Lower\_plot} = E(i+1)_{Upper\_plot}, \quad for \quad i < 63$$

Step 4: All that's left to calculate is the lowest plot energy of the lowest bin (*i*=63). This is merely the actual lower energy:

$$E(i=63)_{Lower\_plot} = E(i=63)_{Lower}$$

Step 5: If $E(i=63)_{Center} > 0$ then skip this step and you are finished.
If the lowest step is zero eV however this means that the final energy step should not measure any real counts and merely measures the background penetrating radiation. A solution is to alter 3 values as follows:

$$E(i=62)_{Lower\_plot} = E(i=62)_{Lower}$$
$$E(i=63)_{Upper\_plot} = E(i=62)_{Lower}$$
$$E(i=63)_{Lower\_plot} = 0$$

This is purely for plotting purposes so you can still see the counts in the lowest step bin but remember it's just a background measure.
(If you wish to plot your energy scale on a log axis you may wish to replace 0 with a non-zero tiny number, perhaps 0.001 eV.)

7.8.4 The Fly-back

This is something that few people worry about. The electrostatic analyzers step down from a high voltage to a low voltage during a sweep. At the end of the sweep, before the next sweep can start, it has to ramp the voltage back up to the highest level again from the low value. This takes a little time and is known as the fly-back, as the voltage is 'flying back' to the high level so it's ready for the next sweep.

For SNG and ELS data there 63 energy steps and the 64th would be the fly-back. However as no data is taken during the fly-back there is never a step 64 in the data files. Yet this non-listed step 64 does take the same amount of time to complete as any of the other individual steps.

i.e. for SNG data, one sweep takes 4 seconds for 63 steps, but the time per step is 4/64 seconds as the 64th is the fly-back. If you find an A-cycle that has no summing (in any of azimuth, polar or energy) you can note the OFFSET_TIME value in the data file will increase in intervals of 62.5 milliseconds (4-seconds/64, except the object is whole numbers so values increase by 62 ms, then 63, then 62, then 63, etc.) all the way up to the end of a sweep, then there is a 125 ms jump to the next record… purely because the fly back step is not included in the data reported.

So if you observer such a jump in the OFFSET_TIME (unrelated to data summing) between the end of the last sweep and start of the next, this is due to the unlisted fly-back. (See OFFSET_TIME section 5.3.6).

7.8.5 High to low energies

Remember that for all CAPS sensors, the voltage sweep tables list Energy Step 1 as the highest energy, and subsequent steps go down to lower energies. Also remember that steps (and anodes) always start counting at 1, never 0.

(By comparison, Galileo's PLS instrument is the opposite, Energies start low and go high, and the lowest energy is step 0 not 1.)

7.9 Settling time vs. accumulation time

For each voltage sweep, each sensor starts at a high voltage and steps down to lower voltages measuring counts as it goes. However the sensor is not always accumulating counts.
For each step the voltage over the electrostatic analyzer has to drop from the last step to the new value of the current step, and then sit at that fixed voltage. At the very beginning of the step the voltage drops until it reaches that fixed voltage, but

that takes a bit of time – and you do not want to be measuring counts during that interval as the actual voltage is unknown and changing.

Therefore at the start of each step is a settling time to allow the voltage to reach its destination and settle to a fixed value. During this time no data is taken.
After the settling time (now the voltage is at its fixed value) data is taken for the rest of the time left in that step, this is know as the accumulation time. So each energy step begins with a settling time followed by an accumulation time.

For IMS datasets (SNG/ION), the settling time is one eighth of the step.
So each step lasts 62.5 ms [=4s/64],
the settling time is 7.8125 ms [= (4s/64)*(1/8)],
leaving the accumulation time as 54.6875 ms [= (4s/64)*(7/8)].
(TOF and LOG also have the $1/8^{th}$ settling time.)

For ELS data, the settling time is one quarter of a step, but in absolute terms is the exact same time as for IMS:
So each step lasts 31.25 ms [=2s/64],
the settling time is 7.8125 ms [= (2s/64)*(1/4)],
leaving the accumulation time as 23.4375 ms [= (2s/64)*(3/4)].

For IBS data, the settling time is one eighth of the step.
So each step lasts 7.8125 ms [=2s/256],
the settling time is 0.977 ms [= (2s/256)*(1/8)],
leaving the accumulation time as 6.836 ms [= (2s/256)*(7/8)].

Note that despite the settling time being the same (absolute duration) for ELS and SNG, the accumulation time for ELS is not simply half that of the accumulation time for SNG.

'Settling time' is not the same as 'dead time', for information on dead time see section 14.3.

When working with data however there is often azimuth, energy or anode summing going on – meaning the total counts reported were taken over a longer accumulation period. i.e. if there was no energy nor anode summing, but 4 azimuths were summed together, then the total accumulation time is 4 times that of a single accumulation period. If there was also energy summing to energy-step-pairs, then it would be 8 times, etc.

7.10 Correcting For Cross Talk

Cross Talk is when ions or electrons that are incident on one anode end up being counted in a different anode (between exiting the top hat analyzer and the anodes a few veer off course to a different anode). It has the effect of smearing out the data over the anodes. It could also be electronic (as is the case for IBS) where a current pulse from a count in one anode induces a detectable current in an adjacent wire. Therefore you want to 'sharpen' the image by removing the cross talk effect from the measured data. Background due to penetrating radiation would only generate cross-talk electronically where as scattered particles can produce it either way; ideally the two should be accounted for separately but there is insufficient calibration to do so. Ideally we would remove electrical cross talk, then remove penetrating radiation ('background') from the data, then remove 'scattered' cross talk, however the following cross talk matrix is the best available and the exact order to apply it is uncertain.

The ion datasets are affected by cross talk: SNG, ION and IBS datasets.
TOF is not affected as there is only one stop anode to measure all ions – so it measures all incident ions.
ELS is not affected by cross talk. Strictly speaking ELS is, but it's less than a 3% issue, too small to bother with.

The figure below (Figure 7.1) graphically describes the cross talk effect. The main ion distribution is observed in 3 anodes with peak incident direction in anode 4, slightly closer to the anode 5 direction than anode 3, but within anode 4. The top panel shows what would ideally happen; the red ions are all those ions incident on anode 4 and measured in anode 4, likewise the blue on anode 5 and green on anode 3. The next row separates out where each of those incident ions actually ended up, split up by incident anode. The center anode shows just those counts originally incident on anode 4, and most of the counts are still measured in anode 4, however a significant proportion of counts ended up being measured in the neighboring anodes, and some in their neighbors. The signal has been smeared out over many anodes. However, it's not just the original anode 4 counts that got smeared out. To the left and right the anode 3 and anode 5 counts got smeared out also. The instrument cannot separate these out, so measures the sum of all, the third panel in the plot. The bottom panel shows what the instrument really sees, the total counts it measured after the individual anode smears.

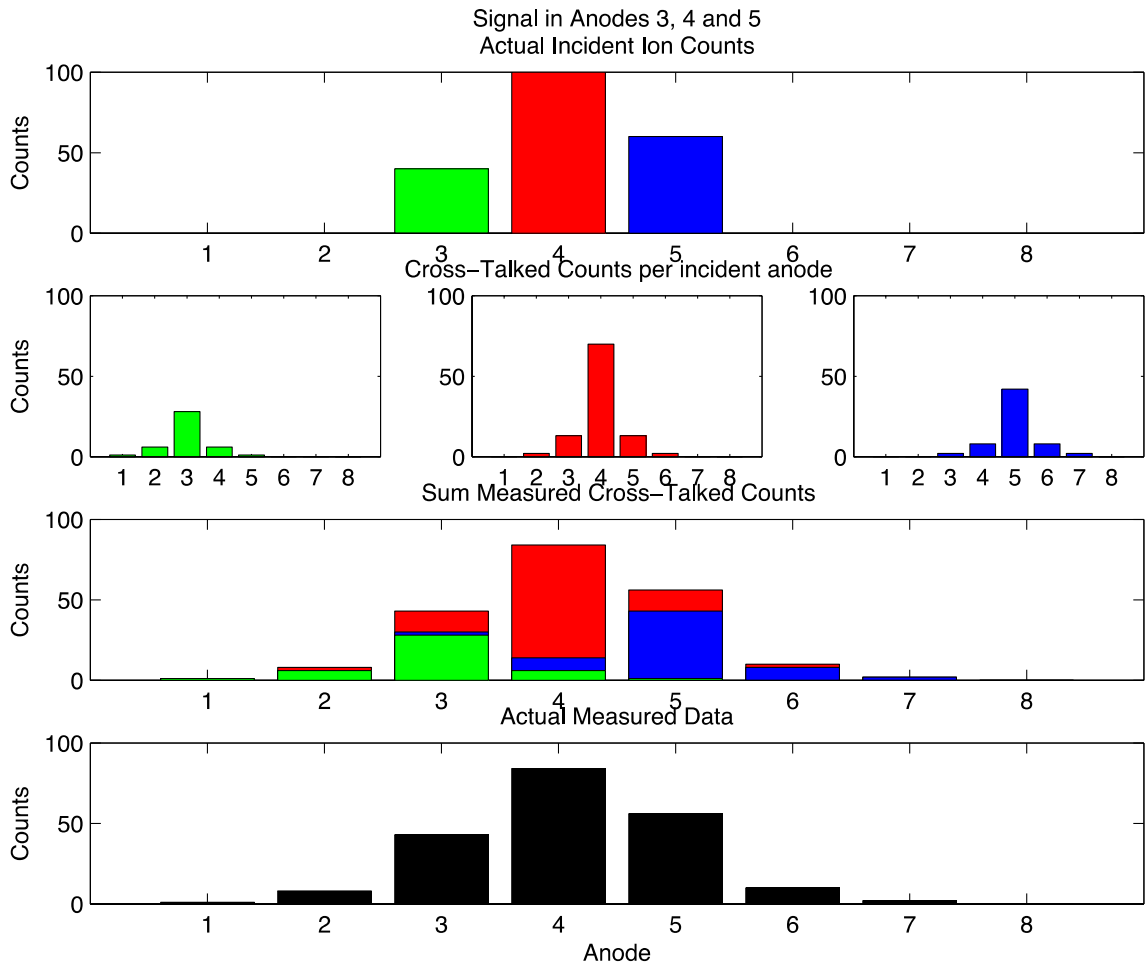

Figure 7.1: Describing Cross Talk.

So a true distribution of the top panel is measured as the bottom panel, wider and less intense.

As previously mentioned, ELS data are unaffected by cross-talk, but a identity matrix could be used if you wish. Although a lot of ELS data assumes isotopic distribution and uses anode 5 only (as it is the least obscured (by the spacecraft body)) anode.

7.10.1 SNG and ION data Cross talk

For ions that are aligned to impact on any given anode, some of them (≈1/4) are deflected into neighboring anodes. For ions that should arrive on anode i, the proportion that fall on node j can be calculated by:

Anode i-j	Proportion
0	0.748
1	0.101
2	0.015
3	0.007
4	0.004
5	0
6	0
7	0

It is to be assumed these figures are species independent and energy independent. (The actual matrix was calculated for protons and is the best and only data existing.)

Full details are given in the LANL report 'Numerical Moments Computation for CAPS/IMS' by *Thomsen and Delapp*, LA-UR-05-1542, which may be downloaded at http://nis-www.lanl.gov/nis-projects/caps/Moments_Computation.pdf

As such if you simulate data you must artificially add the cross-talk effect if you wish to compare your simulation with measured data. This is simply done by a matrix multiplication on data from all anodes of the same sweep and energy step, then adding on any background as required.

$$M = \alpha T + B$$

Where:
M is the measured data (for a particular energy/azimuth), a column vector of size 8.
B is the background data, a column vector of size 8
T is the 'correct' data, a column vector of size 8
And α is the cross-talk matrix, of size 8 by 8.

M, *B* and *T* are all column vectors of size 8, simple the respective counts in anodes 1 through 8 for the same energy step(-sum) and Azimuth(-sum).
Even if you'll ultimately only use 1 anode for data analysis, you still require all 8 anodes of data to correctly account for cross-talk.

The α matrix itself (sometimes known as the p-matrix) was calculated from in-flight calibration using solar wind protons, but should be used for all ion species. It is a symmetric down the diagonal and is shown below (not in an equation to make it easy to copy/paste).

α = [
```
0.748, 0.101, 0.015, 0.007, 0.004, 0.000, 0.000, 0.000
0.101, 0.748, 0.101, 0.015, 0.007, 0.004, 0.000, 0.000
0.015, 0.101, 0.748, 0.101, 0.015, 0.007, 0.004, 0.000
0.007, 0.015, 0.101, 0.748, 0.101, 0.015, 0.007, 0.004
0.004, 0.007, 0.015, 0.101, 0.748, 0.101, 0.015, 0.007
0.000, 0.004, 0.007, 0.015, 0.101, 0.748, 0.101, 0.015
0.000, 0.000, 0.004, 0.007, 0.015, 0.101, 0.748, 0.101
0.000, 0.000, 0.000, 0.004, 0.007, 0.015, 0.101, 0.748
```
]

However it is more common to have the measured data *M* and you wish to extract *T*. A rearrangement of the equation gives:

$$T = \alpha^{-1}\left(M - B\right)$$

where α^{-1} is the matrix inverse of α. To 4 decimal places, α^{-1} looks like:

α^{-1} = [
```
 1.3618, -0.1835, -0.0013, -0.0083, -0.0047,  0.0018, -0.0001,  0.0001
-0.1835,  1.3865, -0.1833, -0.0002, -0.0077, -0.0049,  0.0018, -0.0001
-0.0013, -0.1833,  1.3865, -0.1833, -0.0002, -0.0077, -0.0049,  0.0018
-0.0083, -0.0002, -0.1833,  1.3866, -0.1833, -0.0002, -0.0077, -0.0047
-0.0047, -0.0077, -0.0002, -0.1833,  1.3866, -0.1833, -0.0002, -0.0083
 0.0018, -0.0049, -0.0077, -0.0002, -0.1833,  1.3865, -0.1833, -0.0013
-0.0001,  0.0018, -0.0049, -0.0077, -0.0002, -0.1833,  1.3865, -0.1835
 0.0001, -0.0001,  0.0018, -0.0047, -0.0083, -0.0013, -0.1835,  1.3618
```
]

However rather than typing this in it is better to type in α and let your code do the inverse matrix for you. (IDL command *invert*, Matlab command *inv*.)

As an example, consider A-cycle 3 from DAYQ file 200528400 for SNG. The data are A-cycle resolution (Last azimuth = 8, first azimuth = 1) and energy step 20 has these counts for the 8 anodes: [50 78 73 95 196 525 900 959]^T, (where ^{T} means transpose vector), then after sharpening the data the *T* vector is:
[52.92 83.05 62.68 70.51 149.13 524.86 974.67 1138.18]^T
Note that *T* is no longer whole numbers, and shown here to two decimal places.

7.10.2 IBS data Cross talk

IBS does have a cross-talk effect too, but now the cross talk matrix is sized 3x3, as only 3 anode for IBS. The standard values used are given as the inverse matrix to start with, which is the one required to go from measured counts to corrected counts, using the same matrix formula as for SNG cross talk:

$$T = \alpha^{-1}\left(M - B\right)$$

where α^{-1} = [

```
 1.025750, -0.143420, -0.0385874
-0.135635,  1.026000, -0.0406647
-0.136521, -0.135645,  1.0121700
```

]

Should you want to know the α matrix, inverse α^{-1}.

Note how this matrix is not symmetric down the diagonal.

There is also a second caveat – this matrix was provided for counts below 300. If the counts are above 300 then the reliability is uncertain, however a more sophisticated cross talk matrix has yet to be developed.

7.10.3 ELS data Cross talk

ELS does not have cross talk issues. If you must use an α, make it an identity matrix.

Technically there is a cross talk effect, but it's less than 3%, as such insignificant.

7.11 How to recognize Magnetosphere, Magnetosheath and Solar Wind, intervals from the data

Solar Wind is easy to identify in the SNG data (assuming CAPS has a field of view that covers the solar wind direction) as the solar wind beam is so focused the H^+ ions only appears in 1 or 2 energy steps, far too few to get a science measurement from. The He^{++} solar wind component may be visible in just 1 energy step about 3-4 energy steps above the H^+ peak. However IMS was not designed for the solar wind; IBS with its superior energy resolution was and should be used for any solar wind analysis.

Figure 7.2 shows an example of Solar Wind in SNG data.

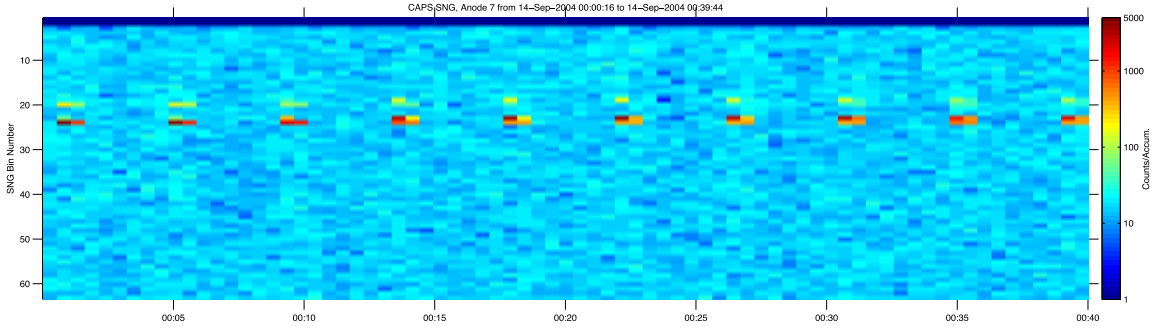

Figure 7.2: Example of solar wind in SNG spectrogram.

If you are in the solar wind you expect the solar wind ion beam to be very focused and in just one anode. But remember that with cross talk in SNG data you may see the above pattern in many anodes – in fact this was used to refine the cross talk matrix.

Magnetosphere and Magnetosheath data are harder to distinguish visually, and again both very dependent on having suitable field of view to capture the plasma in the SNG data. In general the magnetosheath data can be approximated by a Maxwellian distribution. This means that one anode had the peak counts, but the counts drop away quickly in neighboring anodes, such that the counts are quickly below 50% of those in the peak anode. Magnetosheath however has a very broad distribution, such that most (if not all) the anodes that have count rates are ~50% or more of the anode with the peak counts. This technique was used in *Wilson et al.* (2011) and shown in their figure 4, copied here as Figure 7.3:

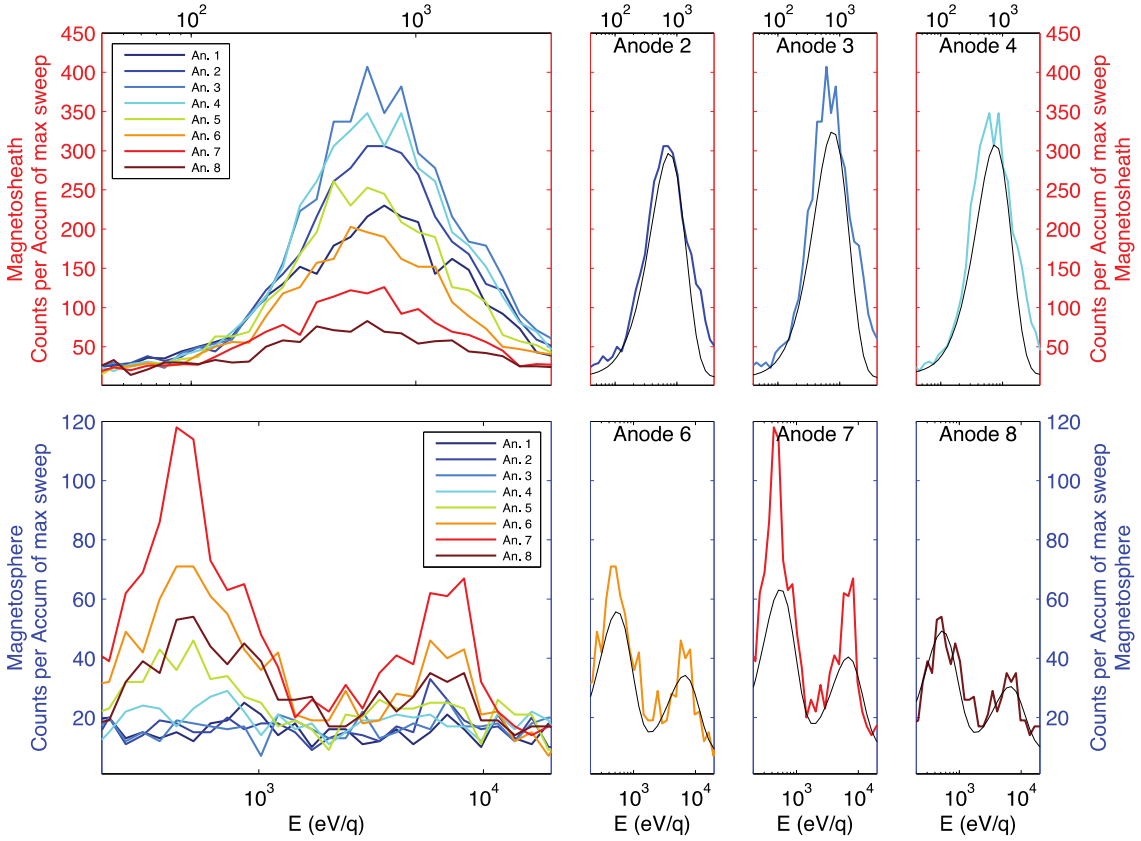

Figure 7.3: Example SNG line plots in the magnetosheath & magnetosphere, top and bottom respectively, for all 8 anodes. Taken from figure 4 of *Wilson et al.* (2011). The x-axis is in units of eV/q rather than energy step (1-63).

Note that you may have to identify calibration periods too or actuator homing runs so that you do not mistakenly use that data for science. Calibrations runs, known as mcp gain tests, are explained in chapter 17. Likewise, actuator homing runs are explained in section 13.3.5.

The following four plots show typical examples of ELS data in the solar wind, magnetosheath, magnetosphere and periapsis.

The first ELS example is typical solar wind data:

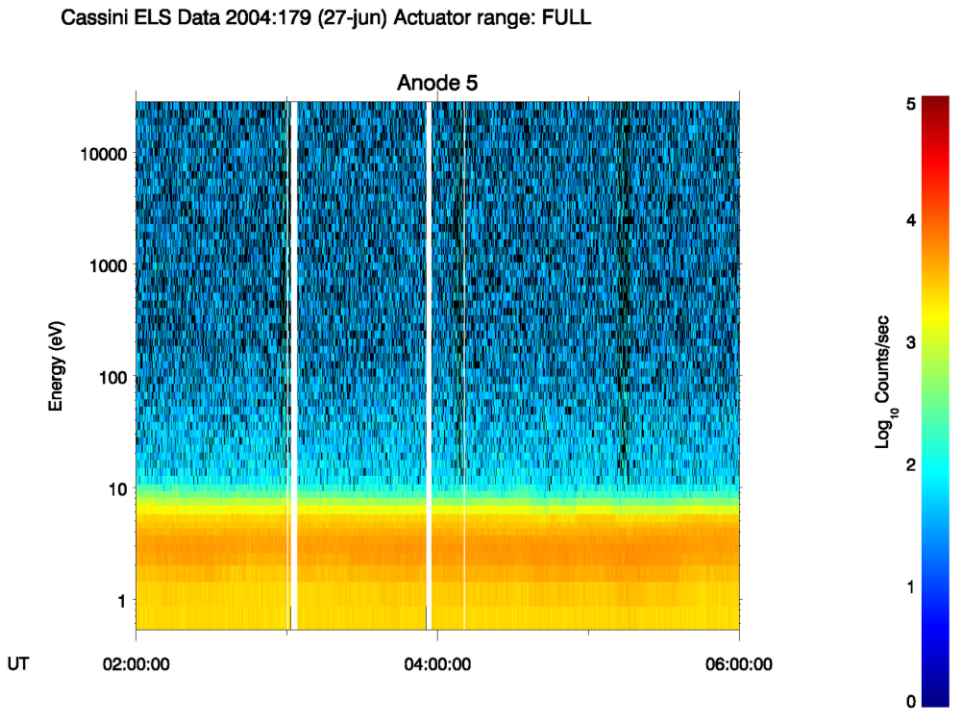

Figure 7.4: Example of solar win in ELS spectrogram.

Next is typical magnetosheath ELS data:

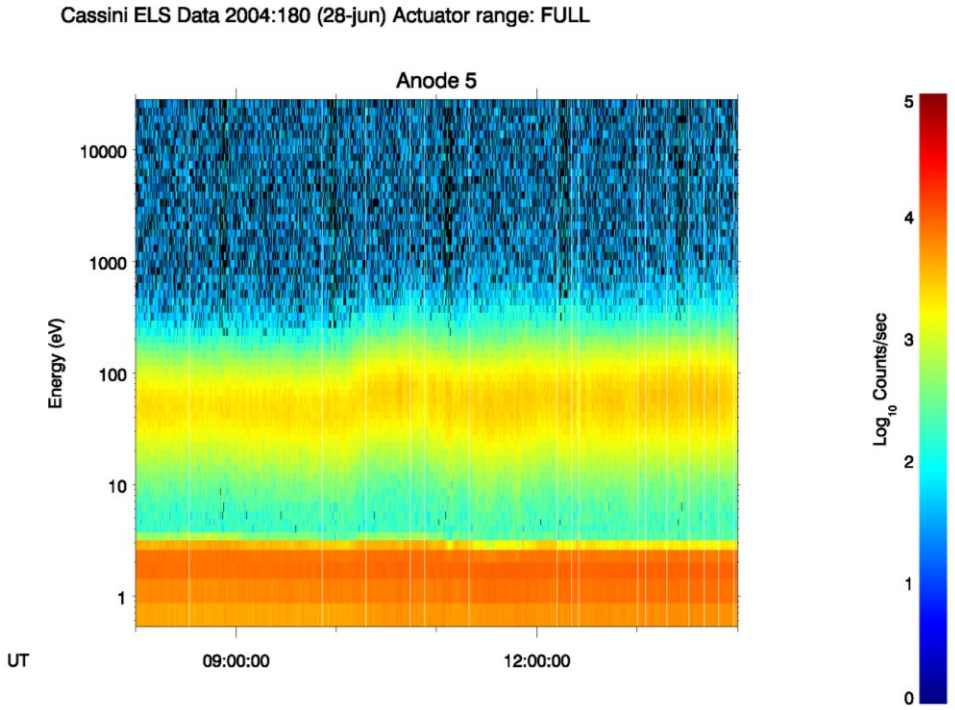

Figure 7.5: Example of magnetosheath in ELS spectrogram.

Typical magnetosphere ELS data:

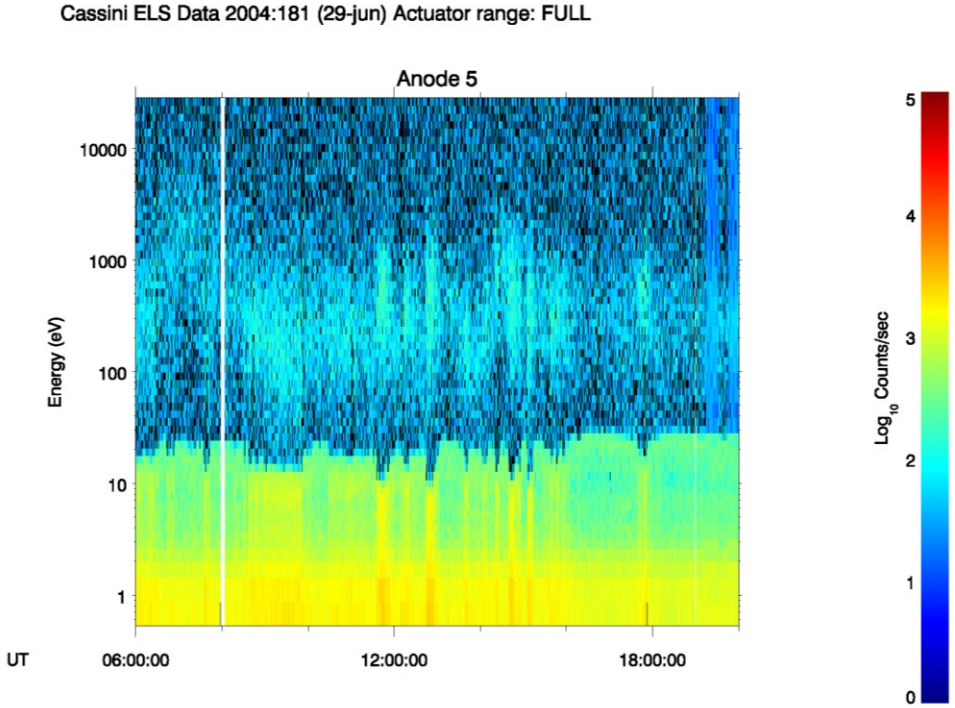

Figure 7.6: Example of magnetosphere in ELS spectrogram.

Then deep within the magnetosphere near periapsis the penetrating radiation is significant and visibly saturates the data at all energies:

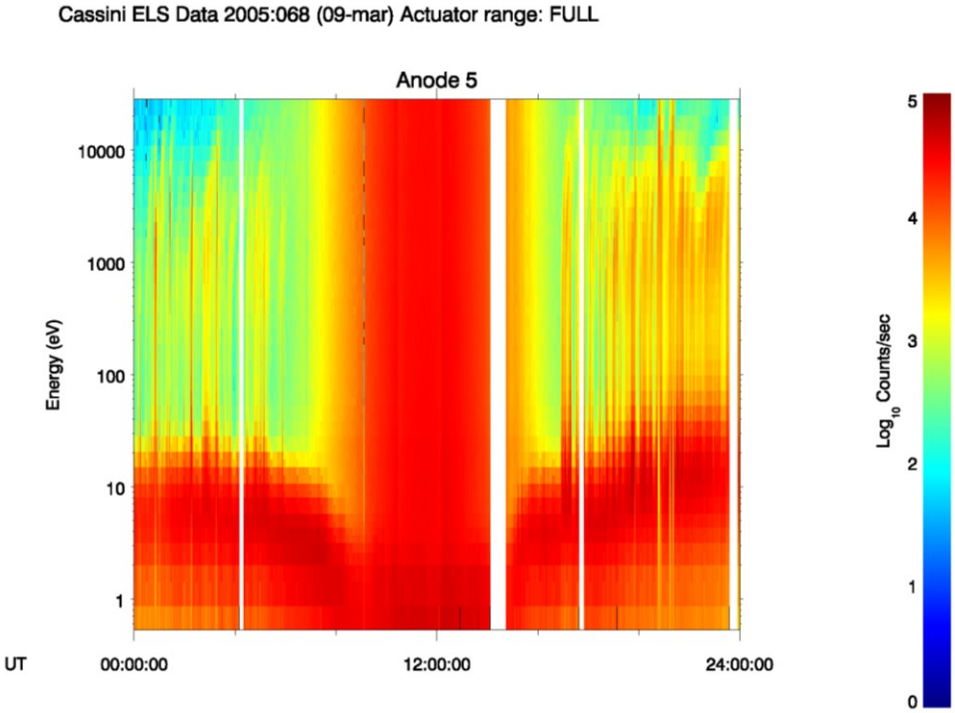

Figure 7.7: Example of penetrating radiation in ELS spectrogram.

See section 9.1.7 for more information on ELS penetrating radiation.

8 SNG Using the Data – General Knowledge

8.1 Things to know about the dataset

The dataset is pretty well behaved, but there are a few oddities to watch out for.

8.1.1 Bins 1 and 2 are bad, while bin 64 is useless (but good).

Bins 1 and 2 (the top two bins of the 63 IMS bins) are bad. Usually unreasonable low counts, and a sharp drop from bin 3. We don't know why, and by the third version of the default voltage sweep table (IMS_SWEEP_TABLE_2_V3.TAB) these top two bins are set to the same voltage as the 3rd bin. Basically do not use them.

Bin 64 is the fly-back bin always, and as such never contains any data and is never listed in the SNG binary files (there are no E-Steps with 64). It is usually ignored, and although there are really 64 bins you may see code written where it defines arrays of 63 bins.

8.1.2 TELEMETRY_MODE 132 sums anodes

SNG anode summing is rare, so much so that there are no data objects (i.e. first/last as with ENERGY_STEP or AZIMUTH_VALUE) that exist to identify it. This only occurs in SNG data when TELEMETERY_MODE = 132, and can also be identified from the DATA object. For SNG the DATA object is 8 values, one for each of the anodes. If the 1st, 3rd, 5th and 7th values are valid numbers but 2nd, 4th, 6th and 8th are the MISSING_CONSTANT value (65535) you know that anode summing has occurred. In this anodes 1 and 2 are summed and the total placed in the column for anode 1, while anode 2 is given the MISSING_CONSTANT value, etc.

In this SNG telemetry mode, used for solar wind, not only are the anodes summed to pairs, but also the azimuths and energies are also summed (Azimuths-Octs and Energy-step-pairs). However instances are found throughout the missing, for example DAYQ 201116112, and file SNG_201116112_U3.DAT.

8.1.3 Data corruptions

These are more prevalent in the early days of the mission, but are worth keeping an eye out for.

8.1.3.1 Fill Values

Remember to check if any of the data are a fill value, 65535 in the case of counts. These are unlikely, and usually only spotted in the two following examples of data corruptions. Unless you are summing over a large period of time you may not see them at all.

8.1.3.2 Slippage

'Slippage' is so called because if you look at a spectrogram of the data it appears as if one column has slipped down in e/q for two adjacent records. Many examples can be seen during SOI. The first azimuth appears to have slipped down. The next azimuth appears to be similarly slipped down, except that the lower energy half is full of fill values. During SOI (with data summed to azimuth-pairs), if an A-cycle is afflicted with this then it's the second half of the A-cycle the slippage occurs in.

This is only observed in the early part of the mission, a flight software update subsequently fixed the issue.

The following example (Figure 8.1) is from SOI, and A-cycles 2344 & 2345. These A-cycles have azimuth-pairs summing (so 4 columns in the spectrogram per A-cycles), and the slippage is seen in the second half of A-cycle 2344.

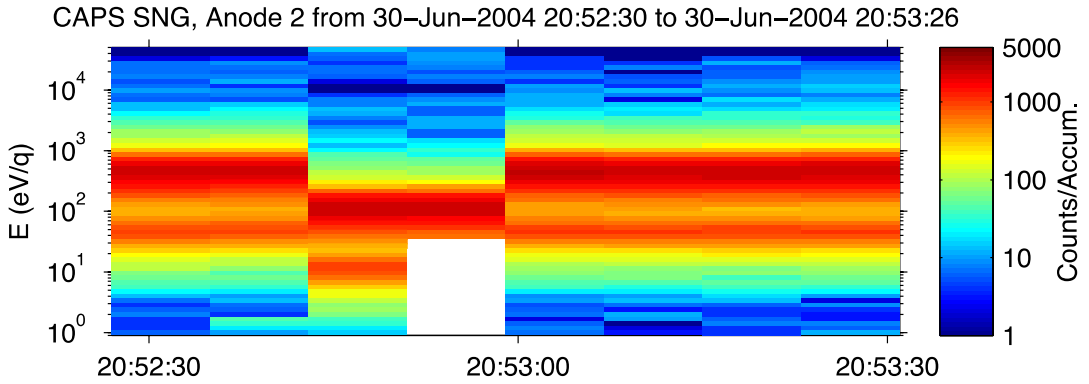

Figure 8.1: Example of SNG 'slippage'.

The explanation for this is that data collection and voltage stepping are handled by different CPUs and they are not synchronized. FSW timing errors can cause them to get out of step with each other. These data could be salvaged by shifting across

sweep boundaries, but no one has ever cared to do that. This also happened on approach (Jan-June, 2004).

8.1.3.3 Telemetry Mode change feature

Sometimes after a telemetry mode change the first azimuth-sum of the second A-cycle in the new telemetry mode is wrong. It is very obvious in the data with a large jump in counts for just that sweep. This is also only observed in the early part of the mission, and the increased counts are in about the top half of the spectra. If you encounter such a feature you should remove it from further analysis.

The top panel of the figure below (Figure 8.2) shows an example where the first 34 energy steps are wrong. The interval is the first few hours of DAYQ 200435800, with the horizontal white line marking the TELEMETRY_MODE number for each azimuth-sum. When TELEMETRY_MODE is 2 the data are summed as azimuth-octs, when TELEMETRY_MDOE is 16 the data are summed as azimuth-quads. Clearly every change to TELEMETRY_MODE 16 has this feature (a 'red vertical line'), but it does not exist when changing back to mode 2.

The lower panel shows a zoom in spectra of the second TELEMETRY_MODE = 16 interval around 00:30, A-cycles 50 to 57 inclusive. Recall this mode has azimuth-quads, therefore 2 columns per A-cycle in the spectra. The first A-cycle (50) is as expected, but the first azimuth-sum of the second A-cycle (51) has outrageously large numbers (but not fill value nor constant) compared to the rest of the interval.

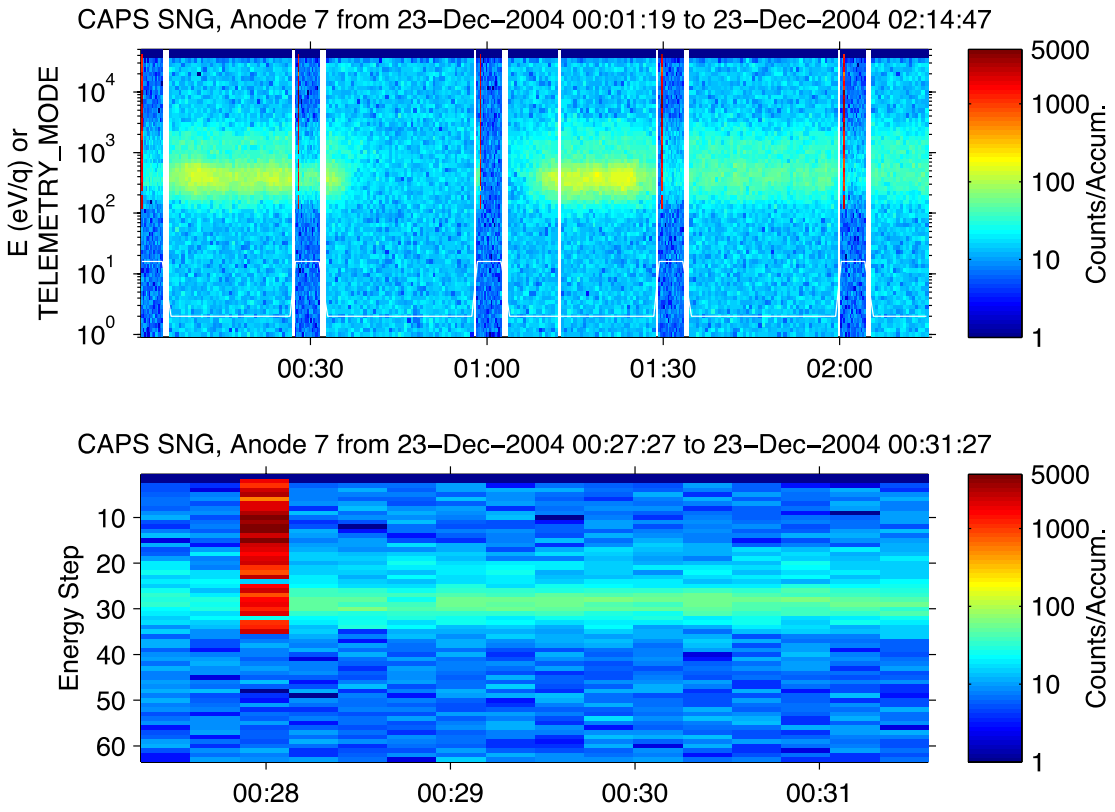

Figure 8.2: Example of SNG Telemetry Mode change feature.

8.1.3.4 SNG Quality Flag

The **ANC** Binaries have an *SNG_QUALITY_FLAG* at A-cycle resolution. It only identifies missing data and bad checksums in the telemetry. (For instance it doesn't flag slippage or mode change errors).

8.2 SNG Geometric Factor values

The geometric factor, GF, for SNG is a function of energy. Therefore you must know the correct eV for each energy step (using the appropriate voltage sweep table).

8.2.1 What to use

The energy dependent geometric factor for one SNG anode is given by:

$$GF(E) = \frac{0.00427 + 0.00169\left(\frac{E}{1000}\right)^{-0.1379}}{8}$$

where E is in eV/q, and GF has units of cm^2 sr eV/eV.

If you want the GF of all 8 anodes, don't divide by 8.
(Source: CAPS team meeting #41)

8.2.2 Where it came from:

Dan Reisenfeld presented the raw data in November 2006 and the 33rd CAPS team meeting. Some snippets of info are pasted below.
A table of 6 GF values was presented, which was then used to construct the formula

Table from Team Meeting (33) presentation 2006 November
Note: In that presentation, Dan forgot it's log10(E(keV)) in the formula.

eV	GF (1 anode) cm^2 sr eV/eV
1	10.80e-4
10	9.30e-4
100	8.06e-4
1000	7.77e-4
10000	6.71e-4
30000	6.68e-4

8.3 SNG Efficiency Values

The efficiency values of the detector are both dependent on energy and ion species, and also the MCP gains.

During the CAPS team meeting #41 the formula were settled on for general use.

The efficiency value, eff, for a particular energy (units of eV/q) is given by:

eff = c*(a*ln((E/1000)+14.557) + b)

where:
- eff is unitless
- E is in eV/q
- and note that ln is natural log (not log10)
- a and b are ion species dependent values
- c is a scaling factor to correct for variations in mcp gain

The CAPS team agreed value for a and b are:

For H^+ ions, a = 0.1110, b = 0.048
For OH^+ ions, a = 0.0665, b = 0.265

Although there are many different ion species that can be detected, only these two have been validated. As such the H^+ values should be used for all light ion species, and the OH^+ values should be used for all other ion species.

The SNG's mcp has been operated at the same mcp level for most of the mission (launch through to 2012-05-16T03:02:34), the CAPS team agree that the scaling factor is a constant for this period, c = 0.85.

A refined value of c as a function of time has yet to be determined, including for dates after 2012-05-16T03:02:34 when the mcp level was raised, as such the best current advice is to use c = 0.85 for all time periods. If you publish your results it is advisable to state what calibration values you applied (i.e. a, b, c).

However there have been a few occasions when the mcp level dropped for a short period of time, predominately during periapsis. The first was during SOI when the gain was lowered for a few hours near closest approach to prevent saturating the detector. In some later approaches the sensor counted too many counts, and for its own safety dropped the mcp level for a period of time before returning to its nominal value. These are very rare occasions and not usually ones encountered. However the ANC object HVU2_ST_DAC can be monitored as this indicates the mcp level. Nominal (i.e. c = 0.85) has an HVU2_ST_DAC around -2611.756104 Volts. If it's more than 30 V different (mcp steps in intervals of ~70.6 or 98.8 V) you know

that you must adjust your value for c accordingly, using the nearest mcp gain test. The operations team does not calculate the c value for these rare cases.

[For more information on the scaling factor see the ELS scaling factor section 9.4. However there are far fewer ions than electrons measured, hence the IMS mcps have not aged enough yet to cause a decrease in gain, as such this value is currently considered to be constant for the entire mission.]

8.4 Converting counts to scientific units

8.4.1 'Counts per accumulation' to 'Counts per second'

This has to be worked out per energy bin or summed energy bin within the sweep – you can not guarantee that the entire HV sweep will have bins all of the same accumulation time. (If the data has no summing at all then they are all the same.)

SNG Accumulation time in seconds of Energy bin # *bin*= dt(*bin*) where:

If TELEMETRY_MODE is not 132
dt(*bin*) = (7 / 8) * (4 / 64) * (LAST_ENERGY_STEP(*bin*) - FIRST_ENERGY_STEP(*bin*) + 1) * ...
(LAST_AZIMUTH_VALUE(*bin*) - FIRST_AZIMUTH_VALUE(*bin*) + 1)

or IF TELEMETRY_MODE = 132
dt(*bin*) = 2*(7 / 8) * (4 / 64) * (LAST_ENERGY_STEP(*bin*) - FIRST_ENERGY_STEP(*bin*) + 1) * ...
(LAST_AZIMUTH_VALUE(*bin*) - FIRST_AZIMUTH_VALUE(*bin*) + 1)

The if statement is for whether anode-summing is going on or not, which only occurs during TELEMETRY_MODE 132. The effect of summing anodes merely doubles the accumulation time per record.

To convert counts per seconds to counts per accumulation you divide by *dt*:

$$C_s = \frac{C_{Accum.}}{dt}$$

8.4.2 PSD, DEF and DNF

Further conversion to scientific units is complicated for SNG data, predominantly because we don't know which ion species are present.

8.4.3 'Counts' to/from 'Phase Space Density'

For a single ion species in the plasma environment:

C(species) = dt * Eff(species,E) * G(E) * v(species,E)^4 * f(species,E)

Where:
C is the measured counts in the time dt at energy E, for a particular ion species.
Eff is the efficiency at energy E for the given species
G is the Geometric Factor at energy E
f is the Phase Space Density (distribution function) of bin 'E' for the given species
v is the velocity of the molecule for any energy bin, given by:

$$v(species, E) = \sqrt{\frac{2\{Energy\_Bin(E)\}}{mass(species)}}$$

$$=> v(species, E)^4 = 4\left(\frac{Energy\_Bin(E)}{mass(species)}\right)^2$$

Remember to convert Energy_Bin in eV into Joules for this calculation.

Note 1: This also assumes that you have already corrected your counts, C, for the effects of cross talk.

Note 2: This formula differs to that quoted for ELS (section 9.3.5) by a factor of 2.

If multiple species are present then the equation becomes:

Total Counts = dt * Eff(species_1,E) * G(E) * v(species_1,E)^4 * f(species_1,E) +
dt * Eff(species_2,E) * G(E) * v(species_2,E)^4 * f(species_2,E) +
dt * Eff(species_3,E) * G(E) * v(species_3,E)^4 * f(species_3,E) +…

This is fine if you have the distribution function of each species already, but if you only have flux you will have to make some assumptions about species partitioning in order to covert them to f.

8.5 Partition counts correctly or else

For a given energy bin we may have 100 counts, but how many of then are H^+ and how many are W^+. Is the ratio 10:90, 50:50 or 90:10? Assumptions have to be made to partition those counts correctly. This is vitally important as there is a v^4 term in the conversion to phase space, which is proportional to m^{-2}.

If you mis-identify ions as H^+ when they should be OH^+, then your v^4 term is now out by a factor of nearly 300 (17^2). Likewise confusing OH^+ for H^+ is just as bad.

But how do you correctly identify what proportion of counts are of each species? You could try to infer proportions from the TOF or ION data, or if you can assume values (if you are in the magnetosheath then you might assume it's all protons). There is no rigorous way to do this – it's left to the user to figure out the best way for their situation.

For this reason you rarely see CAPS ion data in units of phase space density, and spectra tend to be in units of ion counts.

8.6 Energy Summing is very rare

Energy-step summing is very common in ELS data, but is almost non-existent in SNG data. There were a few hours near SOI and in 2011 for DOYs 115 through 121 (at time of writing).

When energy summing occurs it sums energy-pairs except for the first and last record per azimuth-sum, where it is a single energy step. The 32 records in the 'energy summing' telemetry mode will be summed as:

Record #	FIRST_ENERGY_STEP	LAST_ENERGY_STEP
1	1	1
2	3	4
3	5	6
4	7	8
.	.	.
.	.	.
.	.	.
29	57	58
30	59	60
31	61	62
32	63	63

Of the 32 records, records 2 to 31 are summed with neighboring energy-step pairs, while records 1 and 32 are not summed, and have just 1 energy-step.
The difference between energy summing of SNG and ELS is that first record; SNG does not sum a pair but is the singular step 1 with step 2 not used at all (c.f. ELS energy summing record 1 would sum energy step 1 and 2 together to make a pair, see section 7.3.2). As such, both records 1 and 32 have half the accumulation time as records 2 to 32, but if you use the equation given in section 8.4.1 this will be accounted for.

The reason for skipping energy step 2 may seem odd, however as we tend to ignore the top two bins anyway (section 8.1.1) it has no practical consequence. (Also recall that IMS sweep table number 2 has step 1 and 2 at the same value anyway, section 7.8.1)

8.7 IMS Sweep Tables 0, 1 and 2 are similar

IMS sweep tables 0*, 1 and 2 are essentially the same except for the top two and the bottom one. Since we ignore the top 2 data records anyway, we can effectively use table 2 for any table 0 to 2.

[* 0 here refers to IMS_SWEEP_TABLE_0_Std, see section 8.8.]

However if you are using the correct sweep table for each A-cycle as specified in ANC.IMS_SWEEP_TABLE_NUMBER you may notice that while the majority may be 1 or 2 (depending on date) it occasionally has an A-cycle where the value is 0 for a single A-cycle before returning to 1 or 2. i.e. if 10 ANC records have IMS_SWEEP_TABLE_NUMBERs of 2,2,2,2,2,0,2,2,2,2 then the zero is a bug and should really be a 2 with its neighbors. The same is true for 1,1,1,1,0,1,1,1,1,1.

If the IMS Sweep Table does change it will remain changed for many A-cycles, never just 1.

8.8 IMS Sweep Table 0 Could Be Standard or Titans

ANC.IMS_SWEEP_TABLE_NUMBER = 0 could mean either sweep table *IMS_SWEEP_TABLE_0_Std* or *IMS_SWEEP_TABLE_0_Titans*. While it is unfortunate they were given the same number, they do not overlap in time and therefore are easy to separate.

If IMS_SWEEP_TABLE_NUMBER = 0 then:
    If YEAR < 2004 use IMS_SWEEP_TABLE_0_Std
    If YEAR > 2004 use IMS_SWEEP_TABLE_0_Titans

The associated LBL files give the specific date ranges, but the above year separation is sufficient (and neither should occur in 2004).

This rule is only true when the IMS Sweep table number is 0 for more than 1 A-cycle, see section 8.7.

9 ELS Using the Data – General Knowledge

9.1 Things to know about the dataset

The dataset is pretty well behaved, but there are a few oddities to know about.

9.1.1 If using just one anode – use anode 5

Anode 5 is the most commonly used of the 8 anodes for electron studies. So if you can assume isotropic distributions (therefore not having to worry about any pointing requirements) then use anode 5.

This is because its field of view is the least obscured of the 8 anodes. For instance, anode 1 is looking towards the RTG heat shields, and anode 8 is towards the high gain antenna. Those are merely extremes; likewise if the actuator is at an extreme angle then the field of view is nearly along the spacecraft body, which can obscure the path of inbound particles.

This is best illustrated from figure 5 of the CAPS instrument paper (*Young et al.* [2004]) and copied here with its caption (Figure 9.1 also repeated in section 13.1).

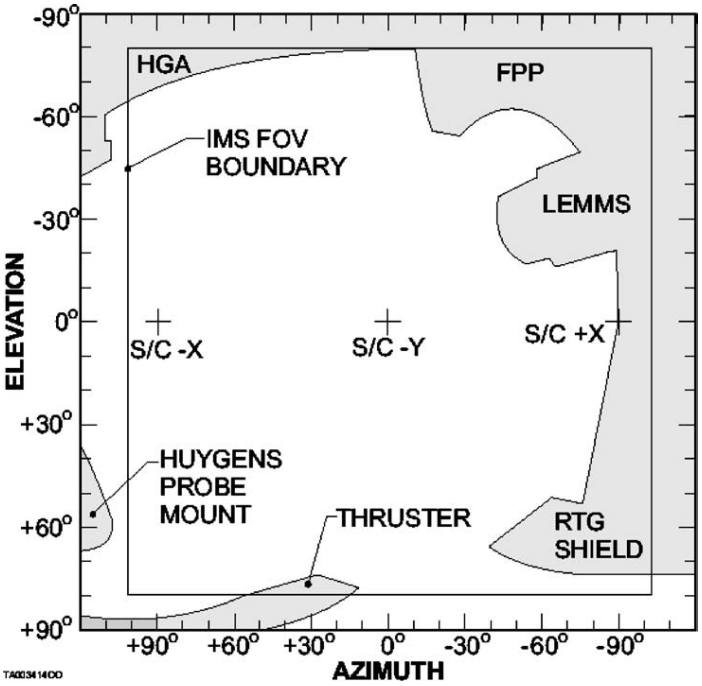

Figure 5. All-sky projection of the CAPS IMS field-of-view. Encroachment on the CAPS FOV are caused by surrounding spacecraft structures (shaded areas). Similar encroachments occur for IBS and ELS sensors.

Figure 9.1: CAPS field-of-view.

9.1.2 Spacecraft Potential is important

Accounting correctly for spacecraft potential is important (see section 16). However remember that different anodes may experience a different potential, so again it must be done on an anode-by-anode basis.

9.1.3 COLLAPSE_FLAG is occasionally 0, 1, 128 or 129

Usually the ELS data has been dead time corrected already, but if the COLLAPSE_FLAG is 0, 128, 1 or 129 you need to do the correction. See the dead time chapter 14, but is simply:

{corrected counts} = {recorded counts} / (1.0 -({recorded counts} * 4.48e-5))

Standard practice has been to only make the correction if it amounts to a whole electron or more, so only if the counts per accumulation are over 149.

The COLLAPSE_FLAG only has these values during flight software versions 1 and 2, hence are not encountered in 2004 and beyond.

9.1.4 COLLAPSE_FLAG is occasionally 136

This is a 'snapshot' (see CAPS instrument paper) and was only used during the cruise phase to Saturn. If your data are from 2004 or later it will not see it. Standard practice by MSSL is to ignore any data from COLLAPSE_FLAG 136.

(More details: snapshots were only taken with the onboard flight software 2 which was between dates 2000-168 and 2003-039; date between Jupiter to Saturn, and rarely looked at. To know which flight software was used to collect the data you can look at the FSW_MAJOR_VERSION object in the ANC files.)

9.1.5 Uneven Azimuth summing can occur pre-Saturn arrival

Normally Azimuths are summed (or not) to a power of 2, i.e. 1, 2, 4, 8 or 16 azimuths may be summed together. However in earlier flight software versions (1 and 2), some telemetry modes summed azimuths together in 3 unequal groups per A-cycle (azimuths summed as 1-5, 6-10, 11-16 giving sums of 5, 5 and 6 azimuths). However the flight software changed (jumped to 4) one year before SOI, therefore this is only observed if you are looking at data long before Cassini's arrival at Saturn.

9.1.6 Entire A-cycles may sample just one energy step

In certain circumstances such as during power-up and during gain tests (see section 17) the ELS sensor can operate while fixed on a single energy step. This will be reflected in the energy step values. During these times, any data gathered cannot be used for general science purposes.

For instance, on 2010-317 from 10:46:47 to 12:33:26, FIRST_ENERGY_STEP = 31 and LAST_ENERGY_STEP = 31 for every single record. In fact the ELS file for DAYQ 201031706 is nothing but this situation. Azimuths are not summed and this is the one time (section 5.3.6) when the ELS.OFFSET_TIME object is useful, and the only way to figure out when (to a ms) a particular record was taken.

9.1.7 Penetrating Radiation and Increased Scale Factor

Penetrating radiation on ELS has the ability to swamp the instrument in counts when close to Saturn and obscuring the intended signal from the local plasma environment. This is generally observed below 6 R_S, although cases have been reported out to 8 R_S.

To avoid totally saturating the detector in counts, ELS lowers the mcp gain when within L=3.5 of Saturn [$L = R*\cos^2(\text{Latitude})$] to reduce the measured counts, which has the effect of increasing the scaling factor. Usually the mcp gain is lowered 3 mcp levels in such situations.

In addition, the mcp gain is lowered for some Enceladus encounters, on a case by case basis, and usually lowered by 4 mcp levels.

See section 9.4 for more about scale factors, and chapter 15 for more details about removing the background caused by penetrating radiation.

9.1.8 Data corruptions

9.1.8.1 Before 2003-040: Energy Step 32 for Anode 6 issue

This is a quirk in the data for one energy step at one anode only, and only occurs when the flight software was version 1 or 2, as such only on dates before 2003-040 (On 2003-040 CAPS moved to flight software version 4, but see variable ANC.FSW_MAJOR_VERSION to check what it was (section 7.5)).

The counts in anode 6 at energy step 32 (~200.9 eV) are elevated above the background. This is easy to spot in the data as a straight line at energy step 32, i.e. the example below for DAYQ 200035000:

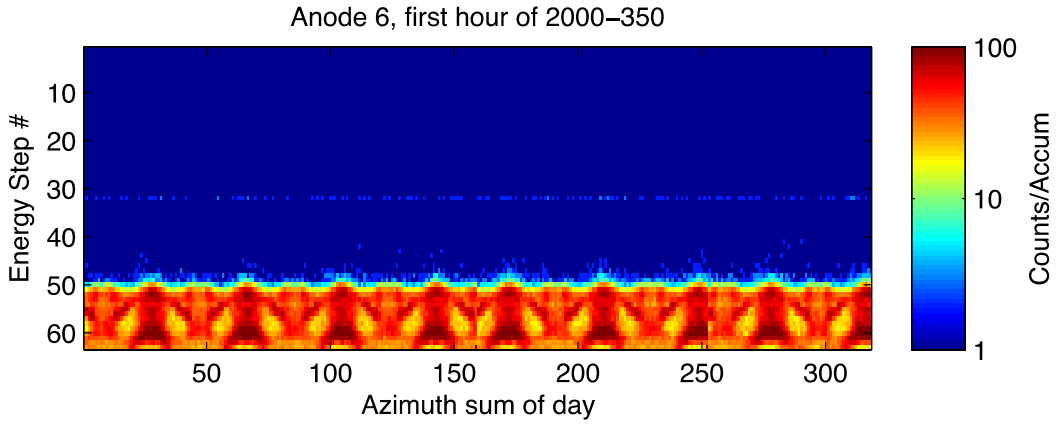

Figure 9.2: Example of ELS anode 6 step 32 pre-2003040 issue.

The usual trick here is to interpolate a value for anode 6 step 32 from anode 6's steps 31 and 33.

9.1.8.2 Use data with ANC.ELS_QUALITY_FLAG = 0 only

The **ANC** Binaries have an *ELS_QUALITY_FLAG* at A-cycle resolution. If this flag is not zero, standard practice is to ignore that data record from your analysis.

9.1.8.3 Quantized counts are not always the expected 256 values

ELS data is quantized, and all ELS.DATA values should be one of those listed (and explained) in section 7.7. For instance, a scan through U3 data found the unexpected values of 255, 511, 767, 1279, 1535, 2559, 5887, and 6143. This is due to a known bug in the ground code that should be fixed for the future U4 data. For now, you may wish to avoid bins containing those numbers.

9.1.9 Safing incident 1st June 2008

During periapsis on 1st June 2008 the environmental penetrating radiation became so strong that ELS automatic safing procedures were invoked. The MCP voltages were dropped several levels until the signal strength was within acceptable limits. However, for a period of about 2 hours the count rate was so high in places that it overflowed the allocated memory buffers. The quality flag does not register this unforeseen problem. Be aware that data from 21:30 to 23:30 UT on this date is unusable for scientific purposes.

9.2 ELS Geometric Factor and Efficiency values

The geometric factor, *GF*, for SNG is a function of energy. The efficiency, *eff*, is a function of anode, energy and also of the mcp gain for said anode. However the mcp gain also varies over time.
To make life simpler, a normalized efficiency table, *neff*, is provided (function of anode and energy) which is multiplied by a scale factor, *SF*, that is per anode and can vary over time: *eff*(E,anode) = *neff*(E,anode) * *SF*(time, anode)

For conversions from counts to scientific units you always need the product *GF* * *eff*, which can be rearranged to *G*(E,anode) * SF(time,anode), where *G* = *GF* * *neff*

G is both energy and anode dependent, but is independent of the time. As there is only one voltage sweep table for ELS, the conversion from ELS energy step to eV/q is also fixed over time. As such the *G* matrix is fixed for the entire mission for ELS.

The following table shows the values for *G*, as a function of Energy Step and anode in units of m^2 ster eV/eV. The data was supplied by the Cassini Ops Team at MSSL (particularly by Gethyn Lewis in 2008 & 2012).

Step	Anode 1	Anode 2	Anode 3	Anode 4	Anode 5	Anode 6	Anode 7	Anode 8
1	6.7104228E-8	7.4592699E-8	7.0812725E-8	5.3971015E-8	4.6505812E-8	4.0354702E-8	3.4282813E-8	2.8610931E-8
2	7.0764056E-8	7.8876361E-8	7.5013209E-8	5.7963555E-8	5.0872651E-8	4.5095842E-8	3.9273487E-8	3.3393660E-8
3	7.4401168E-8	8.3133435E-8	7.9187623E-8	6.1931314E-8	5.5212387E-8	4.9807556E-8	4.4233186E-8	3.8146704E-8
4	7.7995743E-8	8.7340721E-8	8.3313215E-8	6.5852669E-8	5.9501369E-8	5.4464164E-8	4.9134879E-8	4.2844159E-8
5	8.1655013E-8	9.1623728E-8	8.7513058E-8	6.9844599E-8	6.3867542E-8	5.9204581E-8	5.4124792E-8	4.7626158E-8
6	8.5270324E-8	9.5855285E-8	9.1662449E-8	7.3788574E-8	6.8181265E-8	6.3888052E-8	5.9054762E-8	5.2350711E-8
7	8.8898984E-8	1.0010247E-7	9.5827162E-8	7.7747113E-8	7.2510916E-8	6.8588817E-8	6.4002935E-8	5.7092711E-8
8	9.2515552E-8	1.0433549E-7	9.9977995E-8	8.1692459E-8	7.6826138E-8	7.3273915E-8	6.8934618E-8	6.1818906E-8
9	9.6140685E-8	1.0857855E-7	1.0413866E-7	8.5647150E-8	8.1151580E-8	7.7970109E-8	7.3877981E-8	6.6556295E-8
10	9.9775817E-8	1.1283330E-7	1.0831080E-7	8.9612749E-8	8.5488954E-8	8.2679258E-8	7.8834980E-8	7.1306751E-8
11	1.0340247E-7	1.1707814E-7	1.1247321E-7	9.3569101E-8	8.9816214E-8	8.7377426E-8	8.3780420E-8	7.6046130E-8
12	1.0702705E-7	1.2132054E-7	1.1663324E-7	9.7523182E-8	9.4140990E-8	9.2072897E-8	8.8723022E-8	8.0782790E-8
13	1.1066372E-7	1.2557709E-7	1.2080714E-7	1.0149046E-7	9.8480196E-8	9.6784035E-8	9.3682115E-8	8.5535253E-8
14	1.1428066E-7	1.2981056E-7	1.2495841E-7	1.0543622E-7	1.0279587E-7	1.0146963E-7	9.8614316E-8	9.0261946E-8
15	1.1836930E-7	1.3452680E-7	1.2958697E-7	1.0980375E-7	1.0752401E-7	1.0661420E-7	1.0401392E-7	9.5464308E-8
16	1.2722209E-7	1.4407849E-7	1.3899889E-7	1.1837697E-7	1.1633021E-7	1.1630567E-7	1.1403155E-7	1.0538875E-7
17	1.3609489E-7	1.5365178E-7	1.4843208E-7	1.2696958E-7	1.2515631E-7	1.2601906E-7	1.2407182E-7	1.1533563E-7
18	1.4497262E-7	1.6323039E-7	1.5787051E-7	1.3556697E-7	1.3398732E-7	1.3573784E-7	1.3411768E-7	1.2528804E-7
19	1.5380000E-7	1.7275466E-7	1.6725541E-7	1.4411558E-7	1.4276824E-7	1.4540150E-7	1.4410655E-7	1.3518399E-7
20	1.6270191E-7	1.8235935E-7	1.7671954E-7	1.5273638E-7	1.5162330E-7	1.5514675E-7	1.5417976E-7	1.4516350E-7
21	1.7154306E-7	1.9189849E-7	1.8611909E-7	1.6129834E-7	1.6041792E-7	1.6482548E-7	1.6418422E-7	1.5507489E-7
22	1.8029665E-7	2.0132424E-7	1.9540473E-7	1.6975552E-7	1.6910253E-7	1.7439133E-7	1.7408507E-7	1.6488437E-7
23	1.8640361E-7	2.0747781E-7	2.0141845E-7	1.7520982E-7	1.7465007E-7	1.8068475E-7	1.8089129E-7	1.7164397E-7
24	1.9250830E-7	2.1362910E-7	2.0742994E-7	1.8066210E-7	1.8019555E-7	1.8697585E-7	1.8769500E-7	1.7840107E-7
25	1.9862516E-7	2.1979265E-7	2.1345341E-7	1.8612526E-7	1.8575209E-7	1.9327948E-7	1.9451227E-7	1.8517164E-7
26	2.0469797E-7	2.2591182E-7	2.1943351E-7	1.9154907E-7	1.9126861E-7	1.9953772E-7	2.0128044E-7	1.9189346E-7
27	2.1085719E-7	2.3211806E-7	2.2549870E-7	1.9705005E-7	1.9686363E-7	2.0588501E-7	2.0814492E-7	1.9871092E-7
28	2.1692768E-7	2.3823488E-7	2.3147651E-7	2.0247179E-7	2.0237804E-7	2.1214086E-7	2.1491050E-7	2.0543016E-7
29	2.2300745E-7	2.4436107E-7	2.3746346E-7	2.0790181E-7	2.0790089E-7	2.1840628E-7	2.2168644E-7	2.1215968E-7
30	2.2149554E-7	2.4344606E-7	2.3681817E-7	2.0714254E-7	2.0768476E-7	2.1819754E-7	2.2235034E-7	2.1255202E-7
31	2.1991236E-7	2.4246339E-7	2.3610847E-7	2.0632365E-7	2.0741180E-7	2.1792458E-7	2.2295086E-7	2.1287957E-7
32	2.1832594E-7	2.4147872E-7	2.3539732E-7	2.0550309E-7	2.0713828E-7	2.1765106E-7	2.2355260E-7	2.1320779E-7
33	2.1673669E-7	2.4049229E-7	2.3468489E-7	2.0468106E-7	2.0686427E-7	2.1737705E-7	2.2415542E-7	2.1353660E-7
34	2.1515818E-7	2.3951253E-7	2.3397728E-7	2.0386459E-7	2.0659211E-7	2.1710489E-7	2.2475417E-7	2.1386319E-7
35	2.1352369E-7	2.3849079E-7	2.3323793E-7	2.0301723E-7	2.0630805E-7	2.1681971E-7	2.2536376E-7	2.1419318E-7
36	2.1025867E-7	2.3508381E-7	2.3050525E-7	2.0095885E-7	2.0531435E-7	2.1561307E-7	2.2461849E-7	2.1330595E-7
37	2.0689027E-7	2.3156896E-7	2.2768605E-7	1.9883529E-7	2.0428918E-7	2.1436823E-7	2.2384961E-7	2.1239062E-7
38	2.0346240E-7	2.2799205E-7	2.2481707E-7	1.9667424E-7	2.0324592E-7	2.1310141E-7	2.2306716E-7	2.1145914E-7
39	2.0020741E-7	2.2459554E-7	2.2209279E-7	1.9462218E-7	2.0225527E-7	2.1189847E-7	2.2232418E-7	2.1057463E-7
40	1.9686833E-7	2.2111128E-7	2.1929812E-7	1.9251711E-7	2.0123903E-7	2.1066446E-7	2.2156199E-7	2.0966727E-7
41	1.9346331E-7	2.1755822E-7	2.1644827E-7	1.9037047E-7	2.0020272E-7	2.0940609E-7	2.2078476E-7	2.0874200E-7
42	1.9006707E-7	2.1401432E-7	2.1360577E-7	1.8822936E-7	1.9916908E-7	2.0815095E-7	2.2000953E-7	2.0781911E-7
43	1.8676958E-7	2.1057345E-7	2.1084591E-7	1.8615050E-7	1.9816550E-7	2.0693231E-7	2.1925684E-7	2.0692305E-7
44	1.8330953E-7	2.0696297E-7	2.0795000E-7	1.8396917E-7	1.9711244E-7	2.0565360E-7	2.1846705E-7	2.0598282E-7
45	1.8017814E-7	2.0369543E-7	2.0532916E-7	1.8199503E-7	1.9615941E-7	2.0449634E-7	2.1775228E-7	2.0513190E-7
46	1.7652803E-7	1.9988662E-7	2.0227418E-7	1.7969387E-7	1.9504851E-7	2.0314739E-7	2.1691910E-7	2.0414003E-7
47	1.7347792E-7	1.9670390E-7	1.9972137E-7	1.7777098E-7	1.9412021E-7	2.0202017E-7	2.1622288E-7	2.0331119E-7
48	1.6994595E-7	1.9301836E-7	1.9676527E-7	1.7554430E-7	1.9304526E-7	2.0071488E-7	2.1541667E-7	2.0235142E-7
49	1.6662052E-7	1.8954835E-7	1.9398203E-7	1.7344783E-7	1.9203318E-7	1.9948591E-7	2.1465760E-7	2.0144777E-7
50	1.6270001E-7	1.8545738E-7	1.9070073E-7	1.7097621E-7	1.9083998E-7	1.9803703E-7	2.1376270E-7	2.0038242E-7
51	1.5921305E-7	1.8181881E-7	1.8778230E-7	1.6877791E-7	1.8977873E-7	1.9674837E-7	2.1296677E-7	1.9943488E-7
52	1.5651838E-7	1.7900698E-7	1.8552697E-7	1.6707909E-7	1.8895861E-7	1.9575251E-7	2.1235168E-7	1.9870263E-7
53	1.5346322E-7	1.7581899E-7	1.8296994E-7	1.6515301E-7	1.8802878E-7	1.9462343E-7	2.1165430E-7	1.9787242E-7
54	1.4986468E-7	1.7206399E-7	1.7995812E-7	1.6288437E-7	1.8693358E-7	1.9329353E-7	2.1083290E-7	1.9689456E-7
55	1.4557728E-7	1.6759017E-7	1.7636975E-7	1.6018144E-7	1.8562871E-7	1.9170905E-7	2.0985425E-7	1.9572951E-7
56	1.4301547E-7	1.6491698E-7	1.7422563E-7	1.5856638E-7	1.8484903E-7	1.9076230E-7	2.0926949E-7	1.9503337E-7
57	1.4015757E-7	1.6193483E-7	1.7183369E-7	1.5676466E-7	1.8397924E-7	1.8970611E-7	2.0861714E-7	1.9425676E-7
58	1.3685837E-7	1.5849218E-7	1.6907240E-7	1.5468473E-7	1.8297513E-7	1.8848684E-7	2.0786406E-7	1.9336024E-7
59	1.3295624E-7	1.5442040E-7	1.6580649E-7	1.5222469E-7	1.8178753E-7	1.8704475E-7	2.0697336E-7	1.9229988E-7
60	1.2818042E-7	1.4943693E-7	1.6180934E-7	1.4921385E-7	1.8033402E-7	1.8527977E-7	2.0588323E-7	1.9100211E-7
61	1.2202332E-7	1.4301213E-7	1.5665611E-7	1.4533220E-7	1.7846012E-7	1.8300432E-7	2.0447780E-7	1.8932898E-7
62	1.1334537E-7	1.3395688E-7	1.4939305E-7	1.3986132E-7	1.7581901E-7	1.7979725E-7	2.0249697E-7	1.8697085E-7
63	9.8510323E-8	1.1847682E-7	1.3697676E-7	1.3050878E-7	1.7130399E-7	1.7431472E-7	1.9911070E-7	1.8293959E-7

If the ELS data record you are working with has energy summing (i.e. FIRST_ENERGY_STEP is different to LAST_ENERGY_STEP) then you can use the above table to interpolate the value you need for *G*.

9.3 Converting counts to scientific units

Note that all conversions are anode and energy dependent.

9.3.1 'Counts per accumulation' to 'Counts per second' measured

This has to be worked out per energy bin or summed energy bin within the sweep – you can not guarantee that the entire HV sweep will have bins all of the same accumulation time. i.e. When a sweep has energy summing there are 31 records with energy steps summed, and 1 record with a single step to cover the 63 energy steps.

ELS Accumulation time in seconds of Energy bin # *bin*= dt(*bin*) where:

dt(*bin*) = (3 / 4) * (2 / 64) * (LAST_ENERGY_STEP(*bin*) - FIRST_ENERGY_STEP(*bin*) + 1) * ...
(LAST_AZIMUTH_VALUE(*bin*) - FIRST_AZIMUTH_VALUE(*bin*) + 1)

To convert counts per accumulation to counts per second you divide by *dt*:

$$C_s = \frac{C_{Accum.}}{dt}$$

9.3.2 'Counts per accumulation' to 'Counts per second' normalized for gain

To correct for the variable effect of the mcp gain, we must include the scale factor for that time and anode in order to correctly convert to units of counts per second.

$$C_{s\_corrected} = \frac{C_{Accum.}}{(dt)(SF)}$$

9.3.3 Differential Energy Flux

Differential energy flux, DEF, is counts per second divided by *G* (anode and energy step dependent) and has units of m^{-2} $ster^{-1}$ s^{-1}

$$DEF = \frac{C_{Accum.}}{(dt)(SF)(G)}$$

9.3.4 Differential Number Flux

Differential number flux, DNF, is DEF divided by the energy (*E*, in eV) and charge (*q*, in Coulombs) of the particle and has units of m^{-2} $ster^{-1}$ s^{-1} J^{-1}. As we are measuring electrons, q = 1.602176487E-19 C.

$$DNF = \frac{C_{Accum.}}{(dt)(SF)(G)(qE)}$$

9.3.5 Phase Space Density

Phase Space Density, PSD, is DEF essentially divided by v^4 (using $E = 0.5mv^2$), but a factor of 2 out. That extra 2 is already accounted for within *G*.

PSD has units of m^{-6} s^{-3}

$$PSD = \frac{C_{Accum.}}{(dt)(SF)(G)} \frac{m^2}{2q^2E^2}$$

Here m is the mass of the particle in kg, which for an electron is 9.10938188E-31 kg. Note that the 2 is NOT squared, this is different to the equivalent SNG formula in section 8.4.2.

The PSD calculation should be carried out in the spacecraft frame as above, then shifted by the spacecraft potential by applying Liouville's theorem.

9.3.6 Example of unit conversion

For this example the first Azimuth-sum of file ELS_200528400 is used, that is A-cycle 1, and just anode 5 will be examined. It is hoped you can use this table to check your own code.

The first/last energy step/azimuth and the counts per accumulation are shown in the first 5 columns, taken directly from the binary files.
The sixth column is the accumulation period of the record, calculated as shown in section 9.3.1.
The seventh column converts the count to per second.
The last column then corrects that with the scaling factor to adjust for mcp gain.

FIRST_ENERGY_STEP	LAST_ENERGY_STEP	FIRST_AZIMUTH_VALUE	LAST_AZIMUTH_VALUE	Counts/Accum	dt	Measured Counts/s	Gain Corrected Counts/s
1	2	1	4	20	0.1875	106.67	109.18
3	4	1	4	23	0.1875	122.67	125.55
5	6	1	4	43	0.1875	229.33	234.73
7	8	1	4	86	0.1875	458.67	469.46
9	10	1	4	138	0.1875	736.00	753.32
11	12	1	4	176	0.1875	938.67	960.76
13	14	1	4	223	0.1875	1189.33	1217.33
15	16	1	4	283	0.1875	1509.33	1544.86
17	18	1	4	303	0.1875	1616.00	1654.04
19	20	1	4	348	0.1875	1856.00	1899.69
21	22	1	4	398	0.1875	2122.67	2172.63
23	24	1	4	524	0.1875	2794.67	2860.45
25	26	1	4	580	0.1875	3093.33	3166.15
27	28	1	4	789	0.1875	4208.00	4307.05
29	30	1	4	966	0.1875	5152.00	5273.27
31	32	1	4	1228	0.1875	6549.33	6703.50
33	34	1	4	1315	0.1875	7013.33	7178.42
35	36	1	4	1360	0.1875	7253.33	7424.07
37	38	1	4	1315	0.1875	7013.33	7178.42
39	40	1	4	1108	0.1875	5909.33	6048.44
41	42	1	4	903	0.1875	4816.00	4929.37
43	44	1	4	643	0.1875	3429.33	3510.06
45	46	1	4	489	0.1875	2608.00	2669.39
47	48	1	4	336	0.1875	1792.00	1834.18
49	50	1	4	265	0.1875	1413.33	1446.60
51	52	1	4	188	0.1875	1002.67	1026.27
53	54	1	4	158	0.1875	842.67	862.50
55	56	1	4	138	0.1875	736.00	753.32
57	58	1	4	113	0.1875	602.67	616.85
59	60	1	4	208	0.1875	1109.33	1135.45
61	62	1	4	1407	0.1875	7504.00	7680.64
63	63	1	4	873	0.09375	9312.00	9531.20

In this example it is clear there is energy summing, therefore we must interpolate energies and G values for further calculations. For simplicity I have used the average value of the two Energy steps for E, and the average value of the two corresponding G values. The next table repeats the first column of above as a key, then lists the E and G I will use. The final 3 columns are the gain correct counts/s

converted into DEF, DNF and PSD using those E and G values, along with the m and q given earlier.

FIRST_ENERGY_STEP	E (eV)	G (in Matlab)	DEF	DNF	PSD
1	2.4134E+04	4.8689E-08	2.2423E+09	5.7992E+23	6.2228E-23
3	1.7624E+04	5.7357E-08	2.1890E+09	7.7525E+23	1.1392E-22
5	1.2872E+04	6.6024E-08	3.5552E+09	1.7240E+24	3.4685E-22
7	9.4085E+03	7.4669E-08	6.2873E+09	4.1709E+24	1.1480E-21
9	6.8755E+03	8.3320E-08	9.0413E+09	8.2076E+24	3.0914E-21
11	5.0230E+03	9.1979E-08	1.0445E+10	1.2979E+25	6.6916E-21
13	3.6695E+03	1.0064E-07	1.2096E+10	2.0574E+25	1.4520E-20
15	2.6805E+03	1.1193E-07	1.3802E+10	3.2139E+25	3.1049E-20
17	1.9585E+03	1.2957E-07	1.2765E+10	4.0682E+25	5.3792E-20
19	1.4315E+03	1.4720E-07	1.2906E+10	5.6271E+25	1.0180E-19
21	1.0461E+03	1.6476E-07	1.3187E+10	7.8681E+25	1.9479E-19
23	7.6415E+02	1.7742E-07	1.6122E+10	1.3168E+26	4.4627E-19
25	5.5830E+02	1.8851E-07	1.6796E+10	1.8777E+26	8.7094E-19
27	4.0765E+02	1.9962E-07	2.1576E+10	3.3035E+26	2.0986E-18
29	2.9815E+02	2.0779E-07	2.5378E+10	5.3126E+26	4.6143E-18
31	2.1795E+02	2.0728E-07	3.2341E+10	9.2616E+26	1.1004E-17
33	1.5930E+02	2.0673E-07	3.4724E+10	1.3605E+27	2.2117E-17
35	1.1625E+02	2.0581E-07	3.6072E+10	1.9367E+27	4.3143E-17
37	8.4970E+01	2.0377E-07	3.5228E+10	2.5877E+27	7.8866E-17
39	6.2300E+01	2.0175E-07	2.9980E+10	3.0036E+27	1.2485E-16
41	4.5405E+01	1.9969E-07	2.4686E+10	3.3934E+27	1.9354E-16
43	3.3165E+01	1.9764E-07	1.7760E+10	3.3423E+27	2.6098E-16
45	2.4275E+01	1.9560E-07	1.3647E+10	3.5089E+27	3.7432E-16
47	1.7795E+01	1.9358E-07	9.4749E+09	3.3233E+27	4.8362E-16
49	1.2810E+01	1.9144E-07	7.5566E+09	3.6818E+27	7.4431E-16
51	9.3050E+00	1.8937E-07	5.4194E+09	3.6352E+27	1.0117E-15
53	6.9750E+00	1.8748E-07	4.6005E+09	4.1167E+27	1.5284E-15
55	4.9350E+00	1.8524E-07	4.0668E+09	5.1434E+27	2.6990E-15
57	3.7700E+00	1.8348E-07	3.3620E+09	5.5661E+27	3.8233E-15
59	2.6100E+00	1.8106E-07	6.2711E+09	1.4997E+28	1.4879E-14
61	1.4500E+00	1.7714E-07	4.3359E+10	1.8664E+29	3.3333E-13
63	5.8000E-01	1.7130E-07	5.5639E+10	5.9874E+29	2.6733E-12

9.4 Obtaining the Scale Factor

The scale factor is calculated by the Cassini operations team at MSSL, who can provide an IDL save file of the daily scale factors. For further details email the operations team:
http://www.mssl.ucl.ac.uk/planetary/missions/Cassini/cassini.php

Scale factors are used to correct the received data for anode degradation. They vary with time, MCP level and anode. ELS scale factors are provided as a table, with a single value for each anode at every possible operating level (40 to 64) on each day of the mission. [The MCPs were designed to work at a nominal voltage of around 2600V. They can be commanded up or down in steps of 58.73 volts, known as MCP levels. MCP level * 58.73 = voltage, at launch the operating level was 45]

A gain test provides a curve of counts received versus MCP operating level for each anode at a particular point in time. The inflection point of each gain test curve marks the threshold value, in operating levels, of the anode (see Figure 9.3). The increase in threshold value over time is a proxy for anode degradation.

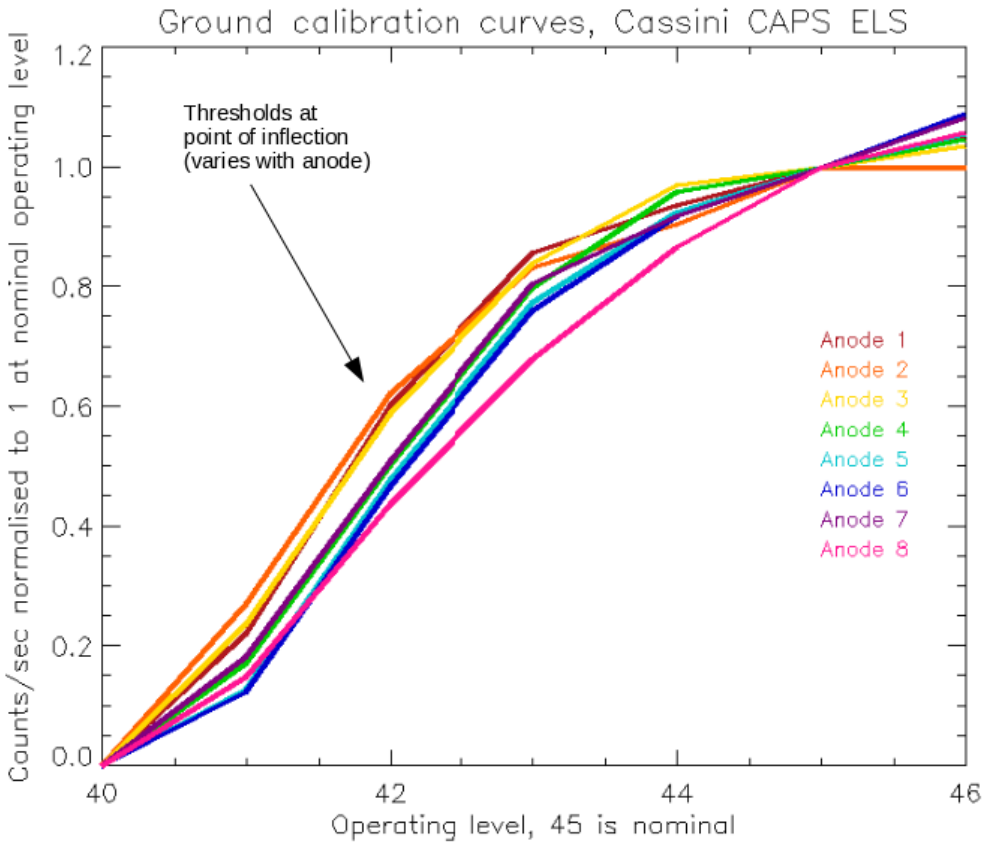

Figure 9.3: Example ELS ground calibration curves.

It is important to note that the anodes only degrade when the instrument is switched on, so account must be taken of periods when it is not. Gain tests occur every 15 days whenever possible, but are sometimes missed due to operational concerns or data loss. Line-fitting to a plot of threshold by time allows extrapolation to values yet to come. Since the launch of Cassini, four anode degradation regimes have been identified. Rapid degradation occurred between launch and the beginning of the cruise phase. Very little degradation occurred during cruise phase, as electron activity was low and the MCPs were not operating at optimal level. Degradation rose as Cassini neared Saturn and began orbit, but the rate of degradation slowed after two years in orbit. Each of the four regimes is represented by a straight-line best fit to the thresholds, with the last line extrapolated to the end of mission. This schema may be changed if a new degradation regime is entered.

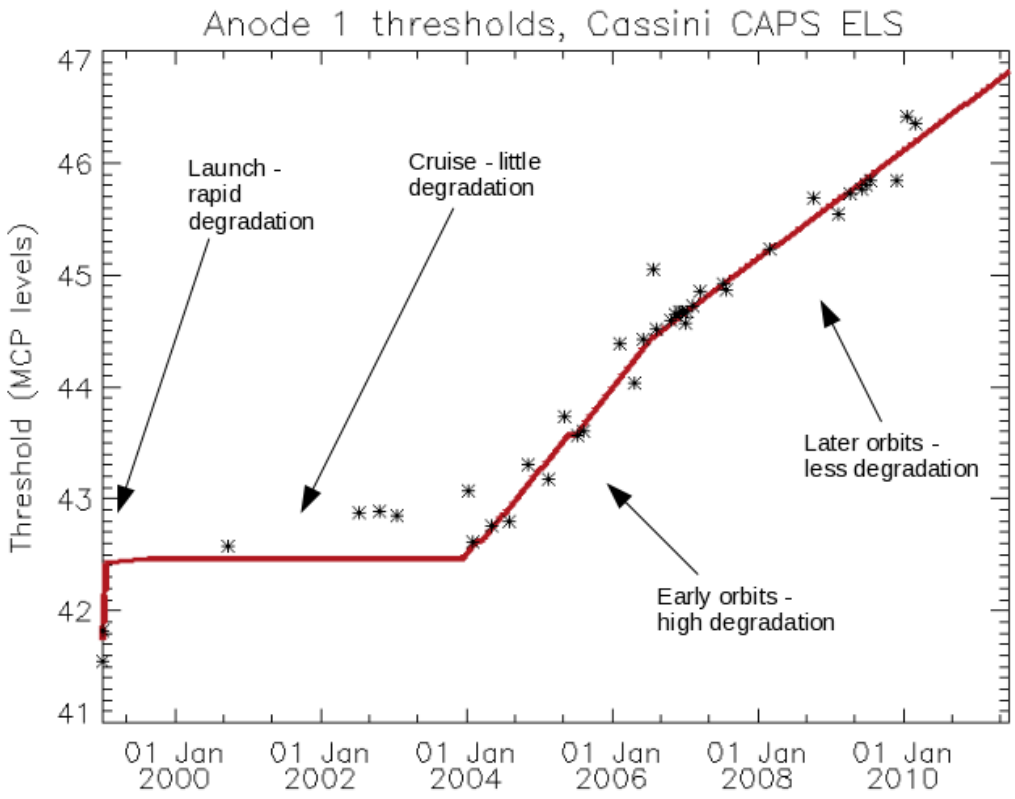

Figure 9.4: Example ELS calibration thresholds vs. time.

To counter the degradation in flight, the MCP operating level is raised approximately once per year. This produces a saw-tooth effect on a plot of effective degradation. Note that the anodes degrade at different rates, but only one MCP setting is applied to all of them, hence that setting has to be a compromise to get the best response across all the anodes.

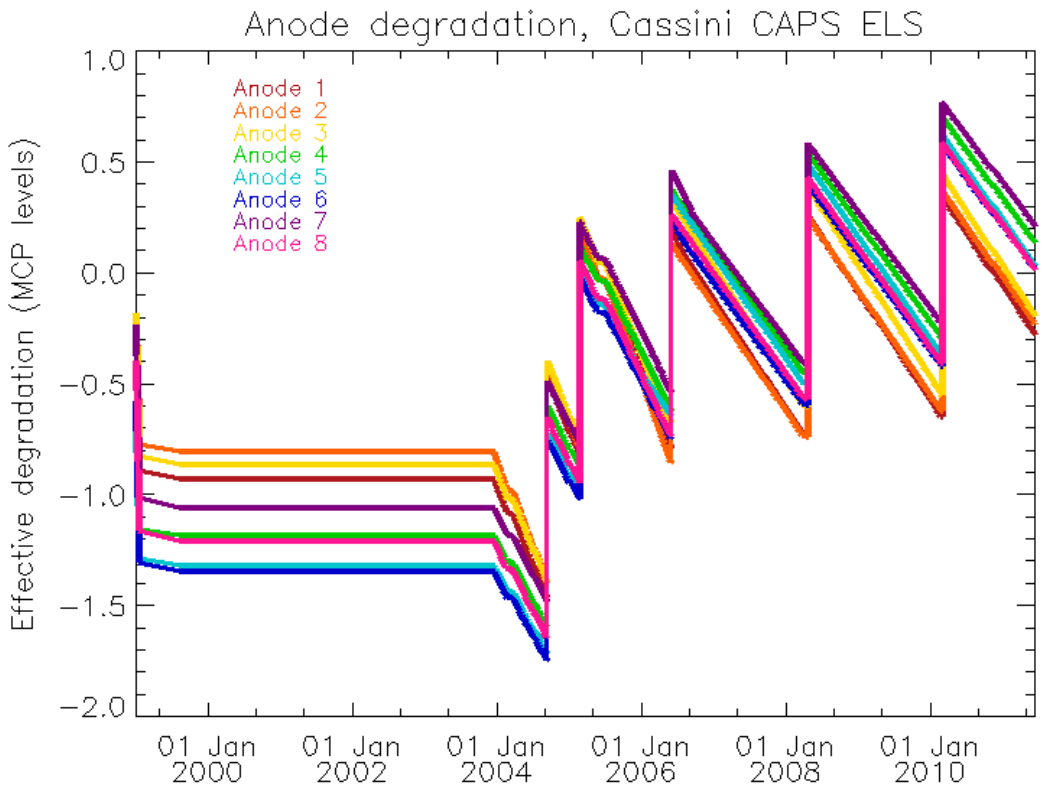

Figure 9.5: ELS anode degradation vs. time.

The scale factor is the correction needed to calculate the number of counts that would have been seen during on-ground pre-flight calibration from the number seen in flight, from which an absolute number of counts can be found. The scale factor is given as a divisor to the value seen.

10 TOF

10.1 What is the principle of operation of the TOF portion of IMS?

The IMS consists of a top-hat electrostatic analyzer for E/q determination, followed by a time-of-flight analyzer for determination of ion velocity, and hence mass-per-charge (m/q) determination. The plate voltage on the electrostatic analyzer is swept every 4 s to cover the nominal E/q range of 1–50 keV in 64 logarithmically spaced bins. Time of flight is measured by the detection of a "start" electron liberated from a thin carbon foil as the incident ion passes through, in coincidence with a "stop" detection of the emerging ion (or its neutralized counterpart) after traversing a flight path of 18.8 cm. The start-only signals (the so-called singles, SNG, data product) are monitored separately from the coincidences, yielding a higher detection efficiency (by roughly an order of magnitude) but no mass determination. The start anode of the time-of-flight section has a position-sensitive readout that enables the direction of motion of detected particles to be identified and binned in 20° segments. In the orthogonal direction, particle velocities are sampled by an actuator that physically rotates the instrument so that the planar field-of-view covers between -80° and +104° relative to the direction that is radially outward from the spacecraft (the spacecraft -y axis).

The ions exiting the electrostatic analyzer are post-accelerated through ~15 kV (14,577 V to be exact) and pass through the carbon foil, where they exit either as negative or neutral fragments. If the incident ion is a molecule, it is also disassociated into its constituent atoms by the foil. The IMS mass analyzer is actually composed of two sensors: a high-resolution sensor (resolution $M/\Delta M \sim 60$) that takes advantage of the linearly-varying electric field (LEF) within the TOF analyzer to detect time-focused positive ions and a medium-resolution sensor ($M/\Delta M \sim 8$) that measures the TOF of species that exit the foil as neutrals, negative ions, or positive ions >16 keV. Although of significantly lower mass resolution, this so-called "straight-through" (ST) sensor detects species with ~20 times the efficiency of the LEF sensor. In addition, identification of the molecular parent ion is possible with the ST sensor, but not as easily with the LEF sensor

10.2 When do we have TOF data?

TOF data are not present all the time, it is only returned to Earth during higher-rate telemetry modes. In practice, this averages out over orbits as about 1 in 6 intervals have TOF data.

TOF data are taken during B-cycles (see section 7.4) such that the TOF binary files always list a B-cycle number that is not the MISSING_CONSTANT value of 65535. SNG/ELS binary data list both A-cycle and B-cycle numbers for each record (although the B-cycle number is often the MISSING_CONSTANT fill value). The TOF

binary records do not list A-cycle numbers however – as for TOF data B-cycle is the lowest resolution and it's impossible to identify which of the many A-cycles within the B-cycle any particular count came from.

10.3 What types of TOF data are there?

There are two types, the straight through (ST) and the linear electric field (LEF). Both data sets are given within the same TOF record. In practice most people use ST data (the DATA_ST variable in the record), as there are far more counts to work with than the DATA_LEF set. However see the CAPS instrument paper for details of which you may wish to use for your work. The following descriptions will apply to both.

10.4 Why are there only 32 different energy steps instead of 63?

TOF is part of the IMS sensor and as such it uses the same top-hat analyzer (and its voltage sweep table) as the SNG data, which we know has 63 different energy steps. So why does TOF data only have 32 steps?

Firstly, the IMS instrument is designed to do TOF, and the SNG's data are actually a consequence of that. To get a time of flight measurement you need to detect an incoming ion and measure how long it takes to travel a fixed distance. This requires you measure a start signal and a stop signal from the ion as it passes through a start foil and a stop foil. SNGs data are this start signal for the TOF. However the incident ion must continue on through the chamber to be clocked for speed yet must still have enough energy to hit a hit the second foil and generate electrons to create a stop signal. This is a second detection that must be carried out, and as such the efficiency for having an ion create a start and stop signal is much lower than the efficiency in just creating a start signal. Not all ions can create a stop signal – and a co-incident start and stop signal has to register for a TOF ion count to be measured. (The term co-incident data is often used here, although doesn't mean simultaneous as there is the duration in time between the start and stop signals. However this is quite short and if no stop signal is recorded within a specific time the instrument knows the ion must have failed to generate a stop signal (it simply cannot be 'late') and can reset itself ready for the next start signal.)

Therefore TOF data in general has far fewer counts than you would expect if you just summed the same SNG data. However the energy table for the SNG data must impact that used in the TOF.

TOF data are a big data product, 512 TOF channels returned for each energy step. It's so big that there simply isn't space to transmit 63 sets (one for each energy) of 512 values down to Earth each B-cycle… hence 32 sets of energies (or energy sums) get downlinked. There are 3 options to compress the full data down:

I. Sum the 63 energy steps into 32 energy step-pairs
II. Return only the even numbered energy steps
III. Return only the odd numbered energy steps

Option I. is the most common set up. Here the energy steps are summed such that:

TOF Energy Step 1 = SNG Energy Step 1
TOF Energy Step 2 = SNG Energy Step 2 + SNG Energy Step 3
TOF Energy Step 3 = SNG Energy Step 4 + SNG Energy Step 5
TOF Energy Step 4 = SNG Energy Step 6 + SNG Energy Step 7
.
.
.
TOF Energy Step 31 = SNG Energy Step 60 + SNG Energy Step 61
TOF Energy Step 32 = SNG Energy Step 62 + SNG Energy Step 63

The TOF Energy Step 1 is just a single SNG Energy Step, not summed. This must be remembered if you convert the data to science units, as the accumulation time will be shorter than all other TOF energy steps.

You can identify which option was used from the binary files. Each record has the variables ST_ENERGY_COLLAPSE and LEF_ENERGY_COLLAPSE. These have the same descriptions of value:

0 = sum adjacent energies (option I. above)
1 = take even energies (option II. above)
2 = take odd energies (option III. above)

In principle the ST and LEF Energy collapse values could be different – but will be the same for all values within the same B-cycle.

10.5 TOF Energy Tables

Many TOF spectrograms are not shown with units of eV, instead keeping to step number (1 to 32). This is for ease of plotting. The following tells you how to convert this to eV.

The section above tells you which SNG energy step maps to a TOF energy step. If the energy collapse is odd or even, then it's a simple look up table as we're mapping 1:1.

However, if adjacent energies are summed (say energy step n and step n+1) then:

The upper energy limit of the TOF step is the upper energy limit of SNG step n.
The lower energy limit of the TOF step is the lower energy limit of SNG step n+1.

That's all you need to plot an accurate spectrogram.

The center energy of the TOF step is the average of SNG step n and step n+1. This is the most trivial to work out, and as such the most commonly used, however it may be more correct to do a log average instead (to represent the log nature of the sweep table). Another option is to use the mid-point between the upper and lower energy limits.

10.6 Does TOF have any pointing information?

ELS, SNG and ION datasets all have 8 anodes, giving 8 different directions, while IBS data has 3 anodes. TOF data however only has 1 large anode summed over all 8 IMS anodes (to count as many ions as possible due to telemetry limitations) – any accurate directional azimuth information is lost completely. In addition, since the duration of a B-cycle is a minimum of 256 seconds, and since this is longer than the time it takes the actuator to do a full angle sweep, all elevation information is lost – the field of view can be as much as half the sky. The one exception is if the actuator is parked in a fixed position throughout – you will then have the same field of view as that of all the SNG anodes field of view. If the actuator is moving however, then during 1 B-cycle's duration the actuator has moved through most of its range and your field of view is probably half the sky.

As a side note, the start of a B-cycle is in no way synchronized with the actuator, it's possible that during a B-cycle IMS doesn't point into the main ion flow direction at all. Normally you can look at SNG data for around the B-cycle and see if that's true or not.

10.7 Are TOF counts quantized?

The TOF data are compressed from 4-bytes to 2-bytes, however it is unlikely that TOF measured enough counts to get outside of the 1:1 linear regime of compression (see section 7.7). In practice this means the TOF data counts were not compressed and suffered no loss of quality.

(c.f. SNG/ELS where the onboard two-byte unsigned integer is compressed to a one-byte integer for transmission to Earth, a standard technique used by many spacecraft instruments of all types.)

10.8 Does TOF have a background that requires removal?

There are three types of background to be aware of in CAPS TOF: penetrating radiation, high flux, and the LEF echo "ghost peaks" (in ST data):

10.8.1 Penetrating Radiation

Co-Incidence data (such as TOF) requires one ion to create two signals (start and stop) in the right order and within a certain time frame of each other before the instrument registers a valid count. As such the risk of penetrating radiation causing a background count is vastly reduced, as it's very unlikely a penetrating particle happens to be heading in the correct direction to hit both the start and stop anodes in that order. For this reason TOF data are largely background free from penetrating radiation… until you are very close to Saturn (<4 R_S). Inside of this there is so much penetrating radiation that they begin to team up – one penetrating radiation particle initiates a start signal, while an utterly separate one just happens to initiate a stop signal within the time window, and the sensor measures that as a count. The sheer amount of background this can cause often overwhelms any signal in the data.

10.8.2 High flux

This is somewhat similar to the case of overwhelming penetrating radiation. Most ions are correctly measured in TOF. But there are so many incident ions that sometimes two of the same ion species team up by chance, one creates a start signal and a different one a stop signal, meaning the TOF channel number registered is utterly meaningless. This can create an enhanced band of counts at all TOF channels at energies matching that of the domination ion species, which are clearly wrong.

This effect can occur for any high count-rate ion species, and could occur in several simultaneously.

In particular, since ION data are derived from TOF data, then for inner magnetosphere data the water group ION product often shows a proton bump on top of the water group, and the proton ION product often shows a water group peak too, albeit at far few counts.

10.8.3 LEF echo in Ghost peaks in ST data

This is an instrument feature, where a vertical enhancement in ST data is seen that is caused by the LEF echo. As it is the only vertical feature in the spectra it is easy to identify.

The so-called "ghost peaks" are not a true background, but rather an instrument effect arising from the fact that the carbon foil at the entrance to the TOF section has the unintended effect of scattering the ions passing through the foil. The consequence of this is that rather than the ions only striking the ST stop anode, they can also hit other surfaces within the instrument interior. Depending on where these ions hit, secondary electrons can be liberated which are then accelerated to and detected by the stop anode. Since the time-of-flight of the electrons is negligible compared to that of the ions, the resulting TOF measurement reflects the travel time of the ion to an unintended surface. Since these TOF measurements tend to cluster in certain time intervals due to the geometry of the instrument interior, broad peaks appear in the TOF spectra referred to as ghost peaks.

A special case of a ghost peak is the "LEF echo", which occurs when a positive ion exiting the foil hits the LEF detector and liberates a secondary electron from the detector surface that is then accelerated down to the ST detector. Since the TOF of the positive ion is energy-independent, the LEF echo always appears at the same channel. The most prominent LEF echo is from H^+ striking the LEF detector. This appears at a ST TOF channel of ~275.

10.9 TOF Channel Range is not fixed.

The TOF instrument has 2048 different TOF channels where data can fall, but only a subset are returned to Earth. At most 512 TOF channels are returned, but telemetry of 256 or 128 channels is also possible. Since the TOF binary files are fixed length, if 256 channels are present then they are spaced at every second value with fill values in between. If 128 channels are present then they are spaced at every fourth value with fill values in between.

To describe this there are two variables in the binary records:
ST_START_CHANNEL and ST_INTERVAL (for ST data)
or
LEF_START_CHANNEL and LEF_INTERVAL (for LEF data)

For a case where all 512 TOF channel bins have data (no MISSING_CONSTANTS at all) then the axis would have 512 tick marks which would be:

xticks = ST_START_CHANNEL + [0:511]*ST_INTERVAL

where [0:511] is an array of 512 elements that are 0, 1, 2, 3,…, 509, 510, 511. However if every 2nd TOF channel bin is a MISSING_CONSTANT value, then there are 256 data points:

xticks = ST_START_CHANNEL + [0:2:510] *ST_INTERVAL

where [0:2:510] is an array of 256 elements from 0 to 510 in steps of 2; i.e. 0, 2, 4, 6,…, 506, 508, 510.

And if three of every four TOF channel bins are MISSING_CONSTANT values, then there are 128 data points:

xticks = ST_START_CHANNEL + [0:4:508] *ST_INTERVAL

where [0:4:508] is an array of 128 elements from 0 to 508 in steps of 4; i.e. 0, 4, 8,…, 500, 504, 508.

Even if 512 channels are present in the TOF data product, they do not necessarily correspond to the same subset of 2048 channels. From SOI up until 2005-058T01:00:00, the 512 channels corresponded to every fourth channel of the full 2048, for both ST and LEF. Thus the full TOF range was evenly sampled. However, it became apparent upon analysis of the first few months of TOF data from Saturn, that much of the TOF range was never populated by counts. Thus the channel selection was modified after 2005-058T01:00:00 to better utilize telemetry such that the 512 ST channels corresponded to every other TOF channel (in 2048 space) starting with channel 40 and continuing up to channel 1062. The 512 LEF channels correspond to every TOF channel between channels 650 and 1161. Though in general this is the binning arrangement, one should be wary of exceptions and always check ST_START_CHANNEL and ST_INTERVAL (or the LEF counterparts).

10.10 TOF data needs to be corrected for time bin variation

Buried in the TOF time digitization electronics is a circuit that parses the TOF interval into individual channels. This sort of sub-nanosecond timing circuitry, at least when CAPS was built in the 1990s, was very state-of-the-art; however, it suffers from a timing artifact that causes individual TOF channels not to all cover the same time interval. Nominally, each channel is 750 picoseconds wide, but this can

vary by as much as ±20%. Thus when you plot a raw TOF spectrum, artificial spikes and dips appear from channel to channel that are not real. They reflect the fact that wider channels will have more counts on average, and narrower channels will have fewer counts. Unfortunately, the degree of variation is not the same throughout the mission, so a single correction pattern cannot be applied for all spectra. The good news is that the pattern repeats every 16 channels (in 2048 space), and can be corrected reasonably well.

No ideal correction method has been agreed upon, and so the data in the PDS is left uncorrected. The method adopted by the CAPS team is to take every *n* TOF channels (where *n* = 16/ST_INTERVAL or 16/LEF_INTERVAL) and sum them into a single *n*-long array. Although it isn't perfect, this tends to average over the real variation in signal, and leave the fixed pattern. The array is then normalized and divided into every *n* channels of the data to remove the time variation pattern.

10.11 Identifying ion species in TOF spectrograms

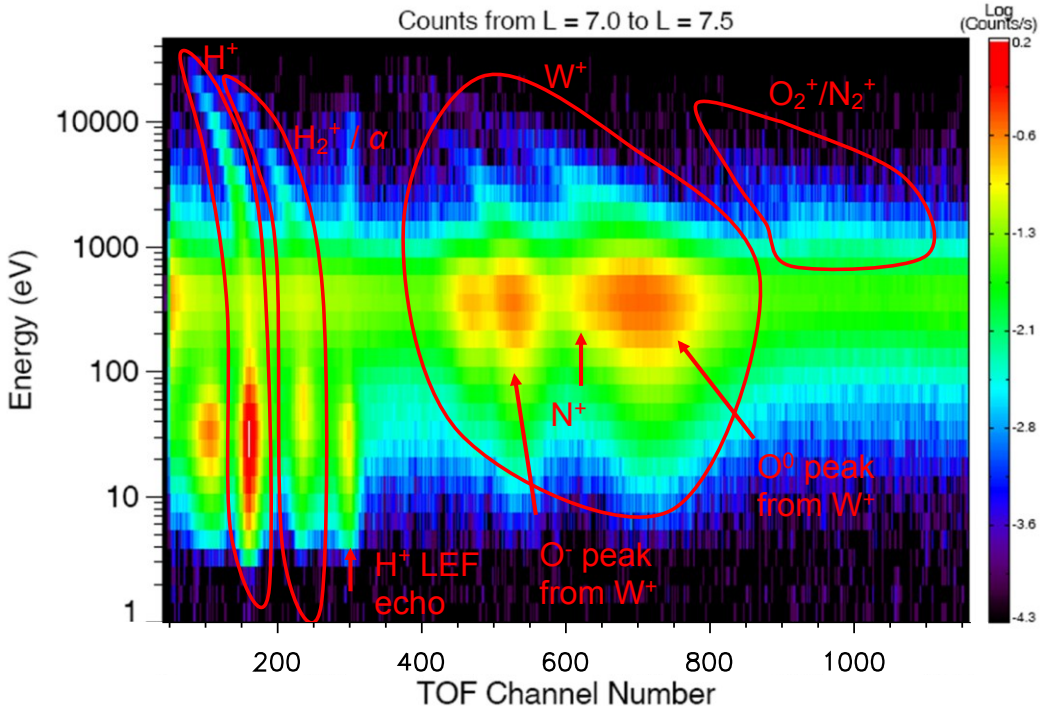

Figure 10.1: A typical magnetospheric TOF spectrogram.

10.11.1 DATA_ST

For a given energy, the TOF peak for a given species is not confined to a specific channel number, but is distributed across a number of channels. The heavier the mass, the broader the peak. The TOF peak will have a maximum at a central TOF channel, which will depend on energy. Figure 10.1 is an energy vs. TOF channel spectrogram. Particular ion species follow a contour in the spectrogram. These contours are generally at a fix channel number at low energies then curve to lower TOF channels at higher energies. There are a number of separate equations to indicate the central TOF channel number. If the exit particle is neutral, then for a given incident mass M (in amu) and charge Q, the contour is a function of incident energy E (in eV, usually the center value of the energy step used) and is given by:

TOF Channel = 36685 * M/Q/(E+14557) + 18752 * sqrt(M/Q/(E+14557)) - 34.647

Note that here, and in the formulas to follow, "TOF channel" refers to the channel number in 2048 space. This should not be confused with the *index* of the TOF channel (e.g. 0 – 511) associated with a given TOF data product. You will need to use the method described in section 1.9 to convert the index into a proper TOF channel, using the appropriate values of ST_START_CHANNEL and ST_INTERVAL.

The previous formula does not work well for the light(est) ions. For a M/Q of 1 or 2 the following in-flight calibrated values are better:

For H^+ (M/Q = 1):
TOF channel = 11462*sqrt(1/(E+14557))
And for H_2^+ or He^{++} (M/Q = 2):
TOF channel = 10859*sqrt(2/(E+14557))

If the exit atomic fragment is hydrogen, carbon or oxygen, there is a finite probability that the exit charge state is negative. For carbon, this probability is very low (~1%). For hydrogen and oxygen, the probability is considerably higher (~5% and ~20%, respectively). For these cases, the central TOF channel is given by:

H^- TOF channel = 893.44 * (sqrt(abs(M/Q) / (E + 14557))) ^ 0.42579
C^- TOF channel = 7507.1 * (sqrt(abs(M/Q) / (E + 14557))) ^ 0.8242
O^- TOF channel = 8009 * (sqrt(abs(M/Q) / (E + 14557))) ^ 0.83178

[Despite the charge being -1 for H^-, C^- or O^- you should use Q = +1, hence the abs to avoid confusion. While (sqrt(…))^n seems an odd way to write formula, this format is shown to keep it similar to earlier equations, but (…)^(n/2) is equivalent.]

Note that although all these formulas formally depend on the incident ion charge Q, for practical purposes, Q = +1 for everything except H_2^+ or He^{++}.

10.11.2 DATA_LEF

For the LEF data, the TOF peaks are also distributed, but the spread is much narrower. As with the ST data, the central TOF channel can be related to mass and energy by a formula:

TOF channel = 278 * sqrt(Mf) – 201 * sqrt(Mf) * exp(-0.00029 * Ef),

where Mf (in AMU) is the mass of the *exit fragment* (not the incident mass), and Ef (in eV) is the incident energy at the foil partitioned to the exit fragment.
For example, Ef for O^+ from H_2O^+ is (16/18)*(E + 14557) eV, or if E = 1000 eV, Ef = 13828 eV.

The dependence of the formula on the charge is left out, since LEF counts are only possible for ions incident as singly-charged. Multiply-charged ions are given so much energy by the post acceleration field between the exit of the ESA and the carbon foil, that the exit fragment cannot be reflected in the LEF field.

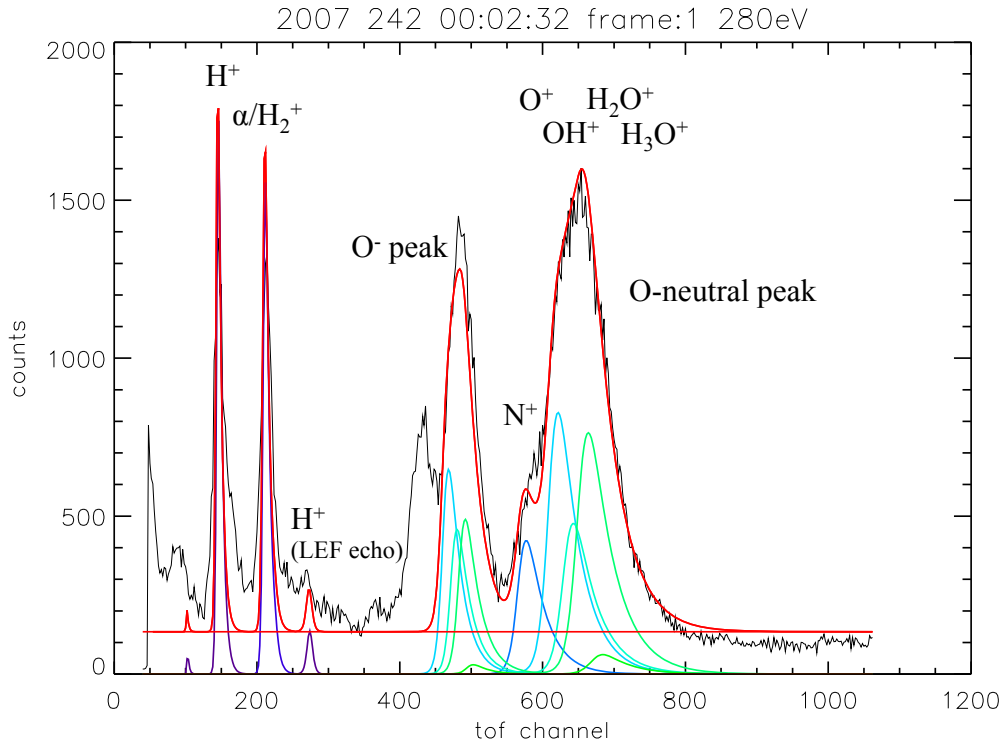

Figure 10.2: Example ST TOF spectrum for a single energy step.
The black trace is the raw TOF data, and the colored curves are least-squared fits to the data using model functions for the various species. The sum of all model consituents is given by the red line. Note there are two W^+ peaks. The right peak is due to oxygen from W^+ that exits the foil as a neutral; the left is from oxygen that exists the foil as a negative ion. The peak to the left of the O^- peak which is not included in the fit is due to TOF starts from postfoil H^- ions rather than from electrons.

10.12 How to remove a background from the TOF spectra

Accurate removal of the background described in section 1.8 is a tricky business, which in principle requires an accurate modeling of the scattering mechanism. However, a reasonable job can be done by subtracting a flat background from the intended TOF peaks. The flat background is determined by averaging the count rates in areas where no TOF peak is expected (see Figure 10.2). In practice, it has been found that using two flat backgrounds, one for the light species (H^+, H_2^+ and He^{++}), and one for the heavy species.

10.13 Turning TOF counts into scientific units

The TOF counts can be converted into phase space density via:

$$f_{ij}(v) = \frac{c_{ij}}{G_j v^4 \varepsilon_{ij} \tau}$$

where $f_{ij}(v)$ is the phase space density for species i and energy step j, c_{ij} is the number of counts associated with species i, G_j is the ESA geometric factor, v is the velocity of the incident particles, ε_{ij} is the TOF detection efficiency (note that it depends on species *and* energy), and τ is the integration time.

The integration time is equal to the amount of time in a given B-cycle that the ESA dwells at a TOF energy step. For most TOF energy steps, this is twice the accumulation time of a single SNG energy step. The exception is TOF energy step 1, which is just a single SNG energy step (see section 10.4). Thus, with this exception in mind, the integration time is $3.5 \times n$ seconds, where n is 1, 2 or 3, depending on whether the B-cycle is 256 s, 512 s, or 1024 s in duration.

Note that care must be taken when choosing the geometric factor because the field-of-view for a given B-cycle is not the same for all B-cycles (see section 10.6), and may not include the entire ion distribution. A common procedure is to cull data for actuation ranges that include the full ion distribution.

10.14 How to Plot TOF data

Plotting a line slice of TOF data for a B-cycle at a particular energy step is trivial. The record already has the object for that energy step's data in an array of 1x512, which can be plotted immediately against the *xticks* value from section 10.9. Remember to remove fill values/missing constants before plotting.
(Figure 10.2 shows an example line slice plot.)

TOF spectrograms for a single B-cycle (all 32 energy steps) are also easily plotted. Each TOF record had a 1x512 array in the DATA_ST object. There are 32 TOF records per B-cycle, so grouping the 32 to make a data array of 32x512 is very easy. Plotting that as a spectrogram with the vertical axis as TOF energy step and the x-axis labeled as *xticks* from section 10.9 is then trivial.
(Figure 10.1 shows an example TOF DATA_ST spectrogram.)

The process is the same if you wish to plot a DATA_LEF spectrogram instead of DATA_ST, except you use the DATA_LEF objects instead (LEF_START_CHANNEL instead of ST_START_CHANNEL, LEF_INTERVAL instead of ST_INTERVAL, LEF_ENERGY_COLLAPSE instead of ST_ENERGY_COLLAPSE).

Your plotting code should make the following checks:

i) For a single B-cycle, all 32 records should have ST_START_CHANNEL numbers that are the same. Likewise for ST_INTERVAL, ST_ENERGY_COLLAPSE, LEF_START_CHANNEL, LEF_INTERVAL and LEF_ENERGY_COLLAPSE.

ii) If you start summing the DATA_ST from several B-cycles to increase your signal you need to make sure that their ST_START_CHANNEL's are all the same value in all those B-cycles. Also for ST_INTERVAL and ST_ENERGY_COLLAPSE. The same if true if for LEF data, if you sum DATA_LEF from several B-cycles the LEF_START_CHANNEL, LEF_INTERVAL and LEF_ENERGY_COLLAPSE must not vary.

iii) Also note that the MISSING_CONSTANT value in DATA_ST and DATA_LEF are different to the one listed in the U3 FMT files (by 1, but different). The MISSING_CONSTANT value from the FMT files is 4294967295, but in the DAT files, the value 4294967296. For safety you should check for both.

10.15 Things to know about the dataset

10.15.1 B-cycle TIMEs

As with all CAPS products, the A-cycle and B-cycle times reported are the start time of the A-cycle or B-cycle. As such A-cycle 4, for instance, should have the same time reported in the TIME object of SNG, ELS, ANC, ACT, etc. files. Likewise for B-cycles.

However TOF.TIME is usually 1s earlier than the TIME object from ANC, ACT, SNG, ION, ELS, IBS for the same B_CYCLE_NUMBER (by which we mean the time of the first A_CYCLE_NUMBER with the same B_CYCLE_NUMBER as the TOF B_CYCLE_NUMBER).

Since B-cycles last 100s of seconds this is of no consequence.

10.15.2 B-cycle Duration, 256, 512 or 1024s

The easiest way to tell it to compare the B_CYCLE_NUMER of the B-cycle in question with the ANC data files, and see how many A-cycles (32s each) that same B_CYCLE_NUMBER is listed under.

10.15.3 Telemetry Mode can change "within a B-cycle"

This is not obvious, as TOF is a higher-rate product, therefore is only available in higher telemetry modes.

The TOF files have a single TELEMETRY_MODE listed for each B-cycle, however if you compare the A-cycle resolution TELEMETRY_MODE values of the ANC files for the same B-cycle numbers you may see the telemetry mode change within the B-cycle.

As such you can only use the TOF.B_CYCLE_NUMBER as a guide.

i.e. TOF binary for DAYQ 200528400 for B_CYCLE_NUMBER = 5, lists a TELEMETRY_MODE of 2.

This is not obvious as mode 2 is one of the lowest and not one that supports TOF data.

However the ANC file for the same DAYQ has eight records (8*32s = 256s) where B_CYCLE_NUMBER = 5 (those are A_CYCLE_NUMBERs 657 to 664 inclusive) list TELEMETRY_MODEs for those 8 records as: 2, 2, 2, 2, 2, 2, 2, 16.

As such it is clear that at the end of this B-cycle accumulation period CAPS is in a telemetry mode that supports TOF data (16), hence it can downlink the B-cycle that just finished accumulating, despite most of the period being during a very low telemetry mode (that does not support TOF downlinks).

10.15.4 Comparing TOF data to SNG or ION data

If you wish to compare TOF data with SNG (or ION) data that was taken simultaneously you should use the TIME variables to align the data for comparison, not simply intervals when the TELEMETRY_MODE goes to a higher mode.

Since TOF is a higher telemetry mode product you might think that just pulling out the SNG (or ION) data when TELEMETRY_MODE goes higher is sufficient, however it is not. This is related to the above section 10.15.3, and is explained by the same example.
For DAYQ 200528400, the (start) TIME of B_CYCLE_NUMBER = 5 is 05:50:10, whereas the first A-CYCLE of the ANC file with B_CYCLE_NUMBER = 5 is A_CYCLE_NUMBER 657, which has a start time of 05:50:11. Essentially this is the same, but 1 second different (see section 10.15.1). So the B-cycle for TOF data was accumulating data from 05:50:10 to 05:54:26 (for 256s).

However had you simply pulled out the SNG data around then when TELEMETRY_MODE = 16, that does not start until 05:53:55 and lasts until 05:58:11 (also 256s).

So while the durations are the same, and they do overlap slightly, they are mostly different intervals. As such you should always use the TIME variables for comparison of data, and not telemetry mode.

11 ION Using the Data – General Knowledge

This document is focusing on SNG and ELS data, however the following information should provide a start to working with ION data.

11.1 What is the ION portion of IMS and why do I want it?

ION data takes the onboard TOF ion counts and partitions them to the different anodes for different ion species, such that the data product resembles SNG data for each ion species. As such all calibration values are those of TOF, and as yet no published calibrated ION data exists. However it has often been used empirically to aid identification of ion species in the SNG data (same energy ranges as the ION data) and to obtain the relative angular distribution of various species. Relative avoids the question of efficiencies, either as a function of species or energy, and the angular distribution by species is not obtainable from any other CAPS data product. To date there has been no systematic effort to calibrate the ION data and as such no ION commonly accepted calibration values exist.

11.2 When do we have ION data?

ION data products are built based on the measured time-of-flight, and as such are only telemetered to Earth during the higher rate telemetry modes. On average this is about 1 in every 6 intervals. ION data intervals overlap with B-cycle intervals.

11.3 What is the SAM_ION_NUMBER and Group Tables

See the CAPS instrument paper for how ION data works, but a quick overview is provided here.

The TOF spectra (energy vs. TOF channel) are divided up into different sections depending upon the Group Table used. For instance, the table may say that for a given energy, protons are expected to fall between TOF channels a and b, so sum up all the counts in TOF channels a to b to find the number of protons at that energy. However the table's location of the protons may miss the wings of the distribution if the table does not accurately represent the true location of the full distribution. Or if you're a to b region is too generous you may include different ion species. The tables have been updated as the mission progressed, but as such there and different tables during the mission.

Specific details on the ION Group Tables can be found in the CALIB directory of the PDS archive for the CAPS data.

The SAM_ION_NUMBER gives you the information you require to know which ion species was being collected (partitioned out) and which group table was used. It is a 5-digit number (if you only have 4 digits then add a leading zero to make it 5 digits, i.e. 1234 should be 01234). The naming convention for the SAM_ION_NUMBER is:

mass *1000 + charge(q) *100 + <2 digit number for group table>

As a user you do not really care about the group table but you do need to know if the number is odd of even.

Even values of group table are from the TOF ST spectra and odd values are from the TOF LEF spectra.

For example, during SOI (DAYQ 200418218) there were 7 different SAM number: 1108, 12108, 14108, 16108, 17108, 18108, 19108

All of these are from the TOF ST detector as they are even, and are mass:charge species of 1:1, 12:1, 14:1, 16:1, 17:1, 18:1, 19:1 respectively, all using a group table represented in 2-digits by 08.

By comparison, on DAYQ 201035318 there are just 3 different SAM numbers: 1116, 13116, 17116
These are for ion mass:charges of 1:1, 13:1 and 17:1 using a group table represented by 16. For this table, 17:1 really means any water group in the mass 16 to 19 amu range.

In later years the group tables were changed (see section 11.4.6) so that just one water group data set is returned, that is to say the group tables grouping was broad enough to encompass ions of mass 16 to 19 amu inclusive. However these are listed as mass 17 in the SAM_ION_NUMBER to keep the convention (see section 11.4.7).

11.4 Things to know about the dataset

11.4.1 Calibrations

The data set is based on TOF data; hence TOF calibrations values should be used.

11.4.2 ION data records and plotting

ION data resembles SNG data in format, so filter the ION data records for the particular SAM_ION_NUMBER you desire then treat plotting the data like SNG data.

Background removal should be done as for SNG, however as ION data are a co-incidence measurement (requiring start and stop signals in TOF) the background should be lower than for SNG data.

11.4.3 ION data vs. SNG data

11.4.3.1 ION has less counts than SNG

ION data will have less total counts than SNGs data, because ION data are a TOF product. Summing all the ION counts together for all ion species is still lower than the SNG counts at the corresponding time. This is mainly due to different efficiencies, but also that ION data returns subsets of the TOF distributions, not the whole of it.

11.4.3.2 Azimuth-Summing can be different

Azimuth-Summing may be the same or different during the same A-cycle in SNG and ION data. i.e. when telemetry mode = 32, SNG data are summing Azimuth-pairs, while ION data are summing Azimuth-quads. If you are trying to do a direct comparison between SNG and ION dataset and their azimuth summing method differs you may want to sum the SNG azimuths yourself to match the ION data.

11.4.3.3 ION data may sum anodes from 1999-2003

ION data never sums anodes in the Saturn mission (2004-) so this is never a concern with Saturn data, however it did on the cruise phase to Saturn from 1999-2003. The usefulness of that cruise phase ION data for science is very questionable though – please seek advice from the CAPS PI if you were to attempt to use it.

While SNG data may sum anodes, in telemetry mode 132 (see section 8.1.2), there is no ION data for that telemetry mode.

11.4.3.4 Fill values are different

While SNG uses the expected unsigned two-byte fill value of 65535, ION uses 28671 (as it is a signed two-byte value).

11.4.4 ION data vs. TOF data

ION is a TOF product, however they are usually offset slightly in time. This is because TOF data are being taken on every 4-second sweep, but most of that never leaves the spacecraft – only getting telemetered to Earth during higher telemetry modes. As such, TOF data are only during B-cycles (that is B_CYCLE_NUMBER is not 65535, the MISSING_CONSTANTS value), but ION data can be available out-side of B-cycles (i.e. when B_CYCLE_NUMBER=65535), but there is always a nearby B-cycle.

This is a consequence of how the TOF data are collected on the spacecraft and returned to Earth.

In practice it means that if you take all the ION data in a B-cycle and add it up you cannot recreate the TOF data with the same B-cycle number, as that was taken over a slightly different, but overlapping, interval.
It is also worth noting that the reported TOF times are often 1 second less that the times reported by ION data (or ANC, ACT, ELS, SNG) for the start of an A or B cycle.

11.4.5 Ghost Peaks of Other Ion Species

Just as with TOF data, there may be a 'High Flux background' (see section 10.8.2) that results in a proportion of ions being mis-identified. For instance if looking at the ION data for a water group partitioned species there may still be a secondary peak for hydrogen, and vice versa in the hydrogen partitioned species.

11.4.6 ION data can have negative counts

Early in the mission (prior to late 2004) it was possible to have some DATA records with negative counts (which is why the DATA object is a signed integer rather than unsigned). Clearly this is unphysical, but is a consequence of the way the data was telemetered (particularly when the deconvolution is applied to noisy data.)

In late 2004 more robust group tables were uploaded that do not produce negative counts. The group table was #6019, shortened to 19 for the SAM number (see section 11.3). As such all ION data with SAM numbers that are ***19 only have positive counts, and if the SAM number's last two digits are less than 19 be aware of the potential for negative counts.

The reason for the negative counts follows, pasted from an email:
If you know the fraction of counts from a given species end up in which bin, you can relate counts from in a given bin to counts from a given species with a system of linear equations. For example, with two species and bins,
*C(Bin1) = A(0,0) * C(H^+) + A(1,0) * C(W^+)*
*C(Bin2) = A(0,1) * C(H^+) + A(1,1) * C(W^+)*
Invert the A matrix, upload it as a table, and you can have the software turn counts per bin into counts per species. You might say you can do the same thing on the ground, but there is an advantage to doing it on spacecraft. You could use 16 bins and species and only send down two of them due to limited data volume. That would give a better deconvolution than you could do on the ground with two bins.

In practice, we found that can get very strange and uninterpretable results if the A matrix isn't exactly right. During SOI for example, the bin centers are off by (if memory serves) 5% and the fractions of species per bin was a little off. (The matrix was based on post-launch tests of the prototype, since there wasn't enough time to calibrate the Flight Model, FM, on very many species and energies; it turned out the prototype and FM were systematically different.) You can't back out the original counts per bin, since we don't send down all the species. After SOI we revised the bin intervals and set the inversion to an identity matrix (so, in practice, it is sending down the bins, even though the software is doing a bunch of multiply by 1, multiply by 0 and sum.)

11.4.7 SAM_ION_NUMBER 171** could be W^+

See the end of section 11.3, but although the SAM_ION_NUMBER suggests the ion species is OH^+ (m:q of 17:1) it could be W^+, that is all of O^+, OH^+, H_20^+, H_30^+. This only applies to 171** values and 161** does mean just O^+. etc..

12 IBS Using the Data

IBS data was not a target of this document, however this document has shown you how to read in the raw binary data.

To aid a user in using IBS data for science, the following 5 pages are a direct copy of a document written by John T. Steinberg (LANL) on details of using the IBS data with only a few formatting changes. Remember that IBS anode 2 has the same field of view as anodes 1 to 8 combined of SNG or ELS. However anodes 1 and 3 and at slanted angles to anode 2, so further conversions will be required to identify their field of view. See the CAPS instrument paper for further details.

A further complication is that the efficiency of the detector depends upon the incident angle of the ion, so as part of your calculation you must infer that angle using the data from several anodes and using the cross fan geometry of IBS to your advantage. However that is beyond the scope of this document.

Calibrated Fluxes from CAPS IBS

April 29, 2005
John T. Steinberg
Los Alamos National Laboratory

12.1 General

The CAPS Ion Beam Spectrometer, or IBS, is a hemispherical section curved plate electrostatic analyzer. It has three rectangular entrance apertures and three channel electron multiplier (CEM) detectors. The CEMs are each mounted 180° away from an aperture. This configuration produces an instrument with 3 detectors, each having a roughly planar fan-shaped field of view. The fans are approximately 150° x 3°, and are arranged so that the fan-plane of each detector intersects the fan-planes of the other two. As the CAPS actuator swings, the fan fields of view are swept back and forth across any streaming charged particle beams. The times and corresponding actuator angles at which the peak particle flux is observed in each of the fans can be used to resolve the particle flow direction.

12.2 IBS counts

IBS data can be arranged into arrays of counts N_{raw} as a function of fan 1, 2, or 3, particle energy-per-charge e/q, and measurement time t as follows. Voltage applied to the IBS electrostatic analyzer plates is varied in a stepped sequence, effectively sweeping the detector response through a set of discrete energy-per-charge levels. While the detector dwells at each energy-per-charge level, counts are accumulated over a fixed integration time τ (see table) simultaneously in each of 3 detectors. Raw counts N_{raw}(fan,e/q,t) need first to be dead-time-corrected.

$$N_{DTCorr} = \left(\frac{\frac{N_{raw}}{\tau}}{1 - \frac{N_{raw}}{\tau}\tau_D} \right) \tau$$

In the above expression τ_D is the IBS dead time (see table) and N_{DTCorr} are dead-time corrected counts. Each IBS detector records counts from particles that directly impact that detector. In addition, each IBS detector responds occasionally to particles that impact one of the other IBS detectors. We refer to this affect as cross-talk, and a correction can be applied to minimize the affect of cross-talk. To apply the correction, take the counts in each of the 3 fans accumulated at the same time t_i and energy/charge step e_i/q, and treat as a simple 3-component vector. Multiply that vector by the cross-talk-correction matrix **β** (see table**)** as shown below.

$$\begin{pmatrix} N(fan1, e_i/q, t_i) \\ N(fan2, e_i/q, t_i) \\ N(fan3, e_i/q, t_i) \end{pmatrix} = \begin{pmatrix} \beta_{11} & \beta_{12} & \beta_{13} \\ \beta_{21} & \beta_{22} & \beta_{23} \\ \beta_{31} & \beta_{32} & \beta_{33} \end{pmatrix} \begin{pmatrix} N_{DTCorr}(fan1, e_i/q, t_i) \\ N_{DTCorr}(fan2, e_i/q, t_i) \\ N_{DTCorr}(fan3, e_i/q, t_i) \end{pmatrix}$$

12.3 IBS Particle Flux

The IBS detectors were designed to measure directed beams of charged particles rather than particles with distributions that are quasi-isotropic. In the following discussion, we assume that the particle distribution producing counts in the IBS detectors is a convecting or beam distribution, and that the thermal width of the particle beam is very much smaller than 150°, roughly the field of view in the "plane of a fan". We allow the thermal width of the beam to extend beyond the field of view in the direction perpendicular to the plane of a fan.

With counts corrected for dead-time and cross-talk, particle flux J ($cm^{-2}sec^{-1}$) can be determined. Were an IBS fan response flat across the field of view, calculating flux would be more straightforward. However, the effective response area A varies with the angle of beam entry into the fan. In general, determining the particle flux requires knowledge of the particle beam direction. Without determining the beam direction, we can get a particle flux lower limit. The effective response area is maximum A_0 (table) near the center of the fan field of view (i.e. beam entry normal to an analyzer aperture). The lower-limit particle flux J_0 in a given fan at time t_i and energy-per-charge step e_i/q is given simply by:

$$J_O = \frac{N(fan_i, e_i/q, t_i)}{A_O \tau} \quad \left(cm^{-2}\sec^{-1}\right)$$

This lower-limit flux J_0 is the same as the actual flux J in the case where a beam enters near the center of a fan. When a beam enters a fan at an angle θ away from fan center, the flux is found as follows:

$$J = \frac{N(fan_i, e_i/q, t_i)}{w(\theta) A_O \tau} \quad \left(cm^{-2}\sec^{-1}\right)$$

In the above expression w(θ) is a weighting function and θ is the angle of a beam relative to the center of a given fan. A lookup table of w(θ) values is provided.

12.4 Beam Flow Direction:

In order to find the flow direction, IBS requires that the CAPS actuator is sweeping the fans' fields of view across a charged particle beam. The flow direction can be determined if at least 2 of the 3 fans detect a peak flux as a function of actuator sweep angle. The procedure is as follows. First, one must determine α_1, α_2, and α_3 the actuator angles for which a peak flux was detected in fans 1, 2, and 3 respectively as the actuator sweeps across the incoming beam. We define the following vectors, which are normal to the IBS fan-planes at the time the beam was detected by each fan.

$$\hat{n}_1 = (-\cos(30^\circ)\cos(\alpha_1), \cos(30^\circ)\sin(\alpha_1), -\sin(30^\circ))$$
$$\hat{n}_2 = (\cos(\alpha_2), \sin(\alpha_2), 0)$$
$$\hat{n}_3 = (-\cos(-30^\circ)\cos(\alpha_3), \cos(-30^\circ)\sin(\alpha_3), -\sin(-30^\circ))$$

If only 2 fans detect the beam, then only 2 of the above vectors can be evaluated. The flow direction is found in turn by taking cross products between pairs of these vectors.

$$\hat{d}_{12} = \hat{n}_1 \times \hat{n}_2 / |\hat{n}_1 \times \hat{n}_2|$$
$$\hat{d}_{23} = \hat{n}_2 \times \hat{n}_3 / |\hat{n}_2 \times \hat{n}_3|$$
$$\hat{d}_{13} = \hat{n}_1 \times \hat{n}_3 / |\hat{n}_1 \times \hat{n}_3|$$

If all three of the IBS fans detected a beam, then the flow direction can be evaluated using each of the 3 expressions shown above. The results for all 3 should be similar. If only two of the IBS fans detect the beam, then the flow direction can only be found by 1 of the 3 expressions above. Once the flow direction is found, then the angle θ_i of beam into a detector can be found as follows.

$$\theta_1 = \arccos\left[(\sin(\alpha_1), \cos(\alpha_1), 0) \bullet \hat{d}\right]\left(\frac{-d_z}{|d_z|}\right)$$
$$\theta_2 = \arccos\left[(\sin(\alpha_2), \cos(\alpha_2), 0) \bullet \hat{d}\right]\left(\frac{-d_z}{|d_z|}\right)$$
$$\theta_3 = \arccos\left[(\sin(\alpha_3), \cos(\alpha_3), 0) \bullet \hat{d}\right]\left(-\frac{d_z}{|d_z|}\right)$$

The values of θ_I so determined can be used to find the effective area weighting function $w(\theta)$ from the table.

12.5 TABLE: Key Calibration Parameters

April 29,2005

Integration time τ 0.0068359375 seconds
Effective area A_O 0.34 cm^2
Dead time τ_D 0.86 microseconds

Cross talk correction matrix (unit-less)

β_{11}= 1.02575
β_{12}= -0.143420
β_{13}= -0.0385874
β_{21}= -0.135635,
β_{22}= 1.02600
β_{23}= -0.0406647
β_{31}= -0.136521
β_{32}= -0.135645
β_{33}= 1.01217

NOTE: the cross talk correction matrix provided here is valid for counts below 300. Above 300 counts its reliability is uncertain. We believe that a more sophisticated cross-talk correction will be appropriate for IBS counts greater than 300, but it remains to be developed.

Effective area weighting factor (unit-less)

θ	w(θ)	θ	w(θ)	θ	w(θ)
-75.0	0.00000	-24.0	0.29462	27.0	0.12677
-72.0	0.00425	-21.0	0.36969	30.0	0.07649
-69.0	0.01558	-18.0	0.47380	33.0	0.04249
-66.0	0.02974	-15.0	0.60340	36.0	0.02762
-63.0	0.05595	-12.0	0.74717	39.0	0.02266
-60.0	0.08428	-9.0	0.86331	42.0	0.02974
-57.0	0.10694	-6.0	0.92564	45.0	0.02479
-54.0	0.12465	-3.0	1.00000	48.0	0.04249
-51.0	0.11331	0.0	0.97026	51.0	0.04887
-48.0	0.11261	3.0	0.97167	54.0	0.04603
-45.0	0.10340	6.0	0.95821	57.0	0.06020
-42.0	0.10836	9.0	0.81161	60.0	0.06657
-39.0	0.14731	12.0	0.62606	63.0	0.06586
-36.0	0.13314	15.0	0.49221	66.0	0.06374
-33.0	0.16360	18.0	0.36827	69.0	0.04249
-30.0	0.17847	21.0	0.23938	72.0	0.03895
-27.0	0.23371	24.0	0.17422	75.0	0.02550

13 Attitude/Orientation from ANC and ACT files

13.1 CAPS Anodes' Field of View

The field of view (FOV) of any CAPS sensor is dependent on both the spacecraft orientation (found from either the ANC binary files or SPICE) and also the ACT binary file (there is no actuator information in SPICE). Both spacecraft orientation and actuator angle MUST be utilized.

Let's call the main spacecraft axis the spacecraft pole vector. The ELS, SNG and ION anodes are all pointed along the same spacecraft azimuth (i.e. longitude, atan2(y,x)), while the 8 anodes are spread out in the spacecraft polar direction (i.e. latitude, asin(z/r)). As such the anodes are fixed in the spacecraft polar frame, irrespective of actuator angle.

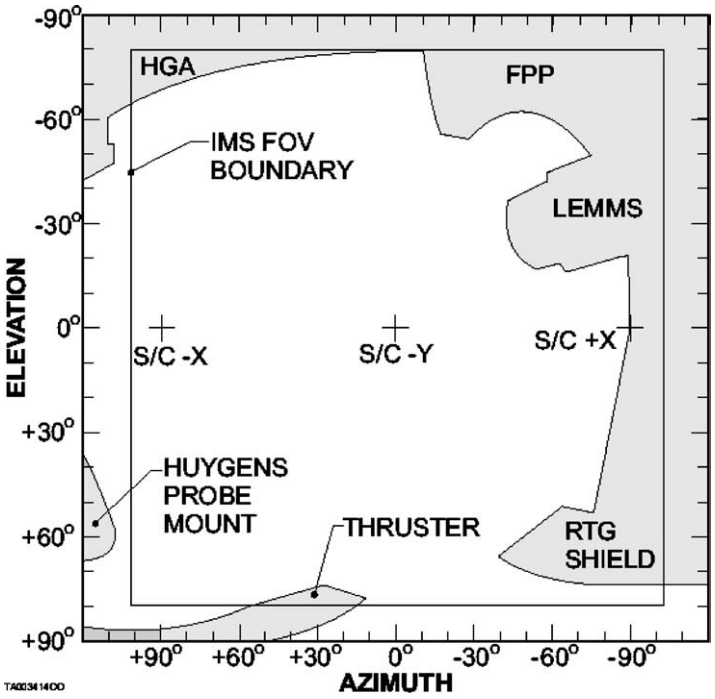

Figure 13.1: CAPS field-of-view.
A direct copy of figure 5 from the CAPS instrument paper (*Young et al.* 2004).

The actuator itself provide azimuthal motion in the spacecraft frame, with the entrance slits to ELS and IMS (SNG/ION) lined up, such that anode 1 of ELS matches the FOV of anode 1 of SNG (or ION) at every instance.

The IBS anode 2 has the same instantaneous FOV as the total sum of the FOVs of anodes 1 through 8 of ELS (or SNG or ION). However IBS's cross-fan geometry means that IBS anodes 1 and 3 do not align with any others.

This however is just the FOV in the spacecraft frame. To get a more useful FOV related to Saturn or a moon requires knowing where the planet/moon is at that given time and applying the appropriate coordinate transformation. To turn that into a magnetic field coordinate system (parallel and two perpendiculars) also requires knowing the magnetic field vector at that instance in a suitable frame (you may have to convert planet centered RTP B-field measurement into a spacecraft centered x, y, z co-ordinate or such first).

13.2 Pointing info (inc. ACT), orientation

Cassini is a 3-axis stabilized spacecraft and rarely spins (rolls), so any azimuth variation (in the spacecraft frame) is caused by the motion of the actuator. The different anodes provide the polar variation of viewing for SNG and ELS.

The segment of space sampled by the instrument is dependent on both the actuator motion and the orientation of the spacecraft, and whether the spacecraft is re-orientating to a new direction (or even rolling) during the data record.

Spacecraft orientation in J2000 co-ordinates may be found from the 9 orientation values in the **ANC** binary files of *SC_ORIENT_XX*, *SC_ORIENT_XY*, *SC_ORIENT_XZ*, *SC_ORIENT_YX*, *SC_ORIENT_YY*, *SC_ORIENT_YZ*, *SC_ORIENT_ZX*, *SC_ORIENT_ZY* and *SC_ORIENT_ZZ*.

At a first level, if these nine values do not change significantly over a time period then the spacecraft pointing may be considered to be stable, not rolling.

13.3 Azimuth Angle Coverage: ACT files

The actuator may sweep through up to 184 degrees (extremes of -80 to +104 degrees in spacecraft azimuth) of the full 360 degrees around the spacecraft's main axis, although usually sweeps through a smaller range than this. It rotates at a speed of 1 degree per second, however slows down within the 24 degrees of its extremes as it changes direction, resulting in one full actuation cycle having a typical duration of ~6 minutes, depending on the range being scanned.

The actuator values may be found in the **ACT** binary file, where each record has a single time stamp in the *TIME* variable (seconds from J2000, Barycentric dynamic time) which is the start time of the A-cycle, and a *DATA* variable that has 32 values. The first of the 32 values occurs at *TIME* + 0 s, the second occurs at *TIME* + 1 s, the third at *TIME* + 2s, ... and the 32nd at *TIME* + 31 s. Whether all 32 values are populated with a real number depends on Telemetry Mode (which is not a variable in the **ACT** binary, use **ANC**), if not fill values are used. It is common to only have the 1st, 9th, 17th and 25th value as real numbers and the other 28 values as fill. The actual

fill value is given in the **ACT** FMT file, where for the *DATA* variable has a MISSING_CONSTANT value specified as -999.

[Note: the telemetry mode may be found from the **ANC** binary *TELEMETRY_MODE* variable, or the **SNG** or **ELS** binary *TELEMETRY_MODE* variable. However the **ANC** binary is usually the better option to use as **SNG/ELS** data are not always available (i.e. if IMS/ELS is not turned on) and occasionally misses an A-cycle, whereas the **ANC** files do not have these issues. Of course if SNG has no data you can't analyze it anyway.)]

13.3.1 How to fill in gaps in ACT data

The intervals with fill values (MISSING_CONSTANT) may be filled in by linear interpolation of the nearest non-fill neighbors. This is perfect during the 1 deg/s phase, but near the ends of the actuator range errors are introduce as it slows down to change direction. The plot below shows the difference between true value (black) and straight lines joining up points 8s apart (colors). This shows that the worst error is only 1/3 degree right at the actuators extreme. Using a running quadratic-fit to the data will be more precise, however a linear fit is usually good enough, except at the end of an ACT file

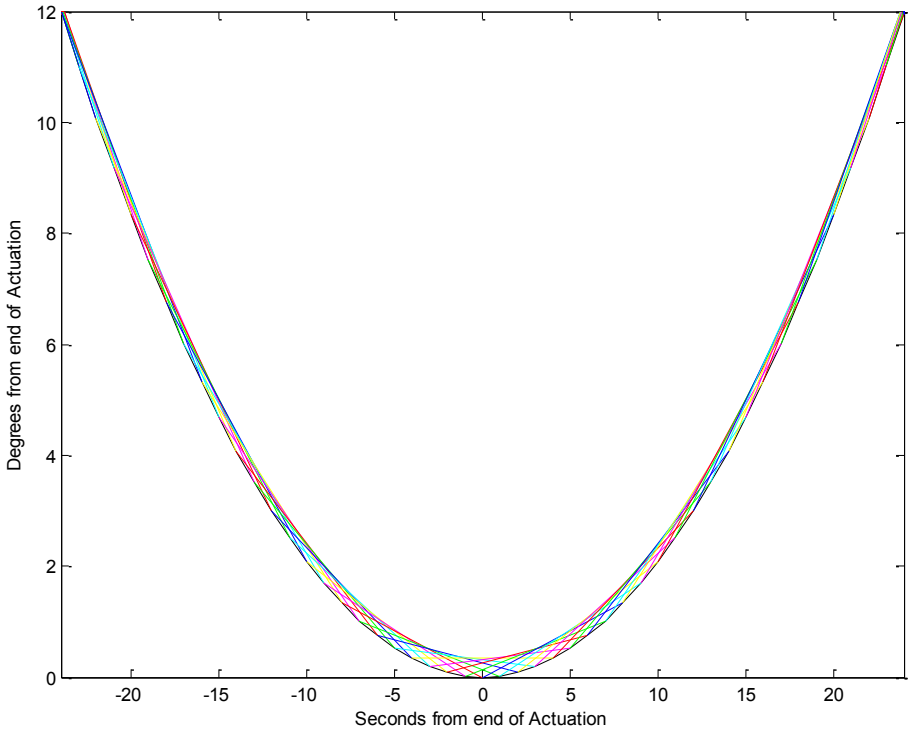

Figure 13.2: Filling in missing ACT data.

13.3.2 Last A-cycle of an ACT file

If the last ACT A-cycle is fully populated there is no need for any interpolation. However if it is every 8ths, then the final ACT.DATA record will be:
V, F, F, F, F, F, F, F, V, F, F, F, F, F, F, F, V, F, F, F, F, F, F, F, V, F, F, F, F, F, F, F
Where V is a real value and F is a fill value (MISSING_CONSTANT).

Linearly interpolating this is justified until the last 8 where you have 7 unknown values at the end.
In this case doing a quadratic fit of the 4 good values you have over all 32 values is the best idea. Alternatively, load in the next DAYQ ACT file so you no longer have an end of file boundary and do have a real value to bracket these fills.

13.3.3 Filling in missing Actuator values the proper way

The following is extracted verbatim from emails from the CAPS ops team. The linear or even quadratic interpolation is good enough for most people, but should you want to be really thorough you may use the following, explained as IDL code.

```
For the "positive" edge of FOV (ie. +104), you can get values by taking
104-acceldecel.  For the "negative" edge of the FOV, you can get values
by taking acceldecel+(-80).  For a +76 to +104 range, you would get the
"corner" on the top edge with 104-acceldecel and for the bottom corner
you would use acceldecel+(+76).  If you are on the "negative" side with
the upper range, you would use (-12)-acceldecel.

AccelDecel_t=[32,31,30,29,28,27,26,25,24,23,22,21,20,19,18,17,16,15,14,
13,12,11,10,9,8,7,6,5,4,3,2,1,0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,
17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32]
AccelDecel=fltarr(n_elements(AccelDecel_t))
for i=0,n_elements(AccelDecel_t)-1 do begin
    if AccelDecel_t[i] LE 24 then $
       AccelDecel[i]=0.5*(1/24.)*AccelDecel_t[i]^2 $
    else $
       AccelDecel[i]= 12+(AccelDecel_t[i]-24)
endfor
```

13.3.4 ACR files

You should not see ACR files at all, these are not provided to the PDS. These are uncalibrated ACT files (i.e. not corrected for homing runs and the like), but otherwise have the same format as ACT files – should you find yourself with one. But ALWAYS use ACT files in preference over ACR files.

13.3.5 Homing runs

The actuator keeps track of its angle by dead-reckoning, and over time the angle it predicts will drift away from its actual angle. To fix this 'homing runs' are carried out where the actuator is slowly told to keep moving up, and up, and up, and up, as far as it will go, until it will go no further. Since the operations team know physically what this upper limit is, they can they use the homing run results to correct for any drift. This correction has already been applied to the data within the ACT files.

It is mentioned as the homing runs are easy to identify in the data, and when homing the actuator is not sweeping back and forth. This makes the ion data taken during the homing run to be of limited use for science, however the electron data (where isotropic assumptions in analysis are often applied) is unaffected and still useful. That said, some people find this fixed viewing angle very useful for studies with IMS data too.

An example of one such homing run is from 2005-10-12T20:00:00 to 2005-10-13T00:00:00 and is shown below. The actuator was in sweeping back and forth, then slowly starts climbing until it hits the top and flat-lines. If you see this shape in the ACT data you know it's a homing run for calibration.

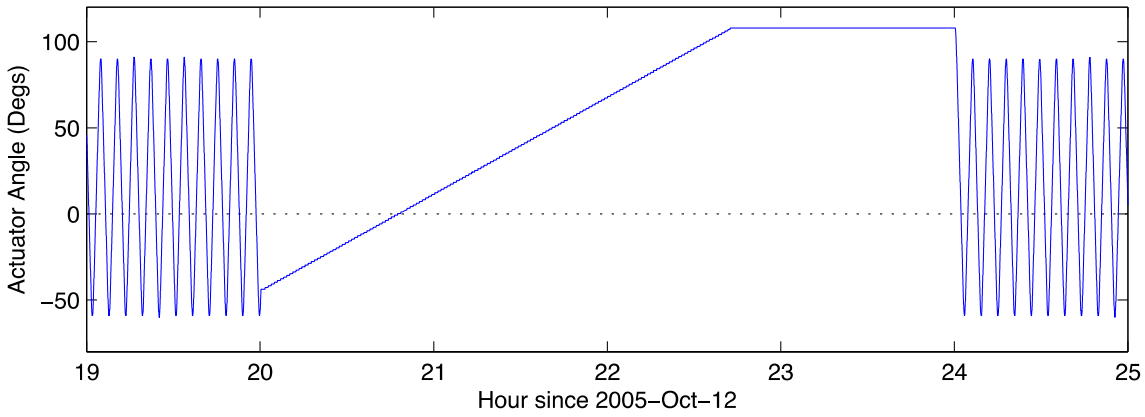

Figure 13.3: Example of an ACT homing run.

13.4 Polar Angle Coverage: Anode positions

The IMS sensor has 8 anodes, each with Angular Resolution of 8.3° x 20° (AZ x EL)$_{FWHM}$, such that they cover -80 to +80 degrees in spacecraft Elevation (aka Polar).

Similarly ELS has 8 anodes, each with Angular Resolution of 5.2° x 20° (AZ x EL)$_{FWHM}$, such that they also cover -80 to +80 degrees in spacecraft Elevation (aka Polar).

The 160-degree spread in polar for ELS, SNG and ION data are separated as follows:

Anode	Elevation range (degrees)	Center Elevation (degrees)
1	+60 to +80	+70
2	+40 to +60	+50
3	+20 to +40	+30
4	+00 to +20	+10
5	-20 to +00	-10
6	-40 to -20	-30
7	-60 to -40	-50
8	-80 to -60	-70

Data from all 8 anodes is always measured, however it is possible that the onboard software would sum neighboring anodes to provide data from 4 anode-pairs (1-2, 3-4, 5-6, 7-8). Note, this does note affect the accumulation time, but does double the area over which the particles are collected, hence the Geometric Factor must be doubled too (see section 8.2 : SNG Geometric Factor values and section 7.3.3 : Anode Summing). Anode summing could only occur with SNG data (all years), or ION data (1993-2003 only, see sections 7.3.3 and 11.4.3.3).

13.5 ANC file

13.5.1 Getting Cassini R, Lat, Local Time: Cassini Ephemeris

13.5.1.1 Radial Distance, Local Time and Latitude from ANC files

While SPICE can be used to produce these values, I like to keep things simpler and use the data we're provided by the CAPS PDS binary files, rather than having to get SPICE working (and keeping all those SPCE files updated, etc.).

All the information you need is provided in the **ANC** files, plus the knowledge of Saturn's pole unit vector in J2000 co-ordinates, provided below.

13.5.1.1.1 Saturn's pole unit vector in J2000

This can be found from SPICE (see chapter 18), however most people have the value hard-wired in their code, yet no one seems to agree exactly on what to use – although all agree to the 4th decimal place. While this value really does change very slowly over time, it's safe to assume it's constant. From SPICE from 1997 through to 2007 inclusive the average is as shown below:

Saturn_Pole_unit(x,y,z) = [0.0854813583, 0.0732355383, 0.9936445508]

The standard deviations are below 0.0000032 for each component.

13.5.1.1.2 Cassini R from Saturn

The **ANC** files contain these 3 variables:
SC_SATURN_POS_X, SC_SATURN_POS_Y, SC_SATURN_POS_Z

The Cassini position vector is then simply:
SC_POS(x,y,z) = (*SC_SATURN_POS_X, SC_SATURN_POS_Y, SC_SATURN_POS_Z*)

Then R is trivially:

R = sqrt(dot(SC_POS , SC_POS))

This is in km, but is trivial to divide by the planet radius to get R_S.

13.5.1.1.3 Cassini's Latitude

The Latitude may be found from the R and Saturn's pole vector (given in sections 4.3.1.1 and 4.3.1.2):

LAT(degs) = 90 - acos(dot(SC_POS , Saturn_Pole_unit) / R) *180/pi

13.5.1.1.4 Cassini's Local Time

This requires the spacecraft position (SC_POS , section 4.3.1.2) and three further variables from the **ANC** files; *SC_SUN_POS_X*, *SC_SUN_POS_Y* and *SC_SUN_POS_Z*.

Calculating Local Time, LT, of Cassini in the Saturn Frame requires the following steps:

SC_SUN_POS(x,y,z) = (*SC_SUN_POS_X, SC_SUN_POS_Y, SC_SUN_POS_Z*)

SUN_POS(x,y,z) = SC_POS(x,y,z) - SC_SUN_POS(x,y,z)
D_S(x,y,z) = cross (Saturn_Pole_unit , SUN_POS)
D_C(x,y,z) = cross (Saturn_Pole_unit , SC_POS)

LT(hours) = (atan2(D_C{y},D_C{x}) - atan2(D_S{y},D_S{x}) + pi)*12/pi

{Note, the {z} components of D_C and D_S are not used.}

Local Time is cyclic, but usually we consider it to lie between 0 and 24 Hours. So best to do a modulus 24 on the end result, or the following two while loops:

```
while ( LT >=  24 )    { LT = LT -  24 }
while ( LT <    0  )    { LT = LT + 24 }
```

13.5.2 How to replicate ANC SC_ORIENT, VEL, POS, SUN POS from SPICE

This is covered in chapter 18: SPICE and generating ANC parameters.

13.6 Co-ordinate transforms

There are several reference frames one could chose to work the CAPS data in, usually one must start at the simplest and do transformation after transformation until you reach the desired co-ordinate system

First there is the CAPS frame, a 2D frame that is simply the anode polar angle and the actuator angle.

The spacecraft frame is 3D (often Cartesian x,y,z), and requires projecting the 2D frame on to the spacecraft.

The spacecraft centered J2000 frame is 3D in inertial space (again Cartesian x,y,z), but has no knowledge of your position related to Saturn.

The Saturn centered spherical RTP frame is often desired.

Or you might want to be in a magnetic field frame, usually spacecraft centered if you wish to forward model anisotropic distributions.

Another general resource of use is the following reference:

Geophysical Coordinate Transformations, C.T. Russell
Originally published in Cosmic Electrodynamics, 2, D. Reidel Publishing Company, Dordrecht-Holland, 184-196, 1971.
Online at: http://www-ssc.igpp.ucla.edu/personnel/russell/papers/gct1.html/

13.6.1 The CAPS Frame

The CAPS frame (for ELS and IMS) is a 2-D frame based on polar angle of the anodes and the actuator angle (in degrees), however this is not the spacecraft frame

Let's call these two values P (the polar angle in degrees) and A (the actuator angle in degrees, where A is the ACT data).

13.6.2 Conversion to the spacecraft frame (RTP)

[RTP = R, theta, phi = Rθϕ]

When A = 0, CAPS is actually pointing along the –y axis of the spacecraft frame. The actuator then rotates in the opposite direction than you might be expecting, such that A=-90 is aligned with the +x axis of the spacecraft frame, and A=+90 the –x axis. See figure 5 in the CAPS instrument paper for more information.

Therefore in the spacecraft frame in spherical co-ordinates, (angles in degrees):

$$\theta = \mathtt{P}$$

$$\phi = \mathtt{270-A}$$

The third dimension is the radius, which is just unity as we want unit matrices.

13.6.3 Conversion to the spacecraft frame (xyz)

Converting the spacecraft frame from RTP to xyz is trivial (R=1 here and is not shown for simplicity), with the 3 components now just:

$$x = \cos(\theta)\cos(\phi) = \cos(P)\cos(270 - A)$$
$$y = \cos(\theta)\sin(\phi) = \cos(P)\sin(270 - A)$$
$$z = \sin(\theta)$$

Putting this into a column vector and simplifying the algebra with some identities:

$$\begin{bmatrix} X_{S/C} \\ Y_{S/C} \\ Z_{S/C} \end{bmatrix}_{Look} = \begin{bmatrix} -\cos(P)\sin(A) \\ -\cos(P)\cos(A) \\ +\sin(P) \end{bmatrix}$$

The unit vector FOV in the spacecraft x,y,z can now be calculated from just P and A. This is the look direction, hence the subscripted to clarify (whereas the plasma flow direction must be the opposite in order to enter the sensor).

13.6.4 Conversion from spacecraft frame (xyz) to J2000 frame (xyz)

Now we have a FOV unit vector in the spacecraft frame we need a 3x3 transformation matrix to convert this to the J2000 frame. Thankfully this matrix is provided in the ANC binaries at A-cycle resolution as 9 separate elements, all beginning SC_ORIENT_ and ending in either XX, XY, XZ, YX, YY, YZ, ZX, ZY, ZZ.

This matrix should not be interpolated as the interpolation changes the unit matrix to just a matrix that needs to be rescaled. If you require sub-32s matrices use SPICE (see SPICE section) to get the appropriate unit matrix for your times.

The ANC file lists the 9 separate elements of the matrix for you to assemble into the 3x3 matrix. The following is the correct matrix to convert a spacecraft frame vector into J2000.

$$\text{SC\_TO\_J2000} = \begin{bmatrix} \text{SC\_ORIENT\_XX} & \text{SC\_ORIENT\_YX} & \text{SC\_ORIENT\_ZX} \\ \text{SC\_ORIENT\_XY} & \text{SC\_ORIENT\_YY} & \text{SC\_ORIENT\_ZY} \\ \text{SC\_ORIENT\_XZ} & \text{SC\_ORIENT\_YZ} & \text{SC\_ORIENT\_ZZ} \end{bmatrix}$$

Note this is the transpose of what you might have assumed based on the letters X-Z, the top line is NOT [SC_ORIENT_XX, SC_ORIENT_XY SC_ORIENT_XZ]!

Now the conversion from xyz co-ordinates in the spacecraft frame to J2000 is the matrix multiplication:

$$\begin{bmatrix} X_{\text{J2000}} \\ Y_{\text{J2000}} \\ Z_{\text{J2000}} \end{bmatrix}_{\text{Look}} = \left[\text{SC\_TO\_J2000}\right] \begin{bmatrix} X_{\text{S/C}} \\ Y_{\text{S/C}} \\ Z_{\text{S/C}} \end{bmatrix}_{\text{Look}}$$

[Remember to renormalize each unit vector as you go to deal with rounding errors introduced on the matrix multiplication computation.]

Should you wish to convert from J2000 co-ordinates to spacecraft you are required to transpose the matrix (note the superscript T):

$$\begin{bmatrix} X_{\text{S/C}} \\ Y_{\text{S/C}} \\ Z_{\text{S/C}} \end{bmatrix}_{\text{Look}} = \left[\text{SC\_TO\_J2000}\right]^{T} \begin{bmatrix} X_{\text{J2000}} \\ Y_{\text{J2000}} \\ Z_{\text{J2000}} \end{bmatrix}_{\text{Look}}$$

13.6.5 Conversion from J2000 frame (xyz) to Saturn centered RTP frame

The data must now be converted from a Cartesian frame to a spherical one, while also moving the origin of the frame to Saturn, which requires knowledge of Saturn and Cassini's position from it.

The ANC binary files gives SC_SATURN_POS_X, SC_SATURN_POS_Y and SC_SATURN_POS_Z, all in J2000 co-ordinates and once per A-cycle.

$$R_{\text{J2000}} = \begin{bmatrix} \text{SC\_SATURN\_POS\_X}_{\text{J2000}} \\ \text{SC\_SATURN\_POS\_Y}_{\text{J2000}} \\ \text{SC\_SATURN\_POS\_Z}_{\text{J2000}} \end{bmatrix}$$

This is then turned into a unit vector:

$$\hat{\mathbf{R}}_{\text{J2000}} = R_{\text{J2000}} / \left| R_{\text{J2000}} \right|$$

Saturn's spin-axis is also required, in particular the declination and right ascension of Saturn's spin pole. While these vary over time very very slowly (use SPICE to get accurate values, chapter 18) they are approximately:
$\text{Saturn\_DEC}_{J2000}$ =83.537544 degrees, Saturn_RA_{J2000} =40.583441 degrees

$$\hat{\Omega}_{J2000} = \begin{bmatrix} \cos(\text{Saturn\_DEC}_{J2000})\cos(\text{Saturn\_RA}_{J2000}) \\ \cos(\text{Saturn\_DEC}_{J2000})\sin(\text{Saturn\_RA}_{J2000}) \\ \sin(\text{Saturn\_DEC}_{J2000}) \end{bmatrix}$$

From here and two other unit (column) vectors can be constructed to make a right-handed set with $\hat{\mathbf{R}}_{J2000}$:

$$\hat{\theta}_{J2000} = \left(\hat{\Omega}_{J2000} \times \hat{\mathbf{R}}_{J2000}\right) \times \hat{\mathbf{R}}_{J2000}$$
$$\hat{\phi}_{J2000} = \hat{\mathbf{R}}_{J2000} \times \hat{\theta}_{J2000}$$

This allows the rotation matrix to be constructed:

$$\text{J2000\_TO\_RTP} = \left[\hat{\mathrm{R}}_{J2000} \quad \hat{\theta}_{J2000} \quad \hat{\phi}_{J2000}\right]^{\mathrm{T}}$$

Note that J2000_TO_RTP is a 3x3 matrix, and the superscripted T means transpose, such that the first row is really the 3 components of $\hat{\mathbf{R}}_{J2000}$.

From here we can now use two rotations to get from spacecraft co-ordinates directly to Saturn centered spherical (KRTP) co-ordinates:

$$\begin{bmatrix} R_{KRTP} \\ \theta_{KRTP} \\ \phi_{KRTP} \end{bmatrix}_{Look} = \left[\text{J2000\_TO\_RTP}\right] \begin{bmatrix} X_{J2000} \\ Y_{J2000} \\ Z_{J2000} \end{bmatrix}_{Look}$$

[Remember to renormalize each unit vector as you go to deal with rounding errors introduced on the matrix multiplication computation.]

Likewise, if you wish to go from Saturn centered RTP co-ordinates to xyz you must transpose the matrix as so:

$$\begin{bmatrix} X_{J2000} \\ Y_{J2000} \\ Z_{J2000} \end{bmatrix}_{Look} = \left[\text{J2000\_TO\_RTP}\right]^{T} \begin{bmatrix} R_{KRTP} \\ \theta_{KRTP} \\ \phi_{KRTP} \end{bmatrix}_{Look}$$

13.6.6 Conversion from spacecraft xyz co-ordinates to Saturn centered RTP

Now you have all the intermediate matrices you can do the conversion in one big matrix multiplication:

$$\begin{bmatrix} R_{\mathrm{KRTP}} \\ \theta_{\mathrm{KRTP}} \\ \phi_{\mathrm{KRTP}} \end{bmatrix}_{\mathrm{Look}} = [\mathrm{J2000\_TO\_RTP}][\mathrm{SC\_TO\_J2000}] \begin{bmatrix} X_{\mathrm{S/C}} \\ Y_{\mathrm{S/C}} \\ Z_{\mathrm{S/C}} \end{bmatrix}_{\mathrm{Look}}$$

13.6.7 Conversion from CAPS frame to Saturn centered RTP

Likewise to go from actuator angle, A, and anode angle, P, can now be carried out in one calculation:

$$\begin{bmatrix} R_{\mathrm{KRTP}} \\ \theta_{\mathrm{KRTP}} \\ \phi_{\mathrm{KRTP}} \end{bmatrix}_{\mathrm{Look}} = [\mathrm{J2000\_TO\_RTP}][\mathrm{SC\_TO\_J2000}] \begin{bmatrix} -\cos(P)\sin(A) \\ -\cos(P)\cos(A) \\ +\sin(P) \end{bmatrix}$$

13.6.8 Conversion from Saturn centered RTP to Magnetic field coordinates

Magnetic field is not measured by CAPS, but rather the MAG instrument on Cassini. However if you wish to use magnetic co-ordinates you must obtain and understand the MAG data set too. Those details are obviously left to the MAG PDS documents, but what follows is how to use them with the CAPS generated transforms above.

13.6.8.1 Obtaining MAG data

Cassini magnetometer data can be obtained from the PDS in KRTP co-ordinates.
http://www.igpp.ucla.edu/cgi-bin/ditdos?volume=COMAG_3XXX&folder=DATA&style=full

A typical description of the data is:

```
Cassini FGM magnetic-field data in the highest time resolution available
for 2006-01-01 (001) in KRTP coordinates. KRTP coordinates are the
standard right-handed spherical triad: R (Saturn to spacecraft), Phi
(parallel to Saturn's equator), and Theta (completes right handed set).
```

However we want a field-aligned co-ordinate system, [$B_{\perp 1}$, $B_{\perp 2}$, $B_{//}$]. The parallel direction is simply that of the unit vector of the magnetic field, **B**.

$$\mathbf{B}_{//} = \begin{bmatrix} B_R \\ B_\theta \\ B_\phi \end{bmatrix}_{\mathrm{KRTP}}$$

$$\hat{\mathbf{B}}_{//} = \frac{\mathbf{B}_{//}}{|\mathbf{B}_{//}|}$$

The first perpendicular direction is found from crossing a purely radial vector (or anything but $B_{//}$) with the parallel direction, and the second perpendicular direction is found from a final cross product.

$$\hat{\mathbf{B}}_{\perp 1} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}_{\mathrm{KRTP}} \times \hat{\mathbf{B}}_{//}$$

$$\hat{\mathbf{B}}_{\perp 2} = \hat{\mathbf{B}}_{//} \times \hat{\mathbf{B}}_{\perp 1}$$

If $B_{//}$ is along B_R ($B_{//} = [1, 0, 0]_{\mathrm{KRTP}}$,) then this won't work, in which case we can replace those equations by crossing it with the unit ϕ vector instead:

$$\hat{\mathbf{B}}_{\perp 1} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}_{\mathrm{KRTP}} \times \hat{\mathbf{B}}_{//}$$

$$\hat{\mathbf{B}}_{\perp 2} = \hat{\mathbf{B}}_{//} \times \hat{\mathbf{B}}_{\perp 1}$$

The transformation matrices then follow (superscripted T is for transpose):

$$\mathrm{RTP\_TO\_MAG} = \left[\hat{\mathrm{B}}_{\perp 1} \quad \hat{\mathrm{B}}_{\perp 2} \quad \hat{\mathrm{B}}_{//}\right]^{\mathrm{T}}$$

RTP_TO_MAG is a 3x3 matrix, and is used as below:

$$\begin{bmatrix} B_{\perp 1} \\ B_{\perp 2} \\ B_{//} \end{bmatrix} = \mathrm{RTP\_TO\_MAG} \begin{bmatrix} B_R \\ B_\theta \\ B_\phi \end{bmatrix}_{KRTP}$$

[Remember to renormalize each unit vector as you go to deal with rounding errors on the computation.]

13.6.8.2 Calculating Magnetic Field Co-ordinates

First we need to convert the RTP magnetic co-ordinate system to our field aligned frame in Cartesian co-ordinates [$\perp_1$, $\perp_2$, //]

$$\begin{bmatrix} \perp_1 \\ \perp_2 \\ // \end{bmatrix}_{\text{Flow}} = \text{RTP\_TO\_MAG} \begin{bmatrix} R_{\text{KRTP}} \\ \theta_{\text{KRTP}} \\ \phi_{\text{KRTP}} \end{bmatrix}_{\text{Flow}}$$

[Note the sub script is Flow, not Look, as when dealing with pitch angles we want to know about the direction plasma is flowing in, not the direction the sensors are looking out at. The conversion between a FOV look vector and a FOV flow vector is simply that one is the negative of the other, purely opposite. $n_{Flow} = -n_{Look}$]

Since the magnetic field is a 3D measurement there are actually 2 pitch angles to calculate. The first, PA, is the angle from the parallel direction and is the most common use of the term pitch angle. However there is a second angle for the perpendicular direction, PerpPA, that is also required if you plan to forward model data in 3D based on a Maxwellian (or other) distribution.
The field-aligned component of a vector can be found from the dot product of the unit vector of $B_{//}$ direction [0,0,1] with the unit vector of the instrument look angle. Since everything is now in our MAG co-ordinates this is simple. When sqrt($\perp_1$*$\perp_1$+$\perp_2$*$\perp_2$+//*//)=1 (i.e. unit vectors, always beware of rounding errors after doing transforms) then the two angles in magnetic co-ordinates are given by:

$\text{PerpPA}_{\text{Rads}} = \text{atan2}(\perp_2, \perp_1)$

and:

$$\cos(\text{PA}_{\text{Rads}}) = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \bullet \begin{bmatrix} \perp_1 \\ \perp_2 \\ // \end{bmatrix}_{\text{Flow}} = 0\perp_1 + 0\perp_2 + 1//_{\text{Flow}} = //_{\text{Flow}}$$

Rearranging this to simply get this "pitch angle":
$\text{PA}_{\text{Rads}} = \cos^{-1}(//_{\text{Flow}})$

This now allows the magnetic field Cartesian co-ordinate system unit vector to be represented as a standard right-handed RTP system:

$$\begin{bmatrix} \perp_1 \\ \perp_2 \\ // \end{bmatrix}_{\text{Cartesian}} \equiv \begin{bmatrix} 1 \\ \text{PA}_{\text{Rads}} \\ \text{PerpPA}_{\text{Rads}} \end{bmatrix}_{\text{RTP}}$$

Therefore to convert RTP to Cartesian components (assuming angles are in Rads):
$\perp_1 = \cos(\pi/2 - PA_{Rads})\cos(PerpPA_{Rads})$
$\perp_2 = \cos(\pi/2 - PA_{Rads})\sin(PerpPA_{Rads})$
$// = \sin(\pi/2 - PA_{Rads})$

These can then simplify to:
$\perp_1 = \sin(PA_{Rads})\cos(PerpPA_{Rads})$
$\perp_2 = \sin(PA_{Rads})\sin(PerpPA_{Rads})$
$// = \cos(PA_{Rads})$

In combination with the previous conversions you can then work back to the spacecraft centered magnetic system.

13.6.9 Converting Velocities from the spacecraft frame to the Saturn frame.

The above transformation matrices are great for positions, however can also be used for velocities. Let's call Cassini's velocity vector *U*, which is provided for you in the ANC files in a J2000 x,y,z coordinate system with objects labeled SC_SATURN_VELOCITY_VX, SC_SATURN_VELOCITY_VY and SC_SATURN_VELOCITY_VZ. For instance, to convert that to Saturn centered RTP is then:

$$\begin{bmatrix} V_R \\ V_\theta \\ V_\varphi \end{bmatrix}_{KRTP\_Cassini} = [J2000\_TO\_RTP] \begin{bmatrix} U_X \\ U_Y \\ U_Z \end{bmatrix}_{J2000}$$

$$= [J2000\_TO\_RTP] \begin{bmatrix} ANC.SC\_SATURN\_VELOCITY\_VX \\ ANC.SC\_SATURN\_VELOCITY\_VY \\ ANC.SC\_SATURN\_VELOCITY\_VZ \end{bmatrix}$$

For analytical purposes, it is often easier to work in a frame centered on the spacecraft (for instance the anodes are at fixed elevations). So from a measured plasma velocity in spacecraft co-ordinates you must first add on the velocity of the spacecraft and then convert to a planetary RTP frame. This is simple vector addition (in any co-ordinate system):

$$V_{Saturn\_Frame} = V_{Cassini} + V_{s/c\_frame}$$

Assuming your calculated plasma velocity is $V_{S/C}$ in spacecraft coordinates, the spacecraft's own velocity is U, then any of the following equations can be used to convert, depending on which reference frame U is in. Here the subscripts KRTP, J2000 and S/C denote RTP, J2000 and spacecraft co-ordinates respectively.

$$\begin{bmatrix} V_R \\ V_\theta \\ V_\phi \end{bmatrix}_{\text{KRTP}} = \left[\text{J2000\_TO\_RTP}\right]\left(\left(\left[\text{SC\_TO\_J2000}\right]\begin{bmatrix} V_X \\ V_Y \\ V_Z \end{bmatrix}_{\text{S/C}}\right) + \begin{bmatrix} U_X \\ U_Y \\ U_Z \end{bmatrix}_{\text{J2000}}\right)$$

$$\begin{bmatrix} V_R \\ V_\theta \\ V_\phi \end{bmatrix}_{\text{KRTP}} = \left[\text{J2000\_TO\_RTP}\right]\left[\text{SC\_TO\_J2000}\right]\left(\begin{bmatrix} V_X \\ V_Y \\ V_Z \end{bmatrix}_{\text{S/C}} + \begin{bmatrix} U_X \\ U_Y \\ U_Z \end{bmatrix}_{\text{S/C}}\right)$$

Alternatively if working in a magnetic field frame these equations may be useful:

$$\begin{bmatrix} V_{\perp 1} \\ V_{\perp 2} \\ V_{//} \end{bmatrix} = \text{RTP\_TO\_MAG}\begin{bmatrix} V_{\text{R}} \\ V_\theta \\ V_\phi \end{bmatrix}_{\text{KRTP}}$$

$$\begin{bmatrix} V_{\perp 1} \\ V_{\perp 2} \\ V_{//} \end{bmatrix} = \text{RTP\_TO\_MAG}\left(\left(\left[\text{J2000\_TO\_RTP}\right]\left[\text{SC\_TO\_J2000}\right]\begin{bmatrix} V_X \\ V_Y \\ V_Z \end{bmatrix}_{\text{S/C}}\right) + \begin{bmatrix} U_R \\ U_\theta \\ U_\phi \end{bmatrix}_{\text{KRTP}}\right)$$

13.6.10 Expected rigid corotation direction:

In Saturn centered KRTP co-ordinates the rigid corotation direction would be purely in the phi direction are rigid corotation speed. However to get the velocity vector as seen from the moving spacecraft frame we much first remove the spacecraft velocity, *U*:

$$\begin{bmatrix} V_R \\ V_\theta \\ V_\varphi \end{bmatrix}_{RTP\_S/C} = \begin{bmatrix} 0 \\ 0 \\ V_{CoRot.} \end{bmatrix}_{KRTP} - \begin{bmatrix} U_R \\ U_\theta \\ U_\varphi \end{bmatrix}_{KRTP}$$

This provides the flow direction of a rigidly co-rotating plasma in the frame of the spacecraft.

13.6.11 Saturn System III or System IV?

We suggest that SPICE or the University of Iowa website is used to find co-ordinate systems that spin with Saturn. The CAPS binary files themselves do not provide the info you require.

The University of Iowa website can be found at:
http://www-pw.physics.uiowa.edu/~jbg/cas.html

13.6.12 Warnings

Each time you do a matrix multiplication of a unit vector it is worth renormalizing the result. This is simply due to computational rounding errors creeping in.

You may wish to do these FOV calculations or co-ordinate transforms on many A-cycles of data, but remember that the actuator is often moving, and that the spacecraft is moving relative to Saturn too. This means that for every data record you require different transformation matrices – remember the actuator can move up to 32 degrees during one A-cycle.

14 Dead Time

14.1 Summary for SNG and ELS Data Analysis

If you use the raw data for higher level processing then:

- ELS does require a dead time correction for high count rates.
  - This probably has already been corrected for in the binary data record, check the ELS.COLLAPSE_FLAG to be sure.
- SNG has a dead time, but is insignificant and can be ignored.

Further details for those interested are in the following sections.

14.2 What is Dead Time?

For a more comprehensive explanation of dead time see section 5.2 of *Paschmann and Daly* (1998).

When an ion enters the sensor it will generate secondary electrons as it passes through the carbon foil, and these electrons get multiplied in the detector to a bigger charge cloud (by the micro-channel plates). That charge cloud is large enough to be measurable and is registered as 1 incoming count on the sensor. But what if a second ion entered the detector just behind the first and their two charge clouds overlapped? Then only 1 incoming count is registered, as it's just one bigger cloud. As such dead time can be thought of as the minimum time after the first ion required so that two separately distinguishable charge clouds can be registered. *Young et al.* [2004] quotes the SNG dead time as 0.2 μs, TOF non-paralyzeable dead time fixed at 2.187 μs, ELS is 1.057 μs And IBS as 0.86 μs.

[ELS value source: Dave Linder, MSSL, their operations team file els_cal001_deadtime.doc, "ELS Calibration Analysis: Deadtime calculations"]

14.3 Settling Time vs. Dead Time

Dead time and settling time are often confused. Settling time is the time the sensor needs to alter the voltage of the ESA from one value to the next and arrive to a new steady value. While the sensor is 'dead' during settling time in that it is not recording any measurements, it's only not taking measurements because it's getting set up at the correct voltage for the next measurement interval rather than it can't.

See section 7.9 for more information on Settling time and values for SNG and ELS.

14.4 SNG Dead Time

No dead time correction has been applied to the raw SNG data in the binary files. Assume there is no dead time in your SNG data analysis (or set it to zero). Only read more of this section if you want more details.

Further comments on Dead Time for SNG data are found from the CAPS team meeting #41, 'Reaching Consensus on IMS Calibration' by Crary. This quotes:

Dead-time correction: Don't
– Fast amplifiers should be good to ~2 Mcps
– Constant fraction discriminators are not
– IMS set to "require coincidence"
– Effects on dead time are not clear
– Check to see if that might be a problem

If you really want to include its effect for SNG data then Section 6.5.2 of *Young et al.* [2004] quotes the SNG dead time as 0.2 μs.

However dead time does give us an upper limit of a maximum counts per $dt_{1\text{-}step}$ per anode. For SNG the upper limit will be the 1-step accumulation time divided by the dead time, 54.6875ms / 0.2 μs = 273437.5 counts.
This is an order of magnitude greater than the maximum quantized count for SNG of 27499, as such dead time is not an issue and ignoring it is a reasonable approximation.

14.5 ELS Dead Time

ELS data analysis does require corrections for dead time. In fact the CAPS data are corrected for this onboard Cassini in most situations – but not all the time therefore you must check the data, then apply the dead time correction if necessary.

The important flag to check is the ELS.COLLAPSE_FLAG, which is given for each and every ELS.DATA record (and will be the same for all records within the same A-cycle).

The ELS.COLLAPSE_FLAG description (From the ELS_U3.FMT files) states:

```
DESCRIPTION    = "Flag indicating how data is collapsed:
                  0: average
                  1: sum
                  2: average with in-flight deadtime correction
                  3: sum with in-flight deadtime correction
                  4: snapshot portion
                  NOTE:  For snapshot, full collapse information is
                         gained by adding 4 (so snapshot portion can be
                         4, 5, 6, or 7 depending upon the collapse.
                         The upper bit will be set to 1 when
                         housekeeping is missing."
```

As such, if ELS.COLLAPSE_FLAG equals 0 or 128 then this is 'average' collapse and a dead time correction must be applied.

The non-paralyzable correction for dead time, t_{dead}, over an accumulation time of t_{acc} is:

$$C_{Cor} = \frac{C_{Obs}}{1 - C_{Obs}\frac{t_{dead}}{t_{acc}}}$$

Where C_{Obs} is the observed counts (from ELS.DATA) and C_{Cor} is the corrected counts. The ELS moments are carried out with t_{dead}/t_{acc} = 44.8e-6

If ELS.COLLAPSE_FLAG equals 1 or 129, then the counts were summed before any dead time correction. As such the correction must be applied, however this is less accurate as we would ideally apply the cross talk correction to the sections prior to summing.

15 Ways to work out the background of Plasma Data

There are several ways to remove a background from plasma data, and it's your choice depending on the environment. After you remove a background from your count data you should check for any negative values, as clearly there cannot be a negative count rate. This is not uncommon though, and a common technique is to change any negative values to zero.

Remember that the background value is likely different for each anode, and each azimuth or azimuth-sum. Unless you are working with a singular 1D slice of data you will have several background values to calculate.

Also remember to check for fill values (MISSING_CONSTANTS) before blindly removing a background – as those should remain fill values and not be touched.

15.1 Zero eV Step

If the sweep table you are using has the lowest step at 0 eV, then this must be a measure of the background.

15.2 Use an energy step outside the main distribution of counts

If the plasma distribution is between energy steps 30 and 60 then energy step 10 could be used as a good proxy for the background. Likewise if the plasma distribution is higher and between energy steps 5 and 40, then an energy step nearly 50 or 60 would be suitable for a background.

Always plot spectrograms of the data to be sure that your chosen 'background step' is indeed at background. The following figure (Figure 15.1) is taken from *Hill et al.* (2012), showing that it is possible to have interesting features at both high and low energies. In this case, an energy step around 300 eV would have made a suitable background value. Had you blindly taken the highest energy step to use as background you would have confused the grain distribution with the background.

Just taking one energy step value may be noisy though, so another common technique is to use a range of energies and take a mean. For instance for a given anode and azimuth sum (assuming the plasma distribution is in energy steps 30 to 60) then a mean of energy steps 5 to 10 will give a more robust background value.

A warning for SNG/ION data though, never use energy steps 1 to 3 for science use, nor for calculating backgrounds (see section 8.1.1).

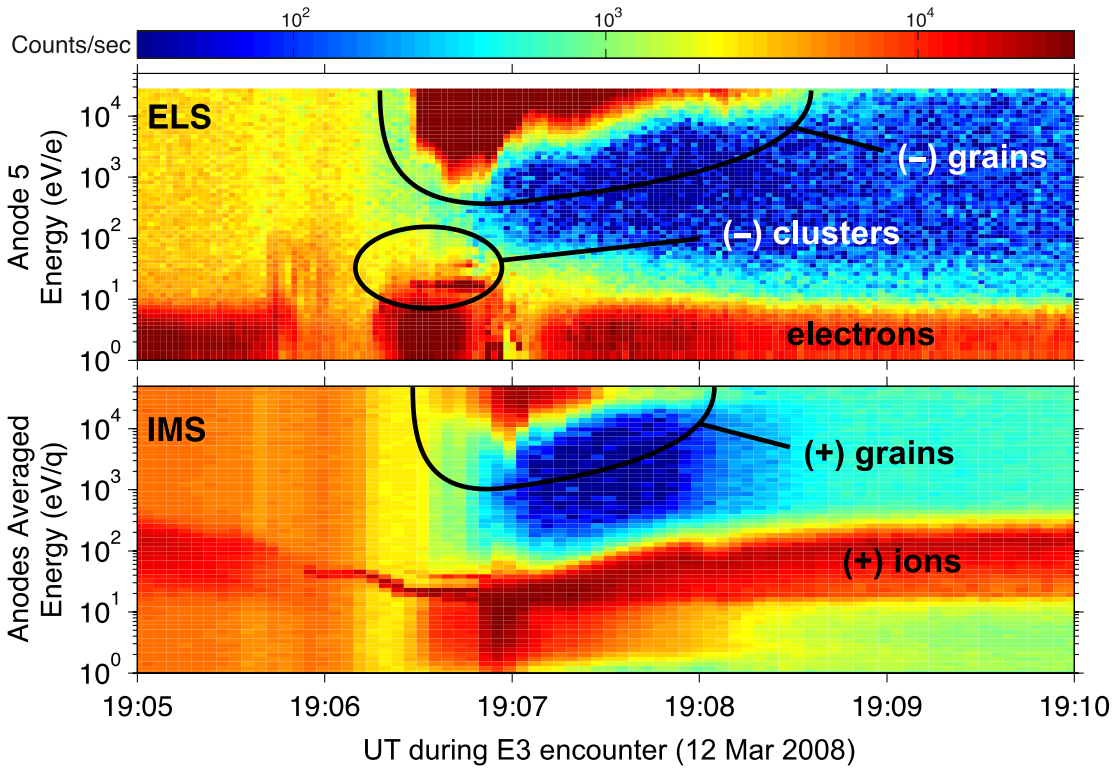

Figure 15.1: Example of a difficult interval to locate a background. This highlights the importance of always plotting the data before picking an energy step to use as a background value (the usual "safe" choice of the highest eV/q energy step would fail here). This is figure 2 from *Hill et al.* (2012), showing ELS (top) and IMS-SNG (bottom) for the third Enceladus encounter.

15.3 The "95%" method

This idea is based on the idea that 95% of the measured intervals (i.e. if 1 azimuth there are 63 intervals of ELS data) must be above the background level. Note this is not 95% of total counts measured, but intervals.
If one sweep for one anode has 63 usable data values then 0.95*63 = 59.85, or 60 to the nearest whole number. Therefore 3 of the 63 energy steps must be at background level or below. Now all that is left is to find the 3rd smallest value of the 63, and call the counts in that energy step the background.

Of course you don't have to use 95% here, you could use 99% or any other value you see fit.

The Poisson distributed background method

For each energy sweep per anode find the lowest value, which we'll call *B*.
Alternatively you may wish to take an average of every 3 energy bins and find the lowest average and use that for *B*, in order to remove noise effect.
If the background is assumed to be Poisson distributed, then the real background is likely 1 standard deviation from this value, then the standard deviation is simply sqrt(*B*). As such the Poisson distributed background method would say the background for that sweep/anode is:

$$\text{Background} = B + \text{sqrt}(B)$$

15.4 IBS only – trust the spacecraft

For IBS data a background value is calculated for you already, once per A-cycle for each of the 3 anodes. These values are not found in the IBS binary files, but the ANC file, and the ANC object DATA_IBS_BKGD. This may not be high enough resolution for your work, in which case use one of the other methods above.

However ANC.DATA_IBS_BKGD may be fill values (65535) or 0, and if all three anodes are the zeros you should calculate your own background value in preference based on the rest of the IBS data. However the run-length encoding of IBS data may make this tricky.

16 How to correct for Spacecraft Potential

Spacecraft potential can be important when converting data into scientific units as it effectively adjusts the energy scale. However there is no instrument on Cassini directly measuring the spacecraft potential. For heavier ion species (i.e. O^+) spacecraft potential has little effect (and is often ignored completely), but for light species (i.e. e^-) it can have a significant effect.

For instance, a positive spacecraft potential will attract electrons to it, accelerating them towards the sensors and hence registering counts at higher energies.

Correctly accounting for spacecraft potential is a higher-order process and as such it is suggested that the user read up on Liouville's theorem before attempting to incorporate it. Below are some very basic comments only.

According to Louisville's theorem, the data in units of PSD can be shifted linearly along the energy values by the magnitude of the potential.

The spacecraft potential corrected energy, E', is given by:

$$E' = E + q*V$$

Where E is the measured E, q is the charge of the ion (-1 for electrons, +1 for singularly charged ions, +2 for doubly charged ions) and V is the spacecraft potential.

An estimate of spacecraft potential can be made from the ELS spectrograms if the spacecraft potential is positive, by reading off the eV level of the photoelectron PSD distribution (*Lewis et al.*, 2008, see section 20.1.2). However if the spacecraft potential is negative then parts of the electron distribution may be below the energy range measured by ELS, in this case you cannot infer the spacecraft potential from electron data. If there is a sufficient flux of ions at low energies, the ion data may be used to determine the spacecraft potential. All observed ions will have an energy greater than E-qV and only noise will be seen at lower energies. A sharp increase in ion counts at a threshold energy can, therefore, indicate the negative spacecraft potential.

Figure 16.1 from the MSSL operations team provides an example of ELS data with the black line at a few eV showing the spacecraft potential they had worked out (scale on the right vertical axis) inferred from the same data.

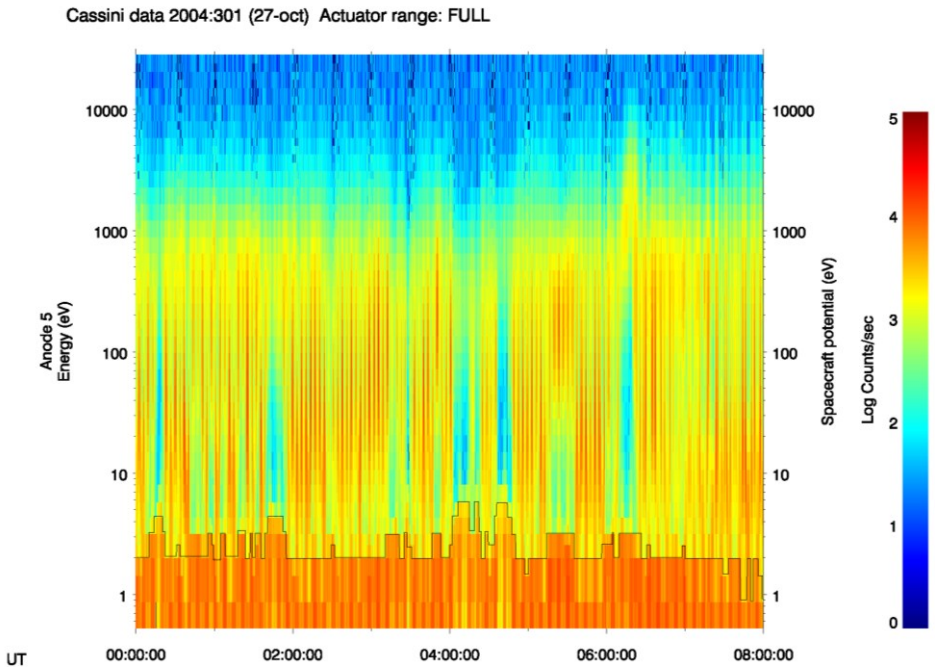

Figure 16.1: Example ELS plot showing photoelectrons.

17 How to Identify MCP Gain Tests

MCP gain tests are carried out regularly on CAPS to monitor the calibration values. Essentially the micro channel plate (mcp) voltages are dropped down, then slowly stepped up again measuring counts. Few counts are registered when the voltages are dropped, and the numbers increase as the level is stepped up again. This feature makes it easy to identify in the data, shown in Figure 17.1.

IMS gain tests are very easy to identify from visually inspecting the data or by filtering the ANC data. For IMS mcp tests a special calibration table is used, table number 16, so a search of the ANC.IMS_SWEEP_TABLE_NUMBER for values equal to 16 immediately identifies it. Likewise, if carrying out scientific analysis you should check the table is not 16 before using it. This special table however does two sweeps through the voltages in the 63 bins, which makes it very easy to spot in energy step vs. time spectra, as shown in the top panel of the following figure.

For IMS singles there is also a voltage monitor that may be checked, which is located in the ANC files. The HVU2_ST_DAC object in the ANC files gives the mcp voltage (side note, this is the same value as the HSK files HVU2_ST_MN object – not that HSK files are included in the PDS). A plot of this variable against time easily shows the steps as the negative values decrease (closer to zero) then step up in magnitude again, as shown in the second panel of Figure 17.1. It looks like this feature only covers the second half of the calibration area – this is correct; IMS has both start and stop signals, therefore two sets of mcps to test, and this is just the start one.

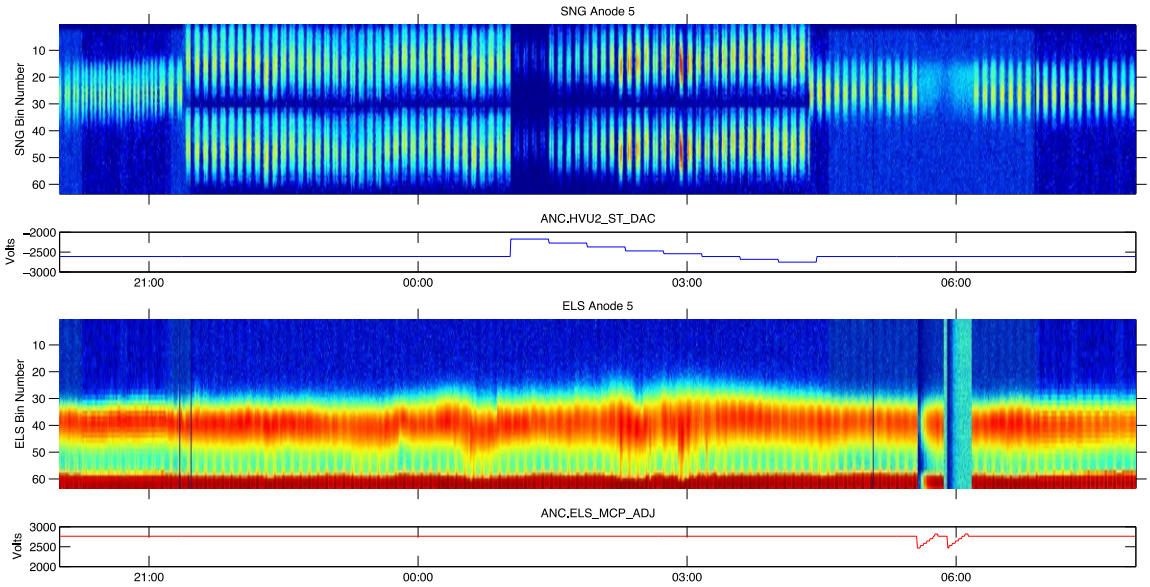

Figure 17.1: Examples of MCP tests with CAPS data.
Time range is 2005-10-19T20:00:00 to 2005-10-20T08:00:00.

Generally IMS and ELS gain tests are done back to back, as such panel 3 of Figure 17.1 shows an ELS energy step vs. time spectrogram for the same interval. The ELS mcp test takes much less time than the equivalent IMS one and as such is easy to miss. And unlike IMS there is no species sweep table for calibration that can be searched for and flagged. The mcp voltage monitor for ELS can be found in the ANC files under object ELS_MCP_ADJ, which is plotted in the final panel of Figure 17.1. The drops down and steps back up are obvious. (The MCP voltage measurement initially comes from the housekeeping data, which only returns samples every second A-cycle. The voltage can change much faster than that, especially during automatic safing event and gain tests, so the ELS_MCP_ADJ value can only be used as a guideline in those circumstances.)

To convert the ELS_MCP_ADJ value to an ELS "mcp level" simply divide it by 58.73 V, see section 9.4.

Note how the IMS voltages (panel 2) were all negative and the ELS voltages (panel 4) are all positive.

18 SPICE and generating ANC parameters

The ANC files contain position and orientation information that is derived from SPICE, with the SPICE kernel files available at the time of ANC file production. The following commands allow you to replicate what is given in the ANC files.

For each day directory of PDS CAPS files there is a file called KERNELS_USED.DAT. This is a text file that lists the actual SPICE kernels that were used for the ANC*.DAT files and looks something like this:

```
SPK: 050105RB_SCPSE_04247_04336.bsp
00: 04293_04298ra.bc
06: 04293_04298ra.bc
12: 04293_04298ra.bc
18: 04293_04298ra.bc
```

If the line begins with 00:, 06:, 12: or 18: (hours) then that SPICE kernel is uniquely for that day-quarter file.

Note 1: it does not list the SPICE kernel files for leap seconds nor planetary constants, just the ones relevant to Cassini's position and orientation.

Note 2: SPICE is completely unaware of what the actuator is doing.
You can use SPICE to tell you where Cassini was and how it was orientated, but not what the actuator angle is. So if you require the CAPS field of view you must get the orientation then manually account for the actuator angle given in the ACT files. (See chapter 13.)

Each ANC's LBL file also lists the kernel files used (the SPICE_FILE_NAME object).

If you use those SPICE kernels listed you should be able to exactly recreate the key position/orientation/velocity values given in the CAPS PDS ANC files.

Naturally over time the SPICE kernels may be refined (i.e. for moon encounters where position to the nearest 1 R_S is too broad). The following will allow you to regenerate the key ANC values should new SPICE position/attitude files be released in the future. Alternatively if you wish to get ANC information at a time resolution of your choosing (rather than A-cycle resolution) then the following codes are useful

All examples are given using 'mice', the SPICE for Matlab package, however similar commands are available for IDL/c/etc. SPICE routines and the function calls are the same. A final example of all is given at the end. These are not necessarily the most efficient or best way of doing it (SPICE is a whole language in itself) – but they seem to give matching values to the ANC files for the same SPICE kernels.

18.1 Time, et

All SPICE commands require the time to be in ephemeris times (*et*) format.
As it happens *et* = Barycentric dynamic time (seconds), so the CAPS TIME variable is already in the correct format/units.

18.1.1 Human Readable Time to *et*

If you want to convert a human readable date to *et* (Barycentric dynamic time) then the command is:

```
et = cspice_str2et( t_cell );
```

where

```
t_cell = …
  {'2005-11-26T00:00:00';'2005-11-26T00:01:00';'2005-11-26T00:02:00'};
```

or to do an array from 2005-Nov-26 22:00:25 to 23:00:41 at 60 second intervals, you need two lines:

```
t=datenum(2005,11,26,22,00,25):(60/86400):datenum(2005,11,26,23,00,41);
t_cell = cellstr(datestr(t,'yyyy-mm-ddTHH:MM:SS'));
```

The first line sets up the time array in Matlab serial time, the second line puts these times into the human readable format the *str2et* will accept, then convert to *et*.

18.1.2 *et* to Human Readable Time

To get a string of characters in human readable date:

```
utcstr = cspice_et2utc( et, 'C', 3 )
```

The 3 is the decimal place of precision for the seconds, i.e. 3 means give to millisecond accuracy.
The 'C' is for 'Calendar format, UTC', which is yyyy MMM dd HH:MM:SS.sss
Therefore the output string will be 24 characters long, and will resemble a time such as 2004 JAN 31 08:06:15.572

To convert that time strings into Matlab serial time (and keep milliseconds):

```
t = datenum(utcstr(:,1:24),'yyyy mmm dd HH:MM:SS.FFF')
```

18.2 Position and Velocity of Cassini

The same SPICE command is used to work out Position and Velocity simultaneously.

18.2.1 General SPICE Command

Assuming *et* is a 1-D array of size n then the position and velocity of Cassini with respect to Saturn in J2000 coordinates is:

```
[state,ltime]= cspice_spkezr('Cassini',et,'J2000','NONE','SATURN');
```

where:

position = state(1:3,1:n);
velocity = state(4:6,1:n);

Note 1: `ltime` is the one-way light time between Cassini and Saturn, and is not required for the calculation or position of velocity at all. It is only included here for completeness of the spkezr command.

Note 2: Replacing 'SATURN' with 'SATURN BARYCENTER' gives different numbers, as they are different co-ordinate systems.

Note 3: replace *spkezr* with *spkpos* returns just position,
i.e. *state* would only be size 3xn

18.2.2 Comparison with ANC variables:

After these commands to get *state1* and *state2*, assuming *et* is again a 1-D array of times of size *n*:

```
[state1,ltime1]= cspice_spkezr('Cassini',et,'J2000','NONE','SATURN');
[state2,ltime2]= cspice_spkezr('Cassini',et,'J2000','NONE','SUN');
```

(`ltime1` and `ltime2` above are for completeness of the command and not required for the following calculations.)

Then the ANC variables listed on the left hand side are given by:

SC_SATURN_POS_X = state1(1,1:n)
SC_SATURN_POS_Y = state1(2,1:n)
SC_SATURN_POS_Z = state1(3,1:n)
SC_SATURN_VELOCITY_VX = state1(4,1:n)
SC_SATURN_VELOCITY_VY = state1(5,1:n)
SC_SATURN_VELOCITY_VZ = state1(6,1:n)
SC_SUN_POS_X = state2(1,1:n)

SC_SUN_POS_Y = state2(2,1:n)
SC_SUN_POS_Z = state2(3,1:n)
SC_SUN_VELOCITY_VX = state2(4,1:n)
SC_SUN_VELOCITY_VY = state2(5,1:n)
SC_SUN_VELOCITY_VZ = state2(6,1:n)

Units: All positions are in km, velocities in km/s.

The specific kernels used for each 6 hour ANC file can be found listed in the ANC's LBL file under the SPICE_FILE_NAME object, or in the

18.3 Spacecraft Orientation

Knowing where the spacecraft is located is only half the battle, you also need to know its orientation to put the instrument field of view into context. ANC files give the rotation matrix from spacecraft co-ordinates into J2000 co-ordinates. As usual it's a one line SPICE command – however we need to set up a few variables first. You also need a few more SPICE kernels loaded up for it to work, including the leap seconds kernel and the sclk kernel (spacecraft clock).

SC_ID = the scalar integer NAIF ID of the spacecraft. Cassini is -82.
SC_Bus = the scalar interger NAIF ID of the spacecraft bus. Cassini is -82000.

Note the minus sign on both, they must be negative.

Toltik = tolerance for pointing data in spacecraft clock ticks. So if you want to within 5 seconds, then you must convert 5 seconds into the number of clock ticks for the spacecraft in question.
Cassini's click has 256 per seconds, but easier to let the code work this out

```
SC_ID  = -82; % Cassini
SC_Bus = -82000; % Cassini

tol_sec = 5;  % Duration in seconds
% so number of ticks in that time is difference between time tol_sec
% and time zero
toltik = cspice_scencd(SC_ID, cspice_sce2s(SC_ID, tol_sec)) - …
         cspice_scencd(SC_ID, cspice_sce2s(SC_ID, 0));
```

The clock time has to be encoded too:

```
% cspice_ckgpav requires encoded spacecraft clock time.
% sclkdp = cspice_scencd(SC_ID, cspice_sce2s( SC_ID, et ));
sclkdp = cspice_sce2c( SC_ID, et );
```

And finally the command can be run:

```
[cmat,av,clkout,found] = cspice_ckgpav(SC_Bus,sclkdp,toltik,'J2000');
```

cmat is a 3x3xn array and the one of use.
You should check the *found* flag to check it found a value within the tolerance you gave.

18.3.1 Comparison with ANC variables:

SC_ORIENT_XX = cmat(1,1,1:n)
SC_ORIENT_XY = cmat(1,2,1:n)
SC_ORIENT_XZ = cmat(1,3,1:n)
SC_ORIENT_YX = cmat(2,1,1:n)
SC_ORIENT_YY = cmat(2,2,1:n)
SC_ORIENT_YZ = cmat(2,3,1:n)
SC_ORIENT_ZX = cmat(3,1,1:n)
SC_ORIENT_ZY = cmat(3,2,1:n)
SC_ORIENT_ZZ = cmat(3,3,1:n)

Note: If doing this in Matlab then to shrink 3D to 1D arrays use one of the following methods:
SC_ORIENT_XX(1:n,1) = cmat(1,1,1:n);
or
SC_ORIENT_XX = squeeze(cmat(1,1,1:n));

18.4 Recreating ANC variable data

The following is Matlab code to recreate the ANC values, which was tested again the ANC file ANC_200533018_U2.DAT. The ANC_200533018_U2.LBL file lists:

```
SPICE_FILE_NAME = {"SPK: 060111R_SCPSE_05320_05348.bsp",
                   "00: 05327_05332ra.bc",
                   "06: 05327_05332ra.bc",
                   "12: 05327_05332ra.bc",
                   "18: 05327_05332ra.bc"}
```

That tells us which spk and ck kernels to use below.
Also required are the lsk kernel for time conversions (leap seconds) and sclk for the spacecraft clock times needed for orientation. For both lsk and sclk use the latest file you can get, they will cover all previous times.

```
%%
% Add SPICE 'mice' to Matlab path, assuming root dir is /Users/wilsonr
addpath('/Users/wilsonr/Documents/Matlab/mice/lib')
addpath('/Users/wilsonr/Documents/Matlab/mice/src/mice')
% Set root path to where you keep your kernel files:
root_SPICE = '/Users/wilsonr/Data/Cassini/SPICE/cosp_1000/data/';

% Clear previous Kernels to avoid mixing.
cspice_kclear

% Load in Kernels - remember to pick (and update) these appropriately.
% The ... just means continue this command on the following line
cspice_furnsh( { fullfile(root_SPICE,'ck/05327_05332ra.bc'      ), ...
                 fullfile(root_SPICE,'lsk/naif0010.tls'         ), ...
                 fullfile(root_SPICE,'sclk/cas00131.tsc'        ), ...
         fullfile(root_SPICE,'spk/060111R_SCPSE_05320_05348.bsp') } );

% Set up Spacecraft ID details
SC_name = 'Cassini';
SC_ID   = -82      ; % Cassini
SC_Bus  = -82000   ; % Cassini

% Make a time array
% Going from 2005-Nov-26 22:00:25 to 23:00:41 in steps of 32 seconds:
% t is time in Matlab serial time
t = datenum(2005,11,26,22,0,25):(32/86400):datenum(2005,11,26,23,0,41);

% convert to a cell array of strings of these times
t_cell = cellstr(datestr(t,'yyyy-mm-ddTHH:MM:SS'));  % must have the T

% Calulate Barycentric Dynamic Time
et = cspice_str2et( t_cell );

n = numel(t); % size of t array
```

```
% Get Position and Velocity details
[state1,ltime1]=cspice_spkezr(SC_name,et,'J2000','NONE','SATURN');
[state2,ltime2]=cspice_spkezr(SC_name,et,'J2000','NONE','SUN'   );


% Calculate a tolerance for pointing data. Use 5 seconds,
% convert to ticks by multiplying by the number of ticks per second.
tol_sec = 5;  % Duration in seconds

% so number of ticks in that time is difference between time tol_sec
% and time zero
toltik = cspice_scencd(SC_ID, cspice_sce2s(SC_ID, tol_sec)) - ...
         cspice_scencd(SC_ID, cspice_sce2s(SC_ID, 0));

% cspice_ckgpav requires encoded spacecraft clock time.
% sclkdp = cspice_scencd(SC_ID, cspice_sce2s(SC_ID, et ) );
sclkdp = cspice_sce2c( SC_ID, et );


% now call SPICE to get orientation info
 [cmat,av,clkout,found]=cspice_ckgpav(SC_Bus,sclkdp,toltik,'J2000');


% Reassign variable names to match those in ANC files:
SC_SATURN_POS_X(1:n)       = state1(1,1:n);
SC_SATURN_POS_Y(1:n)       = state1(2,1:n);
SC_SATURN_POS_Z(1:n)       = state1(3,1:n);
SC_SATURN_VELOCITY_VX(1:n) = state1(4,1:n);
SC_SATURN_VELOCITY_VY(1:n) = state1(5,1:n);
SC_SATURN_VELOCITY_VZ(1:n) = state1(6,1:n);
SC_SUN_POS_X(1:n)          = state2(1,1:n);
SC_SUN_POS_Y(1:n)          = state2(2,1:n);
SC_SUN_POS_Z(1:n)          = state2(3,1:n);
SC_SUN_VELOCITY_VX(1:n)    = state2(4,1:n);
SC_SUN_VELOCITY_VY(1:n)    = state2(5,1:n);
SC_SUN_VELOCITY_VZ(1:n)    = state2(6,1:n);

SC_ORIENT_XX(1:n)          = cmat(1,1,1:n);
SC_ORIENT_XY(1:n)          = cmat(1,2,1:n);
SC_ORIENT_XZ(1:n)          = cmat(1,3,1:n);
SC_ORIENT_YX(1:n)          = cmat(2,1,1:n);
SC_ORIENT_YY(1:n)          = cmat(2,2,1:n);
SC_ORIENT_YZ(1:n)          = cmat(2,3,1:n);
SC_ORIENT_ZX(1:n)          = cmat(3,1,1:n);
SC_ORIENT_ZY(1:n)          = cmat(3,2,1:n);
SC_ORIENT_ZZ(1:n)          = cmat(3,3,1:n);

% Clear Kernels to avoid accidentally using the wrong ones later
cspice_kclear

% All Finished.
disp('J2000')
```

18.5 Finding Radius of Saturn (or Moons)

This requires the use of the pck kernels.

The command is:

```
R = cspice_bodvrd('Saturn', 'RADII', 3);
```

The output is a 3 element array:
R(1) = The first number is the largest equatorial radius (the length of the semi-axis containing the prime meridian)
R(2) = the second number is the smaller equatorial radius
R(3) = the polar radius
[Reference: see the 'Shape models' section of the P_constants (PcK) SPICE kernel file, the above is from pck00010.tpc]

The above command returns the following three numbers:
60268
60268
54364

where units are in km. For large round things like Saturn you can expect R(1) and R(2) to be the same.

Replace 'Saturn' for other moons/planets as required.

i.e. Matlab code would be

```
cspice_kclear  % clear out loaded kernals, for safety
cspice_furnsh( fullfile(root_SPICE,'pck/cpck15Nov2005.tpc') )

% Find radius
R = cspice_bodvrd('Saturn', 'RADII', 3 );
R_Saturn = R(1);

cspice_kclear  % clear out loaded kernals, for safetly
```

18.6 Finding Saturn's Pole Unit Vector in J2000 (or Moons)

The values are essentially constant (see section 13.5.1.1.1)
Saturn_Pole_unit(x,y,z) = [0.0854813583, 0.0732355383, 0.9936445508]
But can be worked out via SPICE, and requires the use of the pck kernels to be used.

The command is:

```
inert2p_mat = cspice_pxform('IAU_SATURN','J2000', et);
```

inert2p_mat is a 3x3xn (where n is the size of et) and as the pole vector is effectively z = (0,0,1) the pole vector is given by the last column of the 3x3 matrix for each time.

Replace 'IAU_SATURN' for other moons/planets as required.
Note, 'SATURN' does not work, must be IAU_SATURN.

i.e. Matlab code would be (with the latest SPICE kernel you can find):

```
cspice_kclear  % clear out loaded kernals, for safetly
cspice_furnsh( fullfile(root_SPICE,'pck/cpck15Nov2005.tpc') )

% Find Pole Vector
inert2p_mat = cspice_pxform('IAU_SATURN','J2000', et);

clear pole % erase arrays first - for safety
pole(1:3,1:n) = inert2p_mat(1:3,3,1:n); % convert the 3D matrix to 2D

cspice_kclear  % clear out loaded kernals, for safetly
```

18.7 Planet Spin Period

This requires the use of the pck kernels.
Since the exact value of Saturn's spin period is still being discussed you may wish to use your own value rather than that fixed in SPICE.

```
PM = cspice_bodvrd('Saturn', 'PM', 3 );
```

The second element of PM is then the number of degrees the planet rotates in 86400 seconds. Therefore planetary period = 86400*360/PM(2) = 10h39m22s or so.

18.8 Relating this to University of Iowa's web ephemeris tool

Iowa's website is very useful and will quickly provide you with Cassini's position in a variety of co-ordinate system. However it does not give co-ordinates in the J2000 system, so is not a 1:1 match to ANC data. The University of Iowa's web ephemeris tool can be found at: http://www-pw.physics.uiowa.edu/~jbg/cas.html

19 Relating Instrument Count Rate to Phase Space Distribution

The formulae for converting from count rates to phase space distributions differ for ELS and SNG by a factor of 2, as show in sections .

The origin of this factor of 2 difference is often questioned. Roberto Livi (SwRI) has documented the explanation, and the following two pages are a direct copy of his document. References given are full listed in chapter 21.

Baumjohann and Treumann (1996) derive a relationship between an instrument's count rate and phase space density (PSD)

$$J(E,\alpha,x) = \frac{2E}{m^2} f_s \tag{19.1}$$

where J is the differential particle flux per unit area at given energy, pitch angle and position. The Cassini Plasma Spectrometer (CAPS) does not collect particles over single energies, as shown above, but over an energy range $\Delta E/E$. Therefore,

$$J(\frac{\Delta E}{E},\alpha,x) = \frac{2E}{m^2} f_s . \tag{19.2}$$

Multiplying both sides by E,

$$J(\Delta E,\alpha,x) = \frac{2E^2}{m^2} f_s . \tag{19.3}$$

A discrepancy exists between this result and that of *Young et al.* (2004), which states that

$$\frac{C}{G\Delta t} = J(\frac{\Delta E}{E},\alpha,x) = \frac{4E^2}{m^2} f_s \tag{19.4}$$

where G is defined as $G = A\epsilon\tau\Delta\Omega\Delta E/E$. The derivation was taken from *Johnstone et al.* (1987), who expressed the following result

$$\frac{C}{G\Delta t} = v_0^4 f_s = \frac{4E^2}{m^2} f_s \tag{19.5}$$

with $G = A\epsilon\tau\Delta\Omega\Delta v/v$.

This conversion is only acceptable if we treat the geometric factor in units of $[cm^2\ sr\ \frac{m}{s}/\frac{m}{s}]$ as was done by *Johnstone et al.* (1987). When using units of $[cm^2\ sr\ eV/eV]$, the conversion from velocity intervals $\Delta v/v$ to energy intervals $\Delta E/E$ is taken into account in the following manner:

$$E = \frac{1}{2}mv^2$$

$$\frac{\Delta E}{E} = \frac{mv\Delta v}{\frac{1}{2}mv^2}$$

$$\frac{\Delta E}{E} = 2\frac{\Delta v}{v}. \qquad (19.7)$$

Therefore, Equation 19.4 needs an extra factor of 2, resulting in

$$\frac{C}{G\Delta t} = \frac{4E^2}{2m^2}f_s = \frac{2E^2}{m^2}f_s \qquad (19.8)$$

as in Equation 19.3. Note: the geometric factor for the Ion Mass Spectrometer (IMS) already includes the extra factor of 2. Thus, when converting ion counts to PSD, equation 19.4 is used.

20 Calculating Plasma Parameters/Moments

20.1 Methods and Useful References

Higher level analysis techniques such as generating moments are far beyond the scope of this document – however you should now have all the tools you need in order to do so (i.e. be able to work out pointing info and convert the raw counts data into scientific units). Several methods for generating plasma densities, temperatures and bulk velocities exist (note you may have to use Cassini Magnetometer data to correctly work out anisotropic temperatures), so here is a selection of references (ordered by date) that may help in replicate various techniques.

20.1.1 CAPS instrument paper

Young, D.T. *et al.* (2004), Cassini Plasma Spectrometer Investigation, *Space Science Reviews*, 114, 1-112.
doi: 10.1007/s11214-004-1406-4

20.1.2 ELS

Lewis, G.R. *et al.* (2008), Derivation of density and temperature from the Cassini Huygens CAPS electron spectrometer, *Planetary and Space Science*, **56**, 901-912.
doi: 10.1016/j.pss.2007.12.017

Arridge, C.S. *et al.* (2009), The effect of spacecraft radiation sources on electron moments from the Cassini CAPS electron spectrometer, *Planetary and Space Science*, **57**, 854-869.
doi: 10.1016/j.pss.2009.02.011

Lewis, G.R. *et al.* (2010), The calibration of the Cassini-Huygens CAPS Electron Spectrometer, *Planetary and Space Science*, **58**, 427-436.
doi: 10.1016/j.pss.2009.11.008

For details about a cold-beam ram-pointing algorithm see:
Coates, A.J. *et al.* (2009), Negative ions in the Enceladus plume, *Icarus*,
doi: 10.1016/j.icarus.2009.07.013
Hill, T. W., *et al.* (2012), Charged nanograins in the Enceladus plume, *J. Geophys. Res.*, 117, A05209,
doi: 10.1029/2011JA017218.

20.1.3 SNG

Wilson, R.J. *et al.* (2008), Cassini plasma spectrometer thermal ion measurements in Saturn's inner magnetosphere, *Journal of Geophysical Research (Space Physics)*, **113**, A12, 12218-+.
doi: 10.1029/2008JA013486

Wilson, R.J. *et al.* (2009), Thermal ion flow in Saturn's inner magnetosphere measured by the Cassini plasma spectrometer: A signature of the Enceladus torus?, *Geophys. Res. Lett.*, **36**, 23104-+.
doi: 10.1029/2009GL040225

Thomsen, M.F. *et al.* (2010), Survey of ion plasma parameters in Saturn's magnetosphere, *Journal of Geophysical Research (Space Physics)*, **115**, A14, A10220.
doi: 10.1029/2010JA015267

Thomsen, M.F. and Delapp, D.M (2005), Numerical Moments Computation for CAPS/IMS, LA-UR-05-1542 (Los Alamos Unlimited Release)
http://nis-www.lanl.gov/nis-projects/caps/Moments_Computation.pdf

20.1.4 TOF/ION

Sittler, E.C. *et al.* (2006), Cassini observations of Saturn's inner plasmasphere: Saturn orbit insertion results, *Planetary and Space Science*, **54**, 1197-1210.
doi: 10.1016/j.pss.2006.05.038

Smith, H.T. *et al.* (2007), Enceladus: The likely dominant nitrogen source in Saturn's magnetosphere, *Icarus*, **188**, 356-366.
doi: 10.1016/j.icarus.2006.12.007

Martens, H. R. *et al.* (2008), Observations of molecular oxygen ions in Saturn's inner magnetosphere, *Geophys. Res. Lett.*, 35, L20103,
doi:10.1029/2008GL035433

20.1.5 IBS

Crary, F.J. *et al.* (2009), Heavy ions, temperatures and winds in Titan's ionosphere: Combined Cassini CAPS and INMS observations, *Planetary and Space Science*, **57**, 1847-1856.
doi: 10.1016/j.pss.2009.09.006

21 References

List of references used throughout the text follow, however section 20.1 includes other references of use for higher order products not discussed in this document.

21.1 CAPS Instrument Paper

Young, D.T. et al. (2004), Cassini Plasma Spectrometer Investigation, *Space Science Reviews*, 114, 1-112.
doi: 10.1007/s11214-004-1406-4

21.2 CAPS PDS file format description (SIS)

Furman, J., et al., 'CAPS standard data products and archive volume software interface specification'. [FURMANETAL2005]
PDS volume: COCAPS_1SAT > DOCUMENTS > CAPS_SIS

21.3 Others

Baumjohann, W., and R. A. Treumann (1996), Basic space plasma physics, p. 329. London: Imperial College Press
NASA ADS Bibliographic code: 1996bspp.book.....B

Crary, 2010, Reaching Consensus on IMS Calibration, CAPS Team Meeting #41

Hill, T. W., et al. (2012), Charged nanograins in the Enceladus plume, *J. Geophys. Res.*, *117*, A05209,
doi:10.1029/2011JA017218

Johnstone, A. D. et al. (1987), The giotto three-dimensional positive ion analyser, Journal of Physics E – Scientific Instruments (ISSN 0022-3735), 20, 795,
doi: 10.1088/0022-3735/20/6/038.

Lewis, G.R. *et al.* (2010), The calibration of the Cassini-Huygens CAPS Electron Spectrometer, *Planetary and Space Science*, *58*, 427-436.
doi: 10.1016/j.pss.2009.11.008

Paschmann, G. and Daly, P.W. (1998), Analysis Methods for Multi-Spacecraft Data. ISSI Scientific Reports Series SR-001, ESA/ISSI, Vol. 1. ISBN 1608-280X, 1998, ISSI Scientific Reports Series, 1.
http://www.issibern.ch/PDF-Files/analysis_methods_1_1a.pdf

Wilson, R. J. et al. (2012), Kelvin-Helmholtz instability at Saturn's magnetopause: Cassini ion data analysis, *J. Geophys. Res.*, *117*, A03212,
doi: 10.1029/2011JA016723