The Voyager PLS Utility Software

September 1, 2023

(Equations are created using *MathType*: *https://www.wiris.com/en/mathtype/*)

Contents

1. *historic_libraries* 2
2. *vgr_read* 2
3. *telemetry_mstart* 6
4. *cruise_list_ssedr* 7
5. *summary_pls_read* 7
6. *summary_mag_read* 8
7. *ISM_roll_analysis* 8
8. *spc_file_from_v12all_file* 9
9. *spectrograms_from_spc_file_TS* 9
10. *explore_sum* 11
11. *nxtedr_read* 11
12. *chkfact* 12
13. *convert_vgr_binary_ascii* 12

This document is an explanation of some of the software utility routines used to handle the data and trajectory information from the Voyager Plasma Science Experiment (PLS). This document is a complement to two other documents: *Voyager Interstellar: A Description of the Voyager Plasma Science Instrument and Data* (referred to as *VI-PLS* below) and *MJSANL: The Positive-Ion Non-Linear Fitting Routine for the Voyager PLS Instrument.*

The headings below refer to directories in the directory *Voyager_PLS_v_01/software_v_01/*.

1. historic_libraries

This directory contains code from Voyager software libraries at MIT.

Although none of this code will run on *CentOS*, it is useful for writing code on *CentOS* to replicate some of the routines from *Solaris*.

For example, our code to read plasma data from *SUMMARY* files on *CentOS*, which you can find in *Voyager_PLS_v_01/software_v_01/summary_pls_read*, was based on the historic code *nxtsum.f* in *historic_libraries/vgr_src_mjs*.

Many of the difficulties in converting code from Solaris to CentOS are related to the big/little endian problem, as you can see in the code in Case 5 below, and by examining the code in *software_v_01/summary_pls_read/nxtsum.f*

.f and *.c* files copied on 8/2/2023 from

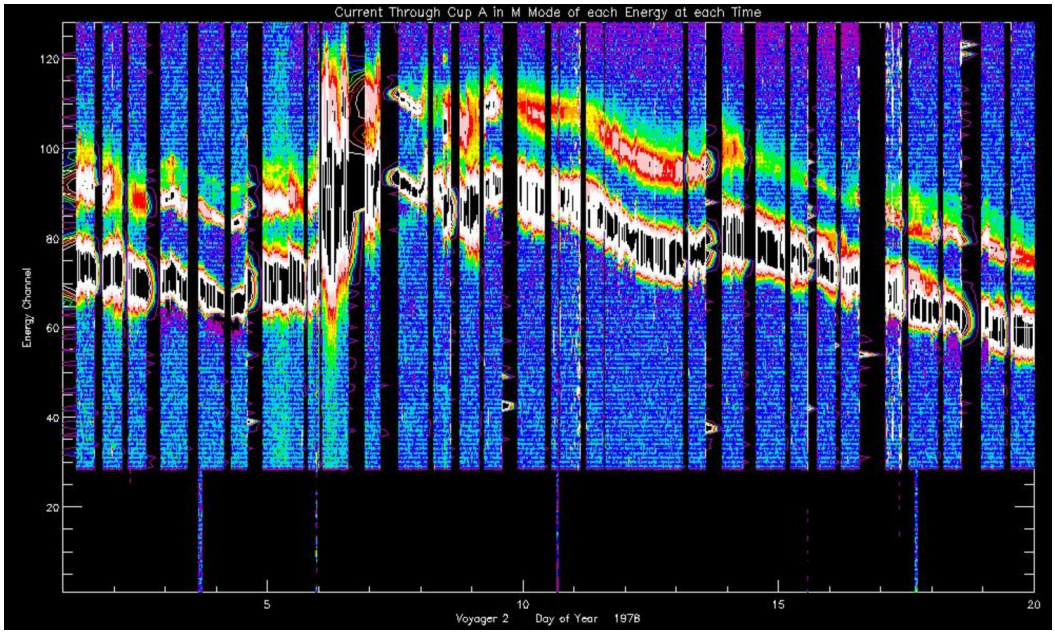
/nfs/carrington/h3/vgr/src/vgr
/nfs/carrington/h3/vgr/src/vec
/nfs/carrington/h3/vgr/src/pls
/nfs/carrington/h3/vgr/src/mjs
/nfs/carrington/h3/vgr/src/sedr

2. vgr_read

The *vgr* or *Mission* files are discussed in Section 7.3 of *VI-PI.* We motivate the use of a read routine for the *vgr* files by starting off with a particular objective and detailing the approach we use to achieve that objective.

We first want to examine cruise data from Voyager 2. We would like to find a Voyager 2 data interval where the correlation between field and velocity is high, to locate Alfvenic periods. We expect we will find those periods in high velocity streams, so the first thing we do is look at the color spectrograms that we can find in *Voyager_PLS_v_01/v2/spc/plots/Acup_Mmode* (see Section 7.4 of *VI-PLS*). The figure below is a spectrogram of the positive ion *M* mode measurements made by the A cup of the Voyager 2 plasma instrument from days 1 to 20 of 1978 (it is *v2_1978_001_020_Acup_Mmode.jpeg* from that directory). This data was taken in the 12 second period (see Section 3.6.2 of *VI-PLS*), and only channels 33-128 were returned. This color

spectrogram is produced from data in the *spc* file *v2_spc_1978_001_020.txt.gz* located in *Voyager_PLS_v_01/v2/spc/data*. Rather than read this *spc* data file, however, I typically go to the *vgr* files first. I can find the *vgr* file that contains this interval in *v2_vgr_1978.001.06_1978.120.23* in *Voyager_PLS_v_01/v2/vgr*.



We see a high velocity stream in doys 10 to 13 in the plot above, and I want to list the data for DOY 12 and plot some data for a few hours from DOY 12, 1978, using *gnuplot*. The *mjs.vgr* script in *vgr_read* is below, followed by the *main.f* program:

```
#!/bin/bash
# define data file using relative addressing
export ft32="../../v2/vgr/v2_vgr_1978.001.06_1978.120.23"
# these definitions are passed to main through a getenv call
# the files are opened in main after this getenv call
# make sure we have the most up to date copies of nxtvgrf.f and vgrtrd.f
# in Voyager_PLS_v_01/software_v_01/mjsanl/code
rm nxtvgrf.f
rm vgrtrd.f
cp ../mjsanl/code/nxtvgrf.f .
cp ../mjsanl/code/vgrtrd.f .
make
# execute the executable file
./lread > output
```

```
      INTEGER*2 JTB(6),JDATA(512),JNE,JTLMOD,JCLK
      character*4 ikinds
      character*64 ftdata
      common ikinds,mindex
      call getenv('ft32',ftdata)
      open(32,file=ftdata)
```

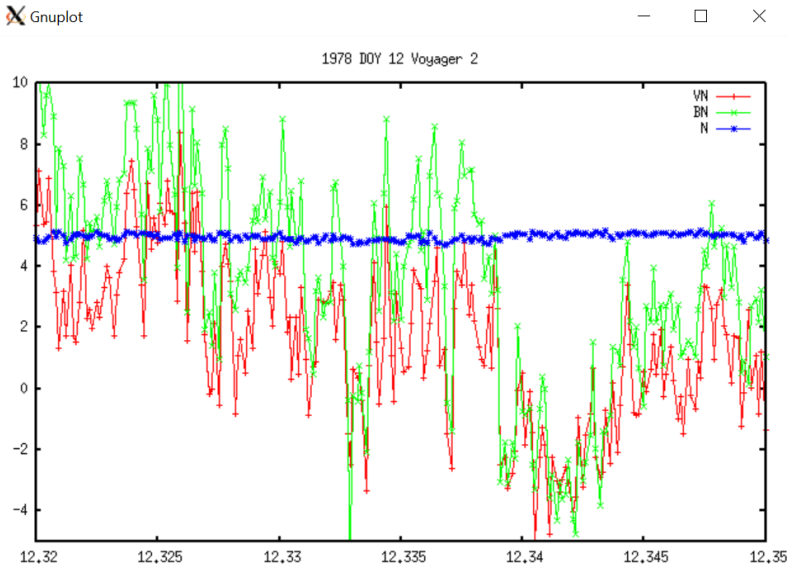
```
  1   call nxtvgrf(ANS,JTB,JDATA,JNE,LSTAT,JTLMOD,JCLK,TEMP,IEOD)
      if(ieod.ne.0) go to 10
      if(jtb(1).ne.1978) go to 1
      if(jtb(2).ne.12) go to 1
      if(jtb(2).gt.12) go to 10
      if(jtlmod.ne.2) go to 1
      decd = 1.*jtb(2)+(1.*jtb(3))/24.+(1.*jtb(4))/(24.*60.)
     * + (1.*jtb(5))/(24.*3600.)
      den = ans(38)
      vr = ans(99)
      vt = ans(100)
      vn = ans(101)
      w = ans(133)
      br = ans(96)
      bt = ans(97)
      bn = ans(98)
      vnp = vn+20.
      factor = 21.8/sqrt(den)
      bnp = -bn*factor
      denp = 5.*den
      write(6,15) decd,jtb,den,vr,vt,vn,w,br,bt,bn,vnp,bnp,denp
  15  format(f15.4,6I5,f8.3,11f8.2)
      if(ieod.eq.0) go to 1
  10  continue
      end
```

Note that in the main program we have calculated some additional variables, such as *vn+20*, *-bn*21.8/sqrt(den)*, and *5*den*, all of which are plotted with *gnuplot* below as *VN*, *BN*, and *N*. We produce an output file that has decimal day in it, and we plot data that file by going into the sub-directory *plot* and using the file */vgr_read/plot/data.p*. The file *data.p* is a *gnuplot* file, and is given below:

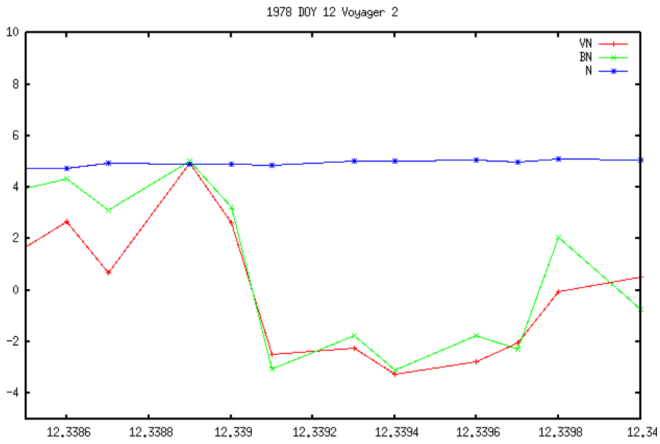
```
# Gnuplot script file for plotting data in file "output"
# This file is called  data.p
# load 'data.p'
set   autoscale                  # scale axes automatically
unset log                       # remove any log-scaling
unset label                      # remove any previous labels
set xtic auto                    # set xtics automatically
set ytic auto                    # set ytics automatically
set title "1978 DOY 12 Voyager 2"
set xlabel " "
set ylabel ""
set xr [12.32:12.35]
set yr [-5.0:10.]
plot    "../output" using 1:16  title 'VN' with linespoints,\
       "../output" using 1:17 title 'BN'  with linespoints, \
       "../output" using 1:18 title 'N' with linespoints
```

We produce a plot with the following sequence in the *plot* directory:

jwb@faraday[73] gnuplot
G N U P L O T
Version 4.6 patchlevel 2 last modified 2013-03-14
gnuplot> load "data.p"



I also modify *data.p* a bit to produce the expanded data plot below:



I was partcularly interested in this period because there is a "discontinuity" around a third of the way through the above plot, and I wanted to look at the 1.92 second magnetic field data at this time from the *summary* file (see Section 7.6 of *VI-PLS*) to see the change in more detail then the 12 second plasma data allow. However, although I have been sucessful at reading plasma data from a *summary* tape (see Case 5), I have not been as yet succeessful at reading the magnetic field data from a *summary* (see Case 6 and the accompanying discussion, although I continue to try).

The plots above show the behavior we expect from an Alfvenic period: relatively constant density, and a good correlation between the variation in *vn* and *-bn*V_{alfven}/B= -bn*21.8/sqrt(den).* This is a period of outward magnetic field, so the negative correlation implies outward Alfven wave propagation.

3. telemetry_mstart

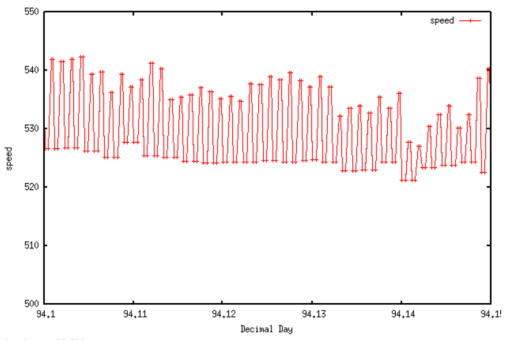
The spacecraft had a variety of telemetry modes over the mission. The *M* mode measurements used a variety of *M* start words over the mission. The M start words are used when the spacecraft is a telemetry mode such that there are not enough words available in the downlink to send back the full 128 channels of the *M* mode. Consider *GS-3*. In that state, the spacecraft could only send back 72 of the 128 channels. In the first of a cycle of two, it sends down measurements made in channels 1-72. In the second, it sends back another 72, but now the measurements made in channels 57 to 128 (128-57+1 – 72). The *M* start word is set to 1 in the first measurement, and 57 in the second measurement, so that you know which channels have been returned.

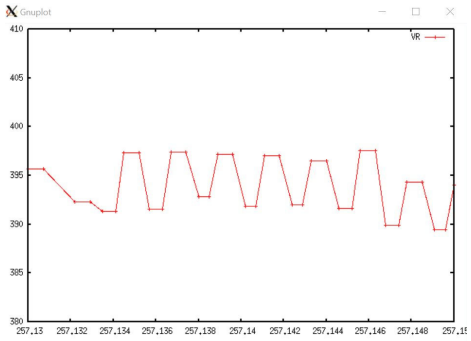
The routine in this directory allows you to dump the telemetry mode and the *M* start word from a Mission (*vgr*) file for a period set in *main.f*. There is a variable in *main.f*, *iprtmdn*. If this variable is 0, we only return the telemetry mode, the M start word, and other parameters, as in the following example.

```
GS-3  1 1979   94    1    5   43  584  2  2  EFEF0000
GS-3 57 1979   94    1    7   19  584  2  2  EFEF0000
GS-3  1 1979   94    1    8   55  584  2  2  EFEF0000
GS-3 57 1979   94    1   10   31  584  2  2  EFEF0000
```

This type of output is in the file *output_no_dn* in this directory. If *iprtmdn* =1, then we print out both this information and the difference between values for the *M* mode A cup for the current measurement minus those of the last measurement. If you examine the file *output_with_dn*, you will see that for a given measurement, either the lower 72 or the upper 72 are the same as in the last measurement, assuming they are contiguous in time.

This is because when you call *nxtmod* to get another *M* mode measurement, you are getting only 72 new channels, although you will get a full 128, partially filled from the last measurement. In the analysis we analyze all 128 channels, even though only 72 of them are recent. This leads to some puzzling artifacts in our analysis, for example at the two times below:





If you see behaviour like this the first thing to do is use this routine to look at what the *dn* values look like from one measurement to the next, to determine where the analysis is flawed. If you can see one *M* start word is consistently giving the incorrect anaysis, you can just chose not to return a measurement with that *M* start word, although this will halve your time resolution.

4. cruise_list_ssedr

This directory has a program which reads and lists some of the information from a short *sedr* (see Section 5.4 *Short SEDR File Format for Cruise* of VI-PLS). A typical read statement looks like the following. We list distances from the Sun in AU, along with the time of the navigation record.

```
      dimension itp(6),itn(6),rnav(56),rpt(9)
  5   read(33,1923,end=999) itn,rnav,itp,mod16,mod60,rpt
      write(6,1923) itn,rnav,itp,mod16,mod60,rpt
 1923 format(6i5,56E20.9,6i5,2i8,9E20.9)
```

The executable is *mjs.cru*. The *cruise ssedr* file you want to list is defined in that executable in an export statement. Executing the executable will result in the production of a file *output*, which lists the time and trajectory variable desired (in this case the radial distance from the sun in AU).

5. summary_pls_read

This directory contains a *CentOS* read routine to read and list *PLS* data from a binary summary tape as generated at *GSFC*. We read only one *M* mode spectra, at time 1977 304 4 59 12 for Voyager 2. This time is specified in the *main.f* routine. We chose this time because it is the time of one of our *mjsanl* fits, see Examples 01 and 04 in *MJSANL: The Positive-Ion Least-Squares Fitting Routine for the Voyager PLS Instrument.* For this time, we output the *DN* values and answer array values in the file *output*. The code is patterned after the historic *Solaris* code that you can find at *Voyager_PLS_v_01/software_v_01/historic_libraries/vgr_src_mjs/nxtsum.f.*

On a *summary file*, when read on *CentOS* from a binary file originally generated at GSFC, there are six kinds of records (see *documentation_v_01/general/Summary_tape_format.pdf*)*:*

AENG, ALFM, AM, L, AE1, AE2

These are engineering, magnetic field data, *M* mode data, *L* mode data, *E1* and *E2* mode data.

When we put the above characters out in hex and integer they are:
C5D5C740 D3C6D440 D4404040 D3404040 C5F14040 C5F24040 and
-741944256 -733986752 -750763968 -974045120 -973979584

We use the integer form to differentiate kinds of records in *nxtsum.f*, which calls *newreadbat.c*, which uses the standard fread to get data off the summary file. The file *nxtsum.f* has a lot of code to reorder the bytes we get from *fread*. This is necessary because of the big/little endian problem. This is also why we choose to use the integer representation of *AENG, ALFM, AM, L, AE1, AE2,* because of the difficulty of converting character variables between

different endian systems. We also convert the *real*4* answer array words on the summary tape from IBM 360 float to in IEEE 754, using a home-grown program *convert.f* based on https://en.wikipedia.org/wiki/IBM_hexadecimal_floating-point.

The entries in the answer array are described in *documentation_v_01/general/Ans_Array.pdf*, *Summary_tape_format.pdf*, *Summary_Tape_Format_Long.pdf* and *vgranl.pdf*, pages 41-50. Note that the *Summary_tape_format.pdf* list is incomplete. As far as I am aware, the listing in *Summary_Tape_Format_Long.pdf* is complete. There is also a list in *voyager_memos/pdf/VoyMemo_075_Voyager_Hourly_Average_Tapes.pdf* and in *VoyMemo_119_Corrected_Hourly_Average_Tape_Format.pdf.*

Note that my decoding of the answer array is partial in the following sense. When the answer array variable is a *real*4*, then I properly decode it. If it is not a *real*4*, then my decoding is wrong, but I think it is not too hard to get the correct decoding. I just have not had time enough to do it. As an example of what I mean, in *Summary_tape_format.pdf* look at page 1.15-5. The first three values of the answer array are of length 32 bits, but consist of two half integers in each of the 32 bit words. But I am decoding these 32 bits as if they were a *real*4*, which is clearly incorrect. And if you look at the answer array values in the file *software_v_01/summary_pls_read/output*, the values there make no sense. We could properly decode the *integer*2* variables in the first three words if we had the patience to, which I don't. Another example is answer array words 16-19, and so on.

6. summary_mag_read

Here is my attempt to write code in *CentOS* that will read magnetic field data off the summary tapes. I have so far not been successful, even though I have had success with reading plasma data off the summaries (see Case 5 above).

If I look in *documentation_v_01/Summary_tape_format.pdf* on page 1.15-4 I see the variables that should appear in the answer array for the magnetic field after the first 32 words in the header. Those 32 words are described on page 1.15-3. I have just taken all the words in the magnetic field answer array and converted them assuming they are *real*4*, as I did for the plasma data (see Case 5). This is 568 words or 2272 bytes, and according to the documentation the vast majority of these words are real*4 and should be magnetic field values for the different components and different averaging periods. But when I look at */software_v_01/summary_mag_read/ouput*, which lists all the mag answer arrays on day 12 of 1978, I see nothing that looks like what I expect, although this is essentially what I did when I was decoding the plasma data. Very rarely I see something that makes sense, for example word 12 of the header, which is supposed to be "Time period of the data block in seconds": 48 seconds is the value I expect. But everywhere else the values make no sense.

7. ISM_roll_analysis

This is the simulation that we use to help interpret the roll data in the interstellar medium, see Section 9 of *VI-PLS*. In the *main.f*. program we set the time of the roll, the mode (*L* or *M*), and the channels we sum over (usually 1 through 1 for the *L* mode and 1 through 8 for the *M* mode).

The example output in *output* is for the M mode summing over channels 1 through 8. We step through the roll in five-degree increments.
Note: even though some of the names of the routines here are identical to names of routine in *software_v_01/mjsanl/code*, **do not assume these routines are identical**.

The file *v2_shortsedrPredict* is a binary version of the Voyager 2 predict, we read it using the option (in *phase4.f*)

```
      open(33, file=
     * 'v2_shortsedrPredict',
     *    form='unformatted', convert='big_endian')
```

To run this code do a make and then ./test > output

8. spc_file_from_v12all_file

See Section 7.4 of VI_PLS. As of this date (September 1, 2023), Leslie Finck is continously producing new *V2all.txt* files as the data is put up at *JPL*, and then I turn them into the smaller *spc* files (the last transformed *v2all* files as of this date is *v2_spc_2023.229.10_2023.236.02.txt*). The way I produce new *spc* files is to copy any new *v2all* files from */nfs/faraday/data/vgr/v2/v2all_dir/v2all_files_for_restart* to *software_v_01/spc_file_from_v12all_file*. I then use those files to make a much smaller file containing the same information, by executing the script *script2p*. I *gzip* the resulting files and transfer them to *Voyager_PLS_v_01/v2/edr/2023*.

9. spectrograms_from_spc_file_TS

Our color spectrograms are made in *idl* using an *idl* routine *SpectraPlotterTenPlots.* Although color spectrograms are useful, they usually contain many days, and this limits their usefulness. Here I show how to get the better resolution color spectrograms on DOY 242 and 243 in 2007, days containing the first observations in the heliosheath and a termination shock crossing, assuming the user has *idl* installed on their system. Before I can use *SpectraPlotterTenPlots,* I have to expand an *spc* file to a much larger file format that *SpectraPlotterTenPlots* expects to see. In *spectrograms_from_spc_file_TS*, that process is done by the routine *readflat.f*. In that file we set the days we want expand (see `iyear=2007 idaystart = 240 idayend = 246`). In the executable *spc.read* we put in the location of the *spc* file we want to expand, in this case *v2/spc/data/v2_spc_2007_201_250.txt.gz*. We unzip that file in *spc.read*, and then when the executable *lread* is executed, it generates the file *v2all_2007_240_246.txt* in the format that *SpectraPlotterTenPlots* requires. We generate the color spectrograms by going into the subdirectory *plot*.

Note that *SpectraPlotterTenPlots* has the Voyager number hardwired. If you want to change that, go to line 31 and line 50 of *SpectraPlotterTenPlots* and make the appropriate changes.

We open *idl*, and issue the command

```
.compile SpectraPlotterTenPlots
```

And the command

```
SpectraPlotter10, '../v2all_1978_207_209.txt',Year=1978,startdate=208,enddate=209
```

Below is the output from *idl*

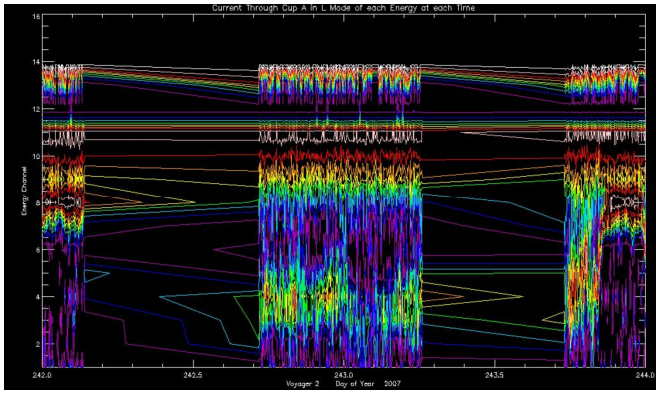
```
jwb@faraday[175] idl
IDL 8.8.0 (linux x86_64 m64).
(c) 2020, Harris Geospatial Solutions, Inc.

Licensed for use by: Massachusetts Institute of Technology
License: 94063
A new version is available: IDL 8.9
https://www.l3harrisgeospatial.com/Product-Downloads/

IDL> .compile SpectraPlotterTenPlots
% Compiled module: SPECTRAPLOTTER10.
IDL> SpectraPlotter10, '../v2all_2007_240_246.txt',Year=2007,startdate=242,enddate=244
infile:../v2all_2007_240_246.txt Number of Entries:                    3851
Year:    2007
DayofYear:Day of Year   2007
XTitle:Voyager 2     Day of Year   2007
YTitle:Energy Channel
startdate:242
FirstDay:242
enddate:244
LastDay:244
TitlePrefix:v2_2007_242_244_
OutFile:v2_2007_242_244_Acup_Lmode.jpeg
About to write OutFile:v2_2007_242_244_Acup_Lmode.jpeg
% Loaded DLM: JPEG.
About to write OutFile:v2_2007_242_244_Bcup_Lmode.jpeg
About to write OutFile:v2_2007_242_244_Ccup_Lmode.jpeg
About to write OutFile:v2_2007_242_244_Dcup_Lmode.jpeg
About to write OutFile:v2_2007_242_244_Acup_Mmode.jpeg
About to write OutFile:v2_2007_242_244_Bcup_Mmode.jpeg
About to write OutFile:v2_2007_242_244_Ccup_Mmode.jpeg
About to write OutFile:v2_2007_242_244_Dcup_Mmode.jpeg
About to write OutFile:v2_2007_242_244_Dcup_E1mode.jpeg
About to write OutFile:v2_2007_242_244_Dcup_E2mode.jpeg
IDL>
```

This produces 10 plots in the directory *plot*, eight of the four cups in *L* and *M* mode, *E1* and *E2*.

The plot below is the *A cup L mode* plot, which clearly shows that we are in the solar wind at the beginning of DOY 242, cross the Termination Shock for the first time in a data gap, are in the heliosheath for most of DOY 243, and then return to the solar wind at the end of DOY 243.



10. explore_sum

This directory has a program to take a summary file and list every kind of record with time, except for engineering data (see *documentation_v_01/general/Summary_tape_format.pdf* and *Summary_Tape_Format_Long.pdf*). Sample output looks like this:

```
B  data  1978   12    0   19   10  311
M  data  1978   12    0   19   58  952
E1 data  1978   12    0   20    8  311
M  data  1978   12    0   20   10  952
E1 data  1978   12    0   20   20  311
M  data  1978   12    0   20   22  952
E1 data  1978   12    0   20   32  311
M  data  1978   12    0   20   34  952
E1 data  1978   12    0   20   44  311
B  data  1978   12    0   19   58  311
M  data  1978   12    0   20   46  952
E1 data  1978   12    0   20   56  311
M  data  1978   12    0   20   58  952
E1 data  1978   12    0   21    8  311
M  data  1978   12    0   21   10  952
E1 data  1978   12    0   21   20  311
M  data  1978   12    0   21   22  952
E1 data  1978   12    0   21   32  311
```

This is a period of 12 second data for *PLS*, so we are returning only an *M* mode and a *E1* mode (see Section 3.6.2 of *VI-PLS*).

11. nxtedr_read

This directory has a program that reads on *CentOS* an *edr* file produced on Carrington. I am having trouble with this program on Faraday (CentOS) but strangely enough it runs fine on my ThinkPad under Fedora 38. On Faraday, the program is returning a segmentation fault, always around the same data file, counting from the beginning of the file, file 235. This is almost certainly a *c* memory allocation error. In this example, I am reading the *edr v2_edr_2007.241.21_2007.247.20*, which contains the termination shock crossing data. On Fedora I have no problem, on Faraday I get the segmentation error.

12. chkfact

This is a program to check the calculation of *fact* on all the encounter *spl* files (see Section 7.5 of *VI-PLS*). *Fact* is the constant *T* defined in 3.5.1.1 of *VI-PLS* for the threshold of the measurement state and is used in the conversion from DN values to femtoamps, see section 3.5.2 of *VI-PLS*. At one point I had thought I had computed it incorrectly, so I wrote *check.f* to make sure they were correct. The user specifies the *spl* file he wants to check in the executable script *mjs.fact*, and when that is executed, the program checks the calculation of *fact* for every mode on the file specified. At the end of the program, we output the number of disagreements. I have only shown the results for Neptune, see the keyboard session capture file *output_v2_nep*. I have done this for all six encounters with similar results (no problems).

13. convert_vgr_binary_ascii

This is code complied on Solaris. It will not compile on CentOS. I am including it for reference purposes only, so it is clear where the variables on my *vgr* ascii files come from. See *documentation_v_01/voyager_memos/pdf/VoyMemo_214_converting_VGR_files_from_binary_to_ascii.pdf*

14. mjsanl

This is the *mjsanl* code directory, see *MJSANL: The Positive-Ion Least-Squares Fitting Routine for the Voyager PLS Instrument*, *documentation_v_01/general/MJSANL_v_02.pdf.*