JPL Publication 14-8

Grid2: A Program for Rapid Estimation of the Jovian Radiation Environment

A Numeric Implementation of the GIRE2 Jovian Radiation Model for Estimating Trapped Radiation for Mission Concept Studies

Europa Clipper Pre-Project

R. W. Evans, Developer
D. E. Brinza, Editor
Jet Propulsion Laboratory

National Aeronautics and Space Administration

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

January 2014

This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

Table of Contents

1. Summary

Grid2 is a program that utilizes the Galileo Interim Radiation Electron model 2 (GIRE2) Jovian radiation model to compute fluences and doses for Jupiter missions. (Note: The iterations of these two softwares have been GIRE and GIRE2; likewise Grid and Grid2.) While GIRE2 is an important improvement over the original GIRE radiation model, the GIRE2 model can take as long as a day or more to compute these quantities for a complete mission. Grid2 fits the results of the detailed GIRE2 code with a set of grids in local time and position thereby greatly speeding up the execution of the model—minutes as opposed to days. The Grid2 model covers the time period from 1971 to 2050 and distances of 1.03 to 30 Jovian diameters (R_j). It is available as a direct-access database through a FORTRAN interface program. The new database is only slightly larger than the original grid version: 1.5 gigabytes (GB) versus 1.2 GB.

2. Introduction

GIRE2 (Ref. 1) has been out for use since 2012, but there has been no way to rapidly make estimates of the fluence and dose for orbit evaluations—complete mission runs can require as much as several days. The older GIRE had an approximate grid version (Ref. 2) developed for it that has been used for about 3 years and that has substantially increased in runtime efficiency (minutes versus hours). It was known that the new version, GIRE2, gives different results than GIRE, but the old Grid has been used until the present because it was the only thing available for rapid estimates.

Unlike GIRE, GIRE2 is dependent on the spacecraft local time (Sun–Jupiter–Spacecraft angle) through a parameter calculated from the Khurana magnetic field model (Ref. 3). Because of this local time dependence, implementing a grid fit for the new model requires the use of extra grids. Here, the new grids (13) necessary to approximate GIRE2 are combined into one file and are selected as needed by the grid program.

The Grid2 model is valid for a range from 1.03 to 30 R_j and for the period 1971 to 2050.

3. Implementation

Grid2 uses a “gridded” array of pre-calculated points and interpolates between these grid positions to estimate the fluence at a specified spacecraft position. The program finds eight points that surround the spacecraft position in a ‘cuboid’ (morphs to a cube if R_j, latitude, and West-longitude are plotted on a Cartesian axis). It then interpolates the flux to the spacecraft position to estimate the flux at the spacecraft.

Grid Size: Grid1 used a data file for the gridded array that is a direct access file (needed data can be accessed directly without reading the complete file to find it) to speed up data access that was 1.2 gigabytes (GB) in size. A straight estimate of the size of the direct access file for the new model was that it would be approximately 8 GB in size. Experimentation showed that the size could be reduced by using a grid in log (Range) instead of Range. The original grid had 300 grid points in Range for each latitude and West longitude pair. It was found that 100 grid points in log (Range) was an adequate number to achieve the desired accuracy. This gives tight groupings at small ranges where the model changes rapidly and a loose grouping where the model varies

slowly with range. Old and new versions use 2-degree spacing in latitude and 3-degree spacing in West longitude. Also, experimentation showed that the local time effects could be broken up into as few as 12 segments.

The grid has a core component range of 1–15 R_j where there are no local time variations. There are 12 "shells" in local time, and that has a range from 15 to 30 R_j. The 12 shells are each smaller than the core because the log (Range) variation is loosely packed there. All together, the core and shells add up to 1.5 GB.

Local Time: Because of the local time variations, the program needs to have the values for when the spacecraft is at a position so its local time can be calculated. For each spacecraft position, the program now reads in a date, an ephemeris time based on the J2000 international time standard (et), range, latitude, and West longitude. Date and et are redundant as covered below in Section 5, **Use of the Fortran Program**.

Since we are interested in the Sun's position, we need the Jupiter mean solar day, which (because of Jupiter's rapid rotation and long orbital period) is only 3 seconds longer than a Jupiter sidereal day. The sidereal day is 9 hr 55 min 30 s, and the solar day is 9 hr 55 min 33 s.

From the date and time, the program calculates an et and then calculates the Sun's sub-solar West longitude on Jupiter. The spacecraft sub-West Longitude minus the sub-solar West longitude is the spacecraft local time (in degrees), 15 degrees = 1 "Jupiter hour." A circular orbit is assumed for Jupiter, calculated a mean solar day for Jupiter and then estimated the solar sub-West longitude. As the actual Jovian orbit is an ellipse, however, Grid 2 then uses JPL's Space Planet Instrument C-Matrix Events(SPICE) (Refs. 4 and 5) to calculate the correct sub-solar West longitudes (or "true" solar day) (available for the period 1971 to 2050). From the two data sets, Grid2 applies a correction to add to the mean solar day to get the correct solar sub-West longitude for any date. The results are within ± 0.3 degrees of SPICE over the range of the above dates.

The et calculations are correct for 2000–2050 (until extra leap seconds are added), but 2 seconds are unaccounted for somewhere between 1971 and 2000. This not an issue here but will be corrected later. SPICE adds 10 leap seconds to the beginning of 1971 (end of 1970 really—they are accumulations from previous years) so the program does not go any further than 1971 in the past. Again, 10 seconds is probably not an issue here. Also, the residuals for my correction to mean solar day look like a noisy sum of sines and cosines, but a fit could not be found for them (see Appendix A, **Epicycles on Epicycles**). Although the program can be pushed past 2050, there may be accumulated errors (small if not pushed too far, say 10–20 years). The correction to mean solar day is fit to a [$\sin^2$] function, so it will not fail catastrophically if extrapolated past the fitting region. The fit does seems reasonably inaccurate for beyond the year 2050, but it does get significantly worse for years before 1971. As with Grid, Grid2 is assumed to be obsolete for a Jupiter mission planned to operate much beyond 2050.

4. Results

The program has been tested on two Europa Clipper trajectories supplied by the project, 11-F5A and 13-F7, at 10-minute resolution and 11-F5A at 1-minute resolution. Table 1 shows the resulting trapped particle fluences corresponding to GIRE and Grid2 for 11-F5A. Table 2 shows similar results for 13-F7. The results for GIRE and Grid2 are within a few percent of each other

for each trajectory. Also the results are similar for the two trajectories, so further discussion will be for trajectory 13-F7 only.

Table 3 shows the results for fluence for trajectory 13-F7 using Grid, which is based on the old GIRE model and which is in use by the Europa Clipper pre-project. Note that Grid produces smaller fluences for electrons for energies less than 20 MeV and larger fluences for energies greater than 20 MeV (approximately). Also Grid1 produces fluences for energies only up to 100 MeV. Fluences in Grid2 above that energy are log-log extrapolations (actually, fluences in both grids and at energies above 30 MeV are strictly extrapolations and should be only used with caution!!). Protons are unchanged between Grid and Grid2 (and their associated GIRE models). That is, they are unchanged from the Divine-Garrett model (Ref. 6). Figure 1 shows the particle fluences from Grid2, GIRE2, and Grid1 as listed in Tables 2 and 3, Protons in GIRE and GIRE2.

Table 4 lists the results for dose using the NOVICE software package (Ref. 7) for Grid2, GIRE2, and Grid1. Figure 2 shows the total dose for Grid2, GIRE2, and Grid1. The new Grid2 (and GIRE2) produces 33% higher doses at 100 mils (2.5 mm) aluminum shielding in the original Grid model. The differences in doses are within 10% for vault level shielding (~800 mils, 20 mm). The last percent entry in the dose table is probably not accurate because Grid did not go as high in energy as Grid2 does.

There are significant time savings in using Grid2 as opposed to GIR2 to generate radiation inputs to transport-analysis codes, such as NOVICE.

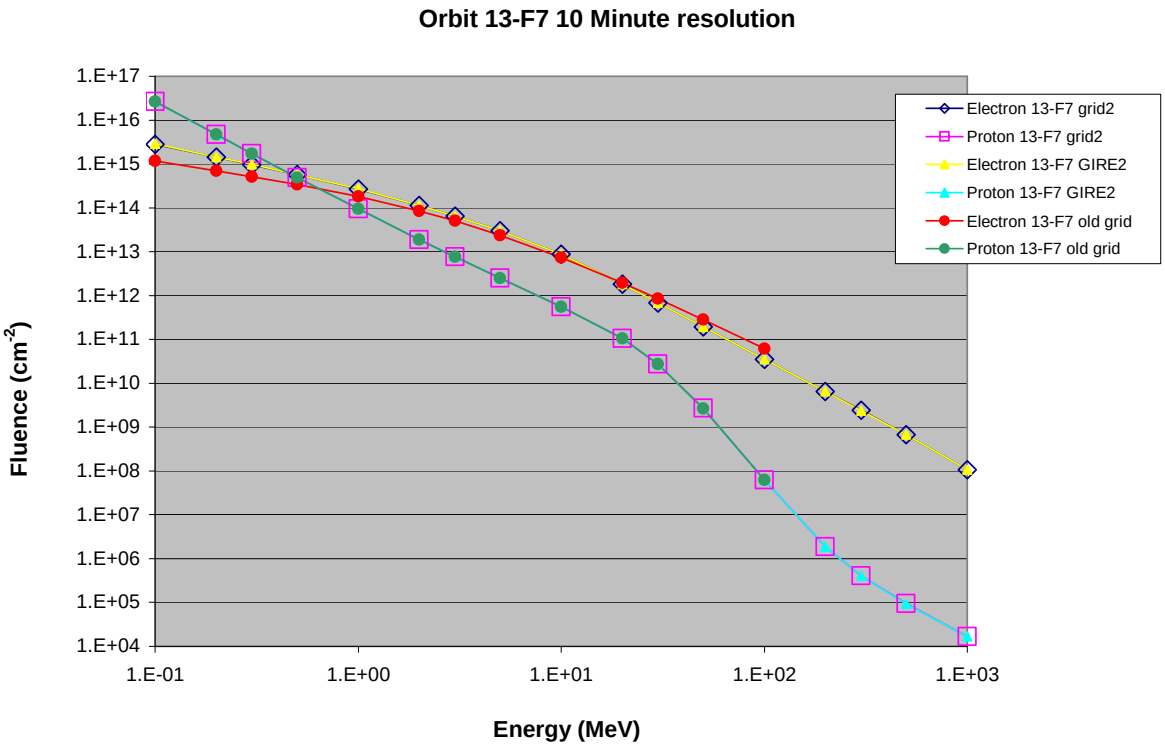

Figure 1. Particle Fluences from Grid2, GIRE2, and Grid (old Grid).

Table 1. Comparison of GIRE2 and Grid2 for Particle Fluences for Trajectory 11-F5A.

Energy (MeV)	GIRE2 Electrons Integral Fluence (cm^{-2})	11-F5A Protons Integral Fluence (cm^{-2})	Grid2 Electrons Integral Fluence (cm^{-2})	11-F5A Protons Integral Fluence (cm^{-2})	% Difference Electrons	% Difference Protons
0.1	2.93E+15	3.17E+16	2.91E+15	3.25E+16	-0.65	2.43
0.2	1.47E+15	5.29E+15	1.47E+15	5.36E+15	-0.34	1.25
0.3	9.87E+14	1.90E+15	9.82E+14	1.91E+15	-0.51	0.63
0.5	5.88E+14	5.33E+14	5.86E+14	5.35E+14	-0.36	0.34
1	2.76E+14	9.94E+13	2.75E+14	9.93E+13	-0.36	-0.10
2	1.17E+14	1.97E+13	1.17E+14	1.96E+13	-0.17	-0.41
3	6.72E+13	7.88E+12	6.73E+13	7.86E+12	0.15	-0.29
5	3.16E+13	2.58E+12	3.18E+13	2.57E+12	0.47	-0.43
10	9.38E+12	5.87E+11	9.44E+12	5.87E+11	0.68	0.00
20	1.95E+12	1.13E+11	1.96E+12	1.14E+11	0.51	0.62
30	7.17E+11	2.97E+10	7.22E+11	3.00E+10	0.64	0.91
50	2.03E+11	2.93E+09	2.04E+11	2.97E+09	0.54	1.23
100	3.71E+10	6.92E+07	3.73E+10	7.03E+07	0.62	1.53
200	6.85E+09	2.10E+06	6.89E+09	2.14E+06	0.63	2.10
300	2.55E+09	4.56E+05	2.56E+09	4.64E+05	0.51	1.82
500	7.22E+08	1.05E+05	7.26E+08	1.07E+05	0.61	1.52
1000	1.15E+08	1.86E+04	1.16E+08	1.89E+04	0.52	1.34

Table 2. Comparison of GIRE2 and Grid2 for Particle Fluences for Trajectory 13-F7.

Energy (MeV)	GIRE2 Electrons Integral Fluence (cm^{-2})	13-F7 Protons Integral Fluence (cm^{-2})	Grid2 Electrons Integral Fluence (cm^{-2})	13-F7 Protons Integral Fluence (cm^{-2})	% Difference Electrons	% Difference Protons
0.1	2.87E+15	2.64E+16	2.83E+15	2.68E+16	-1.39	1.67
0.2	1.45E+15	4.66E+15	1.43E+15	4.70E+15	-1.45	0.92
0.3	9.69E+14	1.71E+15	9.59E+14	1.72E+15	-1.05	0.64
0.5	5.77E+14	4.92E+14	5.72E+14	4.93E+14	-0.87	0.28
1	2.70E+14	9.40E+13	2.68E+14	9.40E+13	-0.81	-0.05
2	1.14E+14	1.89E+13	1.13E+14	1.89E+13	-0.88	0.00
3	6.49E+13	7.63E+12	6.46E+13	7.61E+12	-0.45	-0.30
5	3.02E+13	2.49E+12	3.01E+13	2.48E+12	-0.30	-0.24
10	8.80E+12	5.59E+11	8.82E+12	5.59E+11	0.17	-0.04
20	1.83E+12	1.05E+11	1.83E+12	1.06E+11	0.05	0.48
30	6.77E+11	2.72E+10	6.77E+11	2.74E+10	-0.03	0.88
50	1.92E+11	2.66E+09	1.92E+11	2.68E+09	0.05	0.86
100	3.51E+10	6.23E+07	3.52E+10	6.29E+07	0.14	0.93
200	6.46E+09	1.87E+06	6.48E+09	1.89E+06	0.23	1.07
300	2.39E+09	4.03E+05	2.40E+09	4.08E+05	0.50	1.24
500	6.75E+08	9.29E+04	6.79E+08	9.42E+04	0.58	1.37
1000	1.07E+08	1.66E+04	1.08E+08	1.68E+04	0.56	1.27

Table 3. Results of Grid1 for Particle Fluences for Trajectory 13-F7.

Grid1	13-F7		Grid2 - Grid1	
Energy	Electrons	Protons		
	Integral	Integral	%	%
	Fluence	Fluence	Difference	Difference
	(cm^{-2})	(cm^{-2})	Electrons	Protons
1.00E-01	1.17E+15	2.65E+16	59.23	-0.49
2.00E-01	6.96E+14	4.67E+15	51.99	-0.21
3.00E-01	5.12E+14	1.71E+15	47.15	-0.18
5.00E-01	3.40E+14	4.92E+14	41.04	-0.04
1.00E+00	1.81E+14	9.40E+13	32.93	0.04
2.00E+00	8.45E+13	1.89E+13	25.88	-0.16
3.00E+00	5.01E+13	7.62E+12	22.87	0.08
5.00E+00	2.36E+13	2.49E+12	21.75	0.08
1.00E+01	7.29E+12	5.59E+11	17.13	0.07
2.00E+01	1.94E+12	1.05E+11	-6.17	-0.19
3.00E+01	8.48E+11	2.73E+10	-25.21	-0.29
5.00E+01	2.85E+11	2.66E+09	-48.65	-0.15
1.00E+02	6.20E+10	6.23E+07	-76.64	-0.02

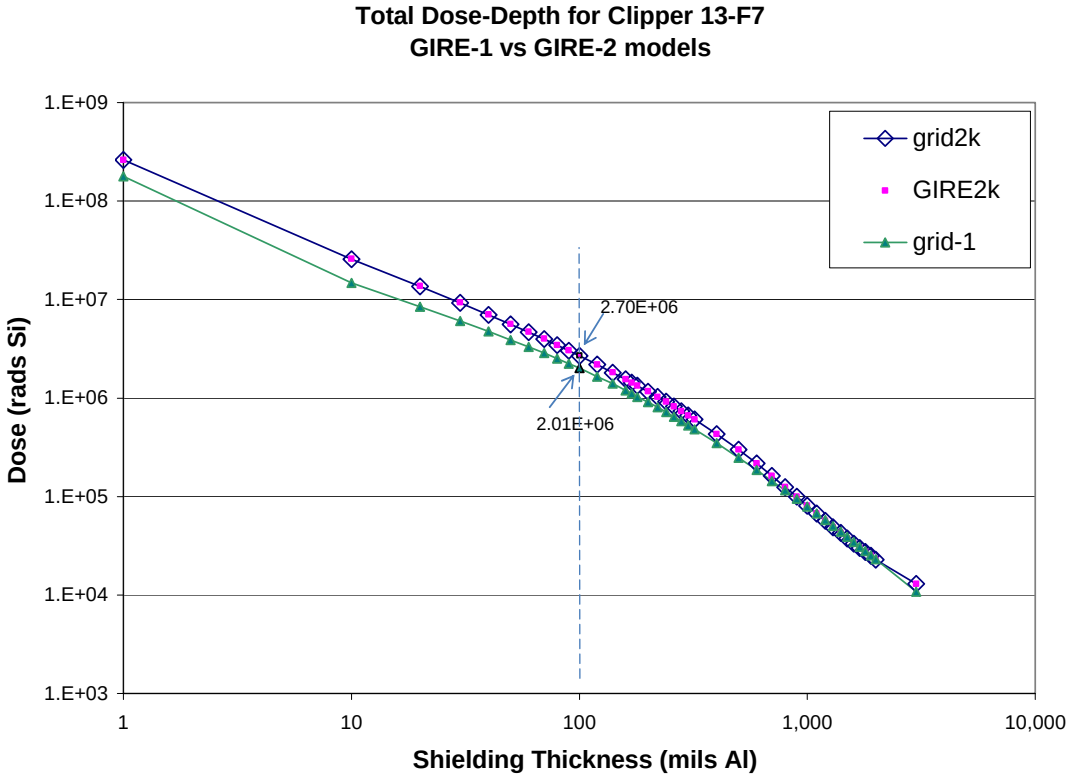

Figure 2. Dose/Depth Curve for Trajectory 13-F7. Old and New Results for 100 mils (2.5 mm) of Aluminum Shielding Are Shown.

Table 4. Total Dose for Trajectory Using Grid2, GIRE2, and Grid.

Shielding Thickness (Aluminum)			Total Dose			%change
g/sq. cm	mm	mils	Grid2	GIRE2	Grid1	Grid2-Grid1
6.86E-03	2.54E-02	1.00E+00	2.60E+08	2.61E+08	1.79E+08	45.50
6.86E-02	2.54E-01	1.00E+01	2.57E+07	2.60E+07	1.47E+07	74.92
1.37E-01	5.08E-01	2.00E+01	1.36E+07	1.37E+07	8.41E+06	61.37
2.06E-01	7.62E-01	3.00E+01	9.25E+06	9.32E+06	6.05E+06	52.97
2.74E-01	1.02E+00	4.00E+01	7.00E+06	7.06E+06	4.75E+06	47.47
3.43E-01	1.27E+00	5.00E+01	5.61E+06	5.65E+06	3.90E+06	43.64
4.12E-01	1.52E+00	6.00E+01	4.65E+06	4.69E+06	3.31E+06	40.28
4.80E-01	1.78E+00	7.00E+01	3.96E+06	3.99E+06	2.87E+06	38.16
5.49E-01	2.03E+00	8.00E+01	3.43E+06	3.46E+06	2.52E+06	35.84
6.17E-01	2.29E+00	9.00E+01	3.02E+06	3.05E+06	2.24E+06	34.53
6.86E-01	2.54E+00	1.00E+02	2.68E+06	2.70E+06	2.01E+06	33.54
8.23E-01	3.05E+00	1.20E+02	2.18E+06	2.19E+06	1.66E+06	31.61
9.60E-01	3.56E+00	1.40E+02	1.81E+06	1.82E+06	1.40E+06	29.55
1.10E+00	4.06E+00	1.60E+02	1.55E+06	1.55E+06	1.19E+06	29.58
1.17E+00	4.32E+00	1.70E+02	1.43E+06	1.43E+06	1.11E+06	29.06
1.23E+00	4.57E+00	1.80E+02	1.33E+06	1.33E+06	1.03E+06	29.43
1.37E+00	5.08E+00	2.00E+02	1.16E+06	1.17E+06	9.08E+05	27.75
1.51E+00	5.59E+00	2.20E+02	1.03E+06	1.03E+06	8.02E+05	28.02
1.65E+00	6.10E+00	2.40E+02	9.14E+05	9.17E+05	7.17E+05	27.44
1.78E+00	6.60E+00	2.60E+02	8.18E+05	8.20E+05	6.43E+05	27.17
1.92E+00	7.11E+00	2.80E+02	7.35E+05	7.37E+05	5.80E+05	26.65
2.06E+00	7.62E+00	3.00E+02	6.65E+05	6.67E+05	5.27E+05	26.32
2.20E+00	8.13E+00	3.20E+02	6.05E+05	6.06E+05	4.80E+05	25.93
2.74E+00	1.02E+01	4.00E+02	4.30E+05	4.30E+05	3.47E+05	24.08
3.43E+00	1.27E+01	5.00E+02	2.98E+05	2.98E+05	2.47E+05	20.39
4.12E+00	1.52E+01	6.00E+02	2.17E+05	2.17E+05	1.86E+05	16.91
4.80E+00	1.78E+01	7.00E+02	1.62E+05	1.62E+05	1.44E+05	12.72
5.49E+00	2.03E+01	8.00E+02	1.25E+05	1.25E+05	1.16E+05	7.95
6.17E+00	2.29E+01	9.00E+02	9.94E+04	9.93E+04	9.47E+04	4.97
6.86E+00	2.54E+01	1.00E+03	8.10E+04	8.10E+04	7.93E+04	2.09
7.54E+00	2.79E+01	1.10E+03	6.72E+04	6.72E+04	6.73E+04	-0.12
8.23E+00	3.05E+01	1.20E+03	5.67E+04	5.68E+04	5.78E+04	-1.85
8.92E+00	3.30E+01	1.30E+03	4.88E+04	4.88E+04	5.03E+04	-2.90
9.60E+00	3.56E+01	1.40E+03	4.25E+04	4.25E+04	4.41E+04	-3.65
1.03E+01	3.81E+01	1.50E+03	3.75E+04	3.75E+04	3.90E+04	-3.74
1.10E+01	4.06E+01	1.60E+03	3.33E+04	3.34E+04	3.47E+04	-3.92
1.17E+01	4.32E+01	1.70E+03	3.01E+04	3.01E+04	3.11E+04	-3.06
1.23E+01	4.57E+01	1.80E+03	2.73E+04	2.73E+04	2.80E+04	-2.60
1.30E+01	4.83E+01	1.90E+03	2.49E+04	2.49E+04	2.54E+04	-1.85
1.37E+01	5.08E+01	2.00E+03	2.29E+04	2.29E+04	2.31E+04	-0.95
2.06E+01	7.62E+01	3.00E+03	1.30E+04	1.30E+04	1.09E+04	19.41

Table 5 shows a benchmark test of the models. Numbers in red compare the times to run Grid2 and GIRE2. Other numbers are estimates for running programs to complete the task.

Table 5. Benchmark.

Benchmark assumed 1 CPU at 3.5 GHz				
	GIRE2		Grid2	
Trajectory	11-F5A	13-F7	11-F5A	13-F7
Lines	1,308,205	187,755	1,308,205	187,755
Render trajectory	5 minutes	1 minute	5 minutes	1 minute
Compute B & L	16 days	20 hours	N/A	N/A
Run Fluences	4 hours	54 minutes	94 seconds	19 seconds
Run Novice	15 minutes	15 minutes	15 minutes	15 minutes
Total	16 days	1 day	21 minutes	16 minutes

5. Use of the FORTRAN Program

Since a date is used in this program, it is not necessary to use uniform time steps. The difference in time (ΔT) time step between orbit points is used to accumulate the fluence. Because of this, the first orbit position does not contribute to the fluence because there is no ΔT associated with it. Also times (et) when the spacecraft is out of range are calculated so the last point out of range and the first point in range of the model produce a ΔT for that first point in range. Out-of-range and in-range crossings happen every time the spacecraft crosses 30 R_j—the outer boundary of the Grid2 model.

Supplied orbits should not have large gaps when the range exceeds 30 R_j because this will produce an incorrect ΔT. A work-around for this situation is to have a supplied large gap but with one data listing outside the 30 R_j boundary with a reasonable et next to a data listing inside 30 R_j.

Because of the date, the data files cannot be read with a free format style (read(*,*)). As implemented, the READ format in FORTRAN is:

```
1     READ(10,401,end=2001)date,et,rj,alat,wlong
401   format(a25,1x,f13.2,3(1x,e13.7))
```

Note that, as is, the program reads both date and et (so one does not have to change how one renders obits from SPICE kernels). et is ephemeris time based on J2000. If et is not supplied, use this format statement:

```
1     READ(10,301,end=2001)date,rj,alat,wlong
301   format(a25,3(1x,f7.3))
```

Or supply “ 0.00” for et (14 characters) with format statement 401 above. No et or a 0.00 et is considered ‘not supplied’. Also if et is supplied,

```
1     READ(10,*,end=2001) et,rj,alat,wlong
```

can be used to allow free formatting. et is not calculated if supplied. The advantages of supplying et is that it avoids the missing 2 seconds between 1971 and 2000, it will automatically add future leap seconds, and it speeds up the program a bit.

An attempt to convert the direct access Grid2 file to binary was made because it would have reduced the file size to ~600 Mb from 1.5 GB. Unfortunately, this was unsuccessful. There appear to be compiler-unique problems with direct access files—indeed, it's not clear that a direct-access file can be written as a binary, at least in the compilers we have used. Each compiler ignored the "FORM='unformattted' statement and wrote an ASCII direct-access file, some with extra blank lines (and 2–3 times the size of the original). One can produce binary files for the regular data files in g77, gfortran, and Intel Fortran. G77 and Intel FORTRAN can read each other's binaries, even across the Windows (DOS) and Linux platforms. However, gfortran does not read other compilers' binaries and does not produce binaries that can be read by other compilers.

Perhaps writing direct-access files as binary files is contradictory, like writing a binary file with a format statement.

Finally, FORTRAN program listings are provided in Appendix B. A sample record with explanation is provided in Appendix C, Sample Record in Gid2 Output File (jfluence). The binary file is available for distribution with the source code.

6. Conclusions

A Grid2 form of the latest GIRE2 model has been produced that can be added to the tools used by flight projects for rapid evaluation of trajectories about Jupiter for trapped-particle fluences. Program output is in the same format as in the original Grid1, so the dose tool produced for Grid1 should work. The dose tool might need additional response functions added because of the extended energy range of Grid2, or the project may choose to ignore the energies past 100 MeV produced by Grid2 (as values past 30 MeV for electrons are suspect at best), so the dose tool will work as is. Grid2 is based on GIRE2. Grid2 is valid from dates 1971 to 2050 for $R < 30\ R_j$. The program can be updated to different time periods without much work. It can also be updated to for changes in the GIRE model (e.g., protons extended past 12 R_j), again without much work (new grids are easily produced with the existing templates). Grid2 is as fast as Grid1 and produces results that agree with GIRE2 to a few percent.

7. References

1. H. B. Garrett, M. Kokorowski, I. Jun, and R. Evans, *Galileo Interim Radiation Electron Model Update—2012* [GIRE2], JPL Publication 12-9, Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, March 2012.
2. Robin Evans, *A Grid for Quick Estimates for Spacecraft Trajectories at Jupiter*, JPL IOM 5132-10-027 (to Insoo Jun) (internal document) , Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, July 4, 2010.
3. K. K. Khurana, "A Generalized Hinged-Magnetodisc Model of Jupiter's Nightside Current Sheet." *J. Geophys. Res.*, vol. 97, p. 6269–6276, 1992.
4. Charles .H. Acton, Jr., Ancillary data services of NASA's Navigation and Ancillary Information Facility, *Planet. Space Sci.*, vol. 44, pp. 65–70, 1996.
5. "SPICE: An Observation Geometry System for Robotic Space Science Missions," *NAIF Planetary Data System Navigation Node*, NASA website, Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, Dec. 30, 2013. http://naif.jpl.nasa.gov/naif/index.html (accessed Jan. 21, 2014)
6. N. Divine, N., and H. B. Garrett, "Charged Particle Distributions in Jupiter's Magnetosphere," *J. Geophys. Res.*, vol. 88, A9, pp. 6889–6903, Sept. 1, 1983.
7. Thomas Jordan, *NOVICE, A Radiation Transport Shielding Code*, Experimental and Mathematical Physics Consultants, Gaithersburg, Maryland, Jan. 2006 (original version was 1993). http://www.empc.com/ (accessed Jan. 21, 2014)

8. Appendices

A. Epicycles on Epicycles

Jupiter travels on an ellipse, not a circle, so mean solar time is not quite adequate for this study. In the days of Copernicus, astronomers kept adding epicycles to fix this. In modern days we decompose signals into sines and cosines.

The variation is almost 10 degrees from my mean solar time which was calculated for 1 Earth year only starting in January 2024 (tuned for the Europa Clipper). The variation has a fixed offset of a little more than 3 degrees. Removing a $\sin^2$ function with a fixed offset removes the major variations and leaves an approximate 0.3 degree "noisy" sine with a 0.2 degree increasing trend from 1971 to 2050. Removing the trend and a further sine term with a period of 5.93 years reduces the residuals to ±0.1 degree. As of this writing, these last two corrections have not been implemented, but the program has notes (and subroutines) on how to do this. We are unable to fit the final residuals to any function.

We notice that Saturn perturbs the solar day as Jupiter passes it (slows, speeds Jupiter in its orbit as they pass every 19.85838 years). The variation is a "pulse" and too small to effect this work. Figure A-1 shows the major variation of sub-solar West longitude and a fit to that variation. The Bottom axis is the ephemeris time et, and the drift and residuals are in degrees. Figure A-2 shows the trend removed from the residuals above. Figure A-3 shows the additional 5.92 year sine function and trend removed. Figures A-1 and A-2 are the resulting residuals after the above corrections. Fitted functions are in grey.

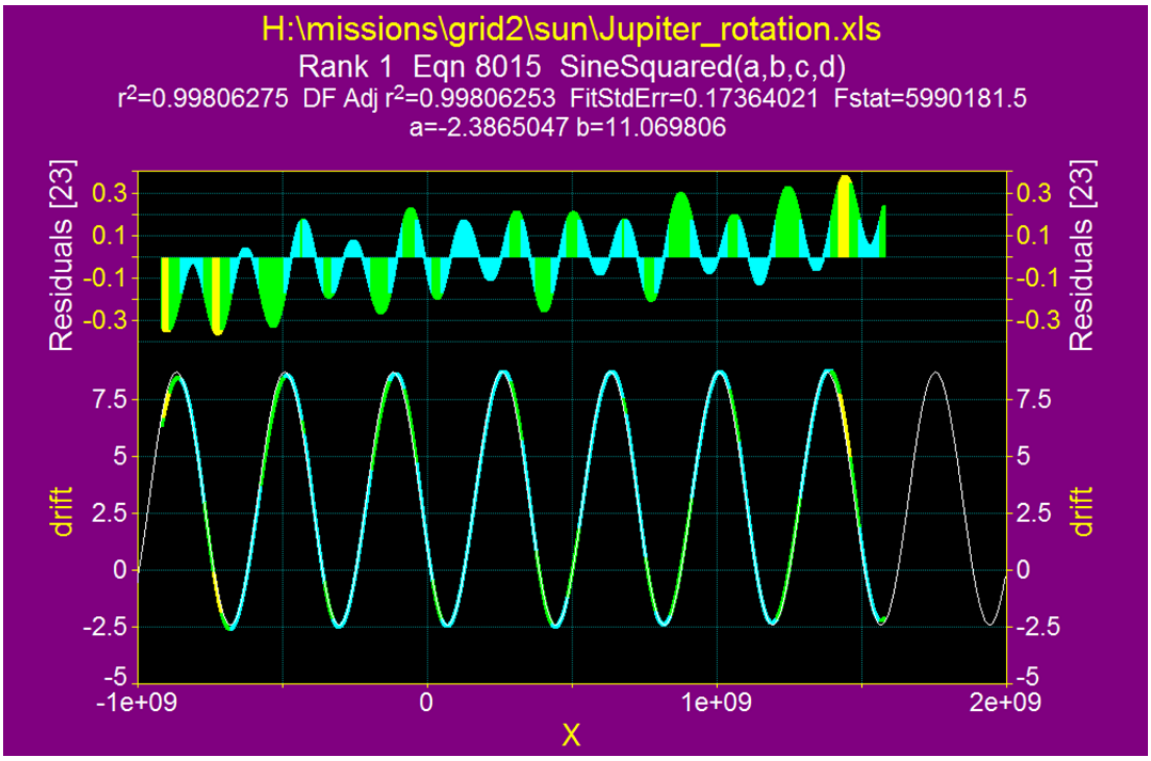

Figure A-1. Sine squared removed from mean solar time to approximate true solar time on Jupiter. The residuals curve is on top. The fixed offset is implemented in the current grid.

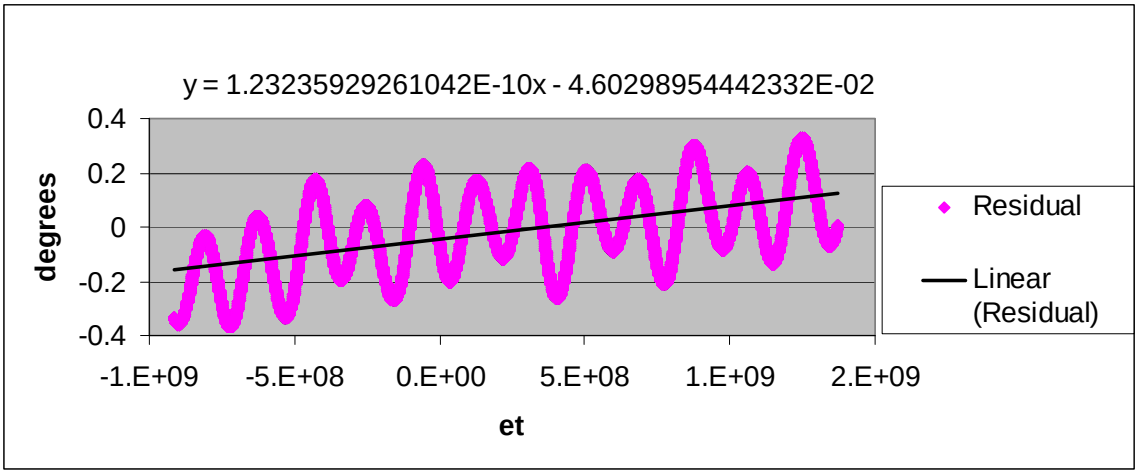

Figure A2. Trend extracted from residuals above to further approximate true solar time on Jupiter. Residuals are in the next figure. This could give higher accuracy, but it is probably not needed. And it has not been implemented as of this writing.

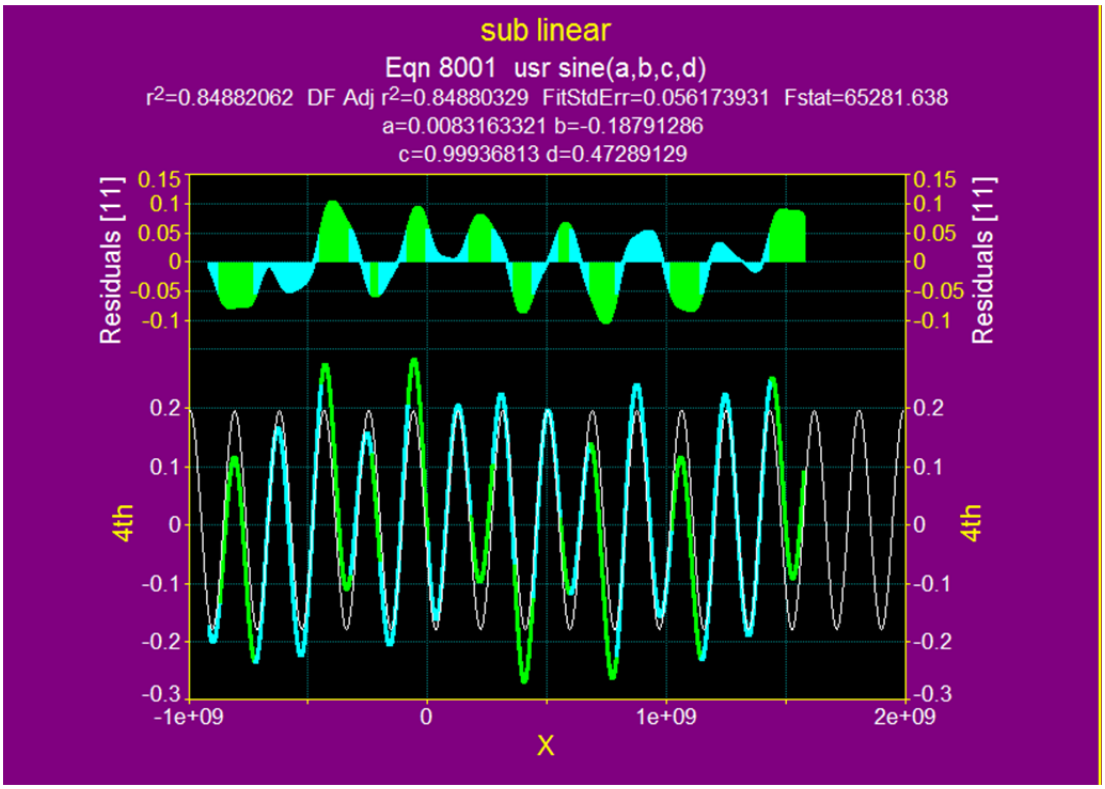

Figure A3. Sine removed from residuals in Figure A-2 to further approximate true solar time on Jupiter. Residuals are on top. This could also give higher accuracy, but it has not been implemented as of this writing.

The program, Table Curve 2D, was used to fit Figure A-1 and A-3 (with a user function). Excel was used to fit Figure A-2.

B. Gridgire2.f—FORTRAN Listing

Program reads

Date	Date in NAIF/Spice format YYYY MMM DD hh:mm:ss Example 2024 JAN 02 00:00:00.0000
Etx	ephemeris time J2000 (sec)
Rj	spacecraft range (R_j)
Alat	spacecraft latitude (deg)
Wlong	spacecraft West longitude (deg)

Options are

```
C 1    READ(10,*,end=2001)etx,rj,alat,wlong          ! program does not calculate et
C 1    READ(10,301,end=2001)date,rj,alat,wlong       ! program calculates et
 1    READ(10,401,end=2001)date,etx,rj,alat,wlong    ! default (reads orbits I produce) does not
                                                     !calculate et
```

Comment out the default and uncomment the one required in the program listing.

Output is a listing of fluences vs. energy

Output file from orbit 13-F7

Fluences estimated by use of a pre-calculated grid of points using

- integral flux from GIRE2 Garrett et al., JPL Publication 12-9 (Ref. 1)
- orbit ends at line 187755

Energy (MeV)	Electron Fluence Integral (cm^-2)	Proton fluence Integral (cm^-2)
1.000E-01	2.830E+15	2.684E+16
2.000E-01	1.429E+15	4.703E+15
3.000E-01	9.588E+14	1.721E+15
5.000E-01	5.720E+14	4.934E+14
1.000E+00	2.678E+14	9.395E+13
2.000E+00	1.130E+14	1.890E+13
3.000E+00	6.461E+13	7.607E+12
5.000E+00	3.011E+13	2.484E+12
1.000E+01	8.815E+12	5.588E+11
2.000E+01	1.831E+12	1.055E+11
3.000E+01	6.768E+11	2.744E+10
5.000E+01	1.921E+11	2.683E+09
1.000E+02	3.515E+10	6.288E+07
2.000E+02	6.475E+09	1.890E+06
3.000E+02	2.402E+09	4.080E+05
5.000E+02	6.789E+08	9.417E+04
1.000E+03	1.076E+08	1.681E+04

Refer to Appendix C for explanation of the yellow highlighted code areas below.

```
      program gridgire2
C Reads orbits with dates, RJ, ALAT, WLONG in format statement
C 301  format(a25,3(1x,f7.3)) or  dates, et, RJ, ALAT, WLONG in format
C statement  401  format(a25,1x,f13.2,3(1x,e13.7))
C 2024 JAN 02 00:00:00.0000 757425667.18 18.000 -45.000 310.000
C 0's after decimal can be left out of date but leave spaces. space and
C 6 characters for remaining numbers.  Use extra spaces for numbers that
C fall short of 6 characters and (-) sign counts as a character.
C Program will find last line of input file which should contain data
C (don't have a blank line at end). Error code is harmless if you forget
C the last instruction.  You can change the format line (301) to suit
C yourself. you will have to do this if you use negative west longitudes
C (-345.000 will not work as is - too many characters).
C
C Reads a direct access file for data.
C
C Dates good from 1971 to 2050.  Beyond 2050 it is probably ok but iffy
C (not tested). Before 1971, I do not know when the extra leap seconds
C fit in (10 dumped into 1971 in spice but they are accumulated from
C previous years). Until fixed, do not go before 1971 without expecting
C errors assumed to start at 10 seconds and increasing backwards in time.
C For R less than 15.1948, program ignores the date, for R greater than
C 15.1948 program uses the date to find the local time of the spacecraft
C to choose which records to use. Data file has a core of 873621 records
C for R less than 15.1948 and shells of 229341 records for each of the
C 12 bins in local time.
C
C Computes fluences from flux * delta T, time steps do not have to be
C uniform, uses ephemeris time (et) to calculate delta T.
C
C Note that if you have a large gap in the orbit the flux for the first
C point after the gap will be bogus, flux * LARGE delta T

      character*25  date
      character*70  line2,line3,line4,line5,line6
      character*418 line
      integer       ict(8),  ireca(14),  irec(8),    irecx
      dimension     alll(34),allh(34),   alhl(34),   alhh(34),ry(8)
      dimension     ahll(34),ahlh(34),   ahhl(34),   ahhh(34),ay(8,34)
      dimension     wlx(14), rbogus(100),e(17),      blaty(8)
      dimension     aly(8),  alay(8),    wly(8),     by(8),   beqy(8)
      dimension     bl(34),  bh(34),     cl(34),     ch(34),  flux(34)
      real*8        rdb(100), et,etx,    flue(34),   etold, rl,rh
C     SHORT Version
C     energies
      data e /  0.10,   0.20,   0.30,   0.50,    1.00,    2.00,    3.00,
     +          5.00,  10.00,  20.00,   30.00,   50.00, 100.00,  200.00,
     +        300.00, 500.00, 1000.00 /
C     real range data for db
      data rdb /
     +1.03459704D0,1.07039094D0,1.10742331D0,1.14573681D0,1.18537581D0,
     +1.22638619D0,1.26881552D0,1.31271267D0,1.35812867D0,1.40511572D0,
     +1.45372856D0,1.50402319D0,1.55605793D0,1.60989285D0,1.66559029D0,
     +1.72321475D0,1.78283274D0,1.84451342D0,1.90832806D0,1.97435033D0,
     +2.04265690D0,2.11332679D0,2.18644142D0,2.26208591D0,2.34034729D0,
     +2.42131615D0,2.50508642D0,2.59175491D0,2.68142176D0,2.77419114D0,
     +2.87016964D0,2.96946883D0,3.07220364D0,3.17849278D0,3.28845859D0,
     +3.40222955D0,3.51993656D0,3.64171576D0,3.76770830D0,3.89805937D0,
     +4.03292036D0,4.17244768D0,4.31680202D0,4.46615028D0,4.62066603D0,
     +4.78052664D0,4.94591856D0,5.11703253D0,5.29406643D0,5.47722530D0,
     +5.66672039D0,5.86277199D0,6.06560659D0,6.27545834D0,6.49257040D0,
```

```
     +6.71719408D0,6.94958782D0,7.19002295D0,7.43877649D0,7.69613600D0,
     +7.96239948D0,8.23787308D0,8.52287960D0,8.81774521D0,9.12281322D0,
     +9.43843555D0,9.76497555D0,10.1028166D0,10.4523439D0,10.8139601D0,
     +11.1880913D0,11.5751657D0,11.9756327D0,12.3899536D0,12.8186092D0,
     +13.2620945D0,13.7209234D0,14.1956263D0,14.6867533D0,15.1948662D0,
     +15.7205639D0,16.2644482D0,16.8271503D0,17.4093189D0,18.0116291D0,
     +18.6347790D0,19.2794857D0,19.9464989D0,20.6365891D0,21.3505535D0,
     +22.0892124D0,22.8534336D0,23.6440945D0,24.4621086D0,25.3084259D0,
     +26.1840210D0,27.0899105D0,28.0271397D0,28.9967957D0,29.9999981D0/
C     db range listings (low precision for space)
      data rbogus / 1.03,1.07,1.11,1.15,1.19,1.23,1.27,1.31,1.36,1.41,
     +              1.45,1.50,1.56,1.61,1.67,1.72,1.78,1.84,1.91,1.97,
     +              2.04,2.11,2.19,2.26,2.34,2.42,2.51,2.59,2.68,2.77,
     +              2.87,2.97,3.07,3.18,3.29,3.40,3.52,3.64,3.77,3.90,
     +              4.03,4.17,4.32,4.47,4.67,4.78,4.95,5.15,5.29,5.48,
     +              5.67,5.86,6.07,6.28,6.49,6.72,6.95,7.17,7.44,7.70,
     +              7.96,8.24,8.52,8.82,9.12,9.44,9.77,10.1,10.5,10.8,
     +              11.2,11.6,12.0,12.4,12.8,13.3,13.7,14.2,14.7,15.2,
     +              15.7,16.3,16.8,17.4,18.0,18.6,19.3,19.9,20.6,21.4,
     +              22.1,22.9,23.6,24.5,25.3,26.2,27.1,28.0,29.0,30.0/
C     record breaks for core, shells
      data ireca / 0,873600,1102920,1332240,1561560,1790880,2020200,
     +             2249520,2478840,2708160,2937480,3166800,3396120,
     +              873600/
C     local time breaks for shells lt = 30 (2 hr) used for 15-45 (1-3)
C     hr, lt = 60 (4 hr) used for 45-75 (3-5 hr), etc.
C     lt = 360 (0 hr) used for 345-15 (23 to 1 hr)
      data wlx  / 0.,  15.,    45.,    75.,   105.,   135.,   165.,
     +               195.,   225.,   255.,   285.,   315.,   345.,
     +               360./
C open trajectory file, output file and data file
      open(10,file='trajd.in',status='old')
      open(11,file='JFluencegridgire.out',status='unknown')
      open(12,FILE='JgridballintGIRESynch1-30.direct',STATUS='OLD',
     +ACCESS='DIRECT',RECL=418)
      n = 0
      do i = 1,34
        Flue(i) = 0.D0
      enddo
C process
      etx = 0.D0
C 1    READ(10,*,end=2001)et,rj,alat,wlong
C 1    READ(10,301,end=2001)date,rj,alat,wlong
 1    READ(10,401,end=2001)date,etx,rj,alat,wlong
      n = n + 1
      if(float(n/10000) .eq. float(n)/10000.)
     +                    print*,n,' ',date,' ',rj,alat,wlong
      irj = 0
      do i =  1,100
        if(rj .gt. rdb(i)) irj = i
      enddo
C find et (don't skip dates where Range is out of bounds to avoid LARGE
C delta T, use etx if available
      if(etx .eq. 0.D0) call xtime(date,et)
      if(etx .ne. 0.D0) et = etx
      if(n .eq. 1)etold = et

C if irj out of range read another record.
      if(irj .eq. 0. .or. irj .eq. 100) go to 1

      if(alat .eq. 90.) alat = alat - 0.0001 ! for proper integer below
      ial = int(alat + 90.)/2
      iwl = int(wlong)/3
```

```
C irec for bounding 'cuboid'
      irec(1) = iwl   + ial*120     + (irj-1)*10920 ! minr minal minwl
      irec(2) = iwl   + (ial+1)*120 + (irj-1)*10920 ! minr maxal minwl
      irec(3) = iwl+1 + ial*120     + (irj-1)*10920 ! minr minal maxwl
      irec(4) = iwl+1 + (ial+1)*120 + (irj-1)*10920 ! minr maxal maxwl
      irec(5) = iwl   + ial*120     +  irj*10920    ! maxr minal minwl
      irec(6) = iwl   + (ial+1)*120 +  irj*10920    ! maxr minal maxwl
      irec(7) = iwl+1 + ial*120     +  irj*10920    ! maxr maxal minwl
      irec(8) = iwl+1 + (ial+1)*120 +  irj*10920    ! maxr maxal maxwl
      if (iwl .eq. 0) then
        irec(1) = 120 + ial*120     + (irj-1)*10920 ! 360 --> 0
        irec(2) = 120 + (ial+1)*120 + (irj-1)*10920 ! 360 --> 0
        irec(5) = 120 + ial*120     +  irj   *10920 ! 360 --> 0
        irec(6) = 120 + (ial+1)*120 +  irj   *10920 ! 360 --> 0
      endif

      if(rj .gt. rdb(80)) then
        call jup(et,wlong,wlt0)
C -15 to 15, 15-45, etc.
        do i = 2,14
          if( wlt0 .ge. wlx(i-1) .and. wlt0 .lt. wlx(i))
     +        irecx= ireca(i) - 79 * 10920
        enddo
        do i = 1,8
          irec(i) = irec(i) + irecx
        enddo
      endif
      do i = 1,8
C read records, irec(i)
        read(12,REC=irec(i))line
        read(line,11) ict(i),ry(i),alay(i),wly(i),by(i),beqy(i),aly(i)
     +               ,blaty(i),(ay(i,j),j=1,34)
      enddo
C for fluence calculations
C     rtp        to keep order straight
      illl = 1
      ilhl = 2
      illh = 3
      ilhh = 4
      ihll = 5
      ihhl = 6
      ihlh = 7
      ihhh = 8
C 8 * 34 fluxes
      do i = 1,34
        alll(i) =ay(illl,i)
        allh(i) =ay(illh,i)
        alhl(i) =ay(ilhl,i)
        alhh(i) =ay(ilhh,i)
        ahll(i) =ay(ihll,i)
        ahlh(i) =ay(ihlh,i)
        ahhl(i) =ay(ihhl,i)
        ahhh(i) =ay(ihhh,i)
      enddo
C interpolate box to spacecraft position
C     Collapse longitudes  2 wl s
      call linear(wly(illl),wly(illh),wlong,alll,allh,bl)
      call linear(wly(ilhl),wly(ilhh),wlong,alhl,alhh,bh)
C     Collapse latitude 2 lats
         call linear(alay(illl),alay(ilhl),alat,bl,bh,cl)
C     Collapse longitudes  2 wl s
      call linear(wly(ihll),wly(ihlh),wlong,ahll,ahlh,bl)
```

```
      call linear(wly(ihhl),wly(ihhh),wlong,ahhl,ahhh,bh)
C     Collapse latitude
         call linear(alay(ihll),alay(ihhh),alat,bl,bh,ch)
C     r's in database at low resolution
            do i= 1,100
              if(ry(illl) .eq. rbogus(i)) rl = rdb(i)
              if(ry(ihhh) .eq. rbogus(i)) rh = rdb(i)
              if(rh .eq. rl) rh = rdb(i+1)
            enddo
C     collapse r           2 r s
            call linear2(rl,rh,rj,cl,ch,flux)
C Calculate Fluence from flux, delta-t
        do j = 1,34
          flue(j) = flue(j) + (et - etold) * dble(flux(j))
        enddo
        etold = et
      go to 1
 2001 continue
      ictold = n
      print*,'orbit ends at line ',ictold
C write output file
      line2 = ' Fluences estimated by use of a pre-caclulated grid of po
     +ints using   '
      write(11,70) line2
      line3 = ' integral flux from GIRE2. Garrett et al., JPL Publication 12-9
     +             '
      write(11,70) line3
      line3 = ' '
      write(11,70) line3
      line4 = '              Electron Fluence     Proton fluence
     +             '
      line5 = 'Energy        Integral             Integral
     +             '
      line6 = '(MeV)         (cm^-2)              (cm^-2)
     +             '
      write(11,*) ' '
      write(11,*) 'orbit ends at line ',ictold
      write(11,*) ' '
      write(11,70) line4
      write(11,70) line5
      write(11,70) line6
      do i = 1,17
        write(11,12) e(i), flue(i),flue(i+17)
        write(*,*) e(i), flue(i),flue(i+17)
      enddo
      write(11,*) ' '
 11   format(i8,41(1x,e9.3))
 12   format(1PE10.3,3x,1PE10.3,11x,1PE10.3)
 70   format(a70)
 301  format(a25,3(1x,f7.3))
 401  format(a25,1x,f13.2,3(1x,e13.7))
      end

C Single precision for Alat and Wlong
      subroutine linear(xl,xh,x,al,ah,s)
      dimension al(34),ah(34),s(34)
      do i = 1,34
         slope      = ( ah(i) - al(i) ) / (xh - xl)
         yintercept = al(i) - slope * xl
         s(i)       = slope * x + yintercept
C         print*,slope,yintercept,xh,xl,x
      enddo
      return
```

```
      end

C Double precision for Range
      subroutine linear2(xl,xh,x,al,ah,s)
      dimension al(34),ah(34),s(34)
      real*8 xl,xh,slope,yintercept
      do i = 1,34
         slope      = ( dble(ah(i)) - dble(al(i)) ) / (xh - xl)
         yintercept = dble(al(i)) - slope * xl
         s(i)       = slope * dble(x) + yintercept
      enddo
      return
      end

      subroutine JUP(et,wlsc,wlt0)
C     find WL of spacecraft WRT sun
      real*8 et ,d2,wls,et0
C Sun at WL = 0.0 at t0 = 875633641.5485 (2027 OCT 01 03:32:54.36615)
C Jupiter (mean) solar day is 35732.86816 sec
C order 0
      et0 = 875633641.5485D0
C      wls = mod((et - et0) * 360.D0/35732.86816D0,360.D0)
      wls = (et - et0)*870.452902871049D0/86400.D0 +0.02015D0
      wls = mod(wls,360.D0)
      if(wls .lt. 0.D0) wls = 360.D0 + wls
C order 1 - correct for eliptical orbit |residuals| < .3 deg)
C equ8015 wrt et, not delta et
      call  eqn8015(et,d2)
      wls = wls + dble(d2)
      wls = wls + 0.3D0
C FIX LATER
C     get rid of this term if beelow implemented
C     wls = wls + 0.3D0
C order 2 - looks like sum of sines and cosines, with a small linear term -
C will fix later but 0.3 degrees ok and my present routine will not do it.
C epicycles on epicycles ...
C can reduce residuals by hand fit to 0.1 deg
C 10 remove a linear trend
C wlt0 = wlt0  -( 1.23245536447971E-10*et - 0.0460284264937504
C then remove another sinw ith period  5.93158 years (see
C SUBROUTINE eqn8001(x,y) below
C equ8001 wrt et, not delta et
C     Call eqn 8001(et,d3)
      wls = wls + dble(d2)
      wls = wls + 0.3D0
      wls0  = real(wls)
      if(wls0 .ge. 360.) wls0 = 360. - wls0
      if(wls0 .ge. 360.) wls0 = 360. - wls0
      wlt0 = (wlsc - wls0)
      if(wlt0 .lt. 0.)   wlt0 = wlt0 + 360.
      if(wlt0 .gt. 360.) wlt0 = wlt0 - 360.
      return
      end


*-------------------------------------------------------------*
      SUBROUTINE eqn8015(x,y)
*-------------------------------------------------------------*
***** TableCurve H:\missions\Grid2\sun\eqn8015.f Jul 11, 2013 2:14:58 PM
***** H:\missions\Grid2\sun\Jupiter_rotation.xls
***** X= et
***** Y= drift
***** Eqn# 8015  SineSquared(a,b,c,d)
```

```
***** r2=0.998062750570461D0
***** r2adj=0.9980625284088292D0
***** StdErr=0.1736402125675777D0
***** Fval=5990181.452082999D0
***** a= -2.386504709810813D0
***** b= 11.06980649551408D0
***** c= 2.527084753103394D0
***** d= 748890926.9842513D0
***** Constraint: d<>0
*-----------------------------------------------------------*
      DOUBLE PRECISION x,y
      DOUBLE PRECISION n
      n=DSIN(6.283185307179586477 0D0*x/748890926.9842513D0+
     12.527084753103394D0)
      y=(-2.386504709810813D0)+11.06980649551408D0*n*n
      RETURN
      END

*-----------------------------------------------------------*
      SUBROUTINE eqn8001(x,y)
*-----------------------------------------------------------*
***** TableCurve C:\Users\rwe\Desktop\New_folder\missions\Grid2\sun\usrsine2.f Jul 24,
2013 9:19:52 PM
***** sub linear
***** X= et
***** Y= sl
***** Eqn# 8001  usr sine(a,b,c,d)
***** r2=0.8488206233876732D0
***** r2adj=0.8488032863031993D0
***** StdErr=0.05617393071274914D0
***** Fval=65281.63833772391D0
***** a= 0.008316332144591362D0
***** b= -0.187912858309531D0
***** c= 0.9993681331650221D0
***** d= 0.472891292232232D0
*-----------------------------------------------------------*
      DOUBLE PRECISION x,y
      TWOPI = 2.*PI

      Y = 0.008316332144591362D0 + 0.187912858309531D0 *
     +SIN(0.9993681331650221D0*TWOPI/187058311.284091 * X +
     +0.472891292232232D0)

      RETURN
      END

      subroutine xtime(xdate,sec)
C find et (sec)
      real*8       sec,nl(25),pl(3)
      character*25 xdate
      character*3  mon,month(12)
      dimension    mm(12)
      data month / 'JAN','FEB','MAR','APR','MAY','JUN','JUL','AUG',
     +             'SEP','OCT','NOV','DEC' /
      data mm /       0,   31,   59,   90,  120,  151,  181,  212,
     +             243,  273,  304,  334 /
      data nl /
     +-883655958.82D0,-867931157.82D0,-852033556.82D0,-820497555.82D0,
     +-788961554.82D0,-757425553.82D0,-725803152.82D0,-694267151.82D0,
     +-662731150.82D0,-631195149.82D0,-820497555.82D0,-583934348.82D0,
     +-552398347.82D0,-520862346.82D0,-457703945.82D0,-378734344.82D0,
     +-315575943.82D0,-284039942.82D0,-236779141.82D0,-205243140.82D0,
     +-173707139.82D0,-126273538.82D0,-94651137.82D0,-79012737.82D0,
```

```
     +-31579136.82D0 /
      data pl / 189345664.18D0,284040065.18D0,394372865.18D0 /
      read(xdate(1:4),104) iyyy
      read(xdate(6:8),203) mon
      read(xdate(10:11),102) ida
      read(xdate(13:14),102) ihr
      read(xdate(16:17),102) imi
      read(xdate(19:24),306) sss

C what is et? (good to < 0.001 sec to 2050, good to < 3 sec 1971)
C will fix later for pre 1971 - problem with Spice documentation on
C leapseconds
      if( iyyy .ge. 2000) then
        sec = 0.D0
C 2000 JAN 1 00:00:00.000 is et  -43135.82
C count full years between 2000 and year before date
        iyrs = iyyy - 2000
        sec = dble(iyrs)*86400.D0*365.D0 ! 365 day years
C first get rid of the -43135.81607 we carry by counting from 2000
C Jan 01 00:00:00 (J2000 is 2000 JAN 01 11:58:55.8200) We would like
C to start at 00:00:00.000
        sec = sec - 43135.81607D0
C take care of previous leap years
        idx = 0
        if(iyyy .gt. 2000) then
          do i = 2000,iyyy-1
          if(((mod(i,4) .eq. 0) .and. (mod(i,100) .ne. 0)) .or.
     +        (mod(i,400)) .eq. 0) idx = idx + 1
          enddo
        endif
C remaining time for year iyyy
        do i = 1,12
          if(mon .eq. month(i)) day = float(mm(i) + ida)
        enddo
        day = day + float(idx)
        sec = sec + dble(day-1)*86400.D0 + dble(ihr)*3600.D0 + dble(imi)
     +       *60.D0 + dble(sss)
C is iyyy a leapyear?
        if(day .gt. 59.) then
          if(((mod(iyyy,4) .eq. 0) .and. (mod(iyyy,100) .ne. 0))
     +       .or.(mod(iyyy,400)) .eq. 0) sec = sec + 86400.D0
        endif
C finally leap seconds !!!!! Last leap 2012 June 30 !!!! add more as needed
        do i = 1,3
          if (sec .ge. pl(i)) sec = sec + 1
        enddo
      endif

C 2000 JAN 1 00:00:00.000 is et  -43135.82
C count full years between 2000 and year of date - include date year
      if(iyyy .le. 1999) then
        idx = 0
        iyrs = iyyy - 2000
        sec = dble(iyrs)*86400.D0*365.D0
C starting from 2000 Jan 1 00:00:00.000
        sec = sec - 43135.81607D0
C take care of leap years
        do i = iyyy,1999
          if(((mod(i,4) .eq. 0) .and. (mod(i,100) .ne. 0)) .or.
     +    (mod(i,400)) .eq. 0) idx = idx - 1
        enddo
C backtrack to date
        do i = 1,12
```

```
          if(mon .eq. month(i)) day = float(mm(i))
        enddo
        day = day + float(ida -1)
        sec = sec + dble(day)*86400.D0 + dble(ihr)*3600.D0 + dble(imi)*
     +        60.D0 + dble(sss)
C is iyyy a leapyear?
        if(day .gt. 59.) then
          if(((mod(iyyy,4) .eq. 0) .and. (mod(iyyy,100) .ne. 0))
     +        .or.(mod(iyyy,400)) .eq. 0) sec = sec + 86400.D0
        endif
C finally leap seconds !!!!! Last leap 2012 June 30 !!!! add more as needed
        do i = 1,25
          if (sec .le. nl(i)) sec = sec - 1.D0
        enddo
      endif
 102  format(i2)
 104  format(i4)
 203  format(a3)
 306  format(f6.3)
      return
      end


*-------------------------------------------------------------*
      SUBROUTINE eqn8015(x,y)
*-------------------------------------------------------------*
***** TableCurve H:\missions\Grid2\sun\eqn8015.f Jul 11, 2013 2:14:58 PM
***** H:\missions\Grid2\sun\Jupiter_rotation.xls
***** X= et
***** Y= drift
***** Eqn# 8015  SineSquared(a,b,c,d)
***** r2=0.998062750570461D0
***** r2adj=0.9980625284088292D0
***** StdErr=0.1736402125675777D0
***** Fval=5990181.452082999D0
***** a= -2.386504709810813D0
***** b= 11.06980649551408D0
***** c= 2.527084753103394D0
***** d= 748890926.9842513D0
***** Constraint: d<>0
*-------------------------------------------------------------*
      DOUBLE PRECISION x,y
      DOUBLE PRECISION n
      n=DSIN(6.283185307179586477 0D0*x/748890926.9842513D0+
     12.527084753103394D0)
      y=(-2.386504709810813D0)+11.06980649551408D0*n*n
      RETURN
      END

*-------------------------------------------------------------*
      SUBROUTINE eqn8001(x,y)
*-------------------------------------------------------------*
***** TableCurve C:\Users\rwe\Desktop\New_folder\missions\Grid2\sun\usrsine2.f Jul 24,
2013 9:19:52 PM
***** sub linear
***** X= et
***** Y= sl
***** Eqn# 8001  usr sine(a,b,c,d)
***** r2=0.8488206233876732D0
***** r2adj=0.8488032863031993D0
***** StdErr=0.05617393071274914D0
***** Fval=65281.63833772391D0
***** a= 0.008316332144591362D0
```

```
***** b= -0.187912858309531D0
***** c= 0.9993681331650221D0
***** d= 0.472891292232232D0
*-------------------------------------------------------------*
      DOUBLE PRECISION x,y
      TWOPI = 2.*PI

      Y = 0.008316332144591362D0 + 0.187912858309531D0 *
     +SIN(0.9993681331650221D0*TWOPI/187058311.284091 * X +
     +0.472891292232232D0)

      RETURN
      END

      subroutine xtime(xdate,sec)
C find et (sec)
      real*8       sec,nl(25),pl(3)
      character*25 xdate
      character*3  mon,month(12)
      dimension    mm(12)
      data month / 'JAN','FEB','MAR','APR','MAY','JUN','JUL','AUG',
     +             'SEP','OCT','NOV','DEC' /
      data mm /       0,   31,   59,   90,  120,  151,  181,  212,
     +             243,  273,  304,  334 /
      data nl /
     +-883655958.82D0,-867931157.82D0,-852033556.82D0,-820497555.82D0,
     +-788961554.82D0,-757425553.82D0,-725803152.82D0,-694267151.82D0,
     +-662731150.82D0,-631195149.82D0,-820497555.82D0,-583934348.82D0,
     +-552398347.82D0,-520862346.82D0,-457703945.82D0,-378734344.82D0,
     +-315575943.82D0,-284039942.82D0,-236779141.82D0,-205243140.82D0,
     +-173707139.82D0,-126273538.82D0,-94651137.82D0,-79012737.82D0,
     +-31579136.82D0 /
      data pl / 189345664.18D0,284040065.18D0,394372865.18D0 /
      read(xdate(1:4),104) iyyy
      read(xdate(6:8),203) mon
      read(xdate(10:11),102) ida
      read(xdate(13:14),102) ihr
      read(xdate(16:17),102) imi
      read(xdate(19:24),306) sss

C what is et? (good to < 0.001 sec to 2050, good to < 3 sec 1971)
C will fix later for pre 1971 - problem with Spice documentation on
C leapseconds
      if( iyyy .ge. 2000) then
        sec = 0.D0
C 2000 JAN 1 00:00:00.000 is et  -43135.82
C count full years between 2000 and year before date
        iyrs = iyyy - 2000
        sec = dble(iyrs)*86400.D0*365.D0 ! 365 day years
C first get rid of the -43135.81607 we carry by counting from 2000
C Jan 01 00:00:00 (J2000 is 2000 JAN 01 11:58:55.8200) We would like
C to start at 00:00:00.000
        sec = sec - 43135.81607D0
C take care of previous leap years
        idx = 0
        if(iyyy .gt. 2000) then
          do i = 2000,iyyy-1
          if(((mod(i,4) .eq. 0) .and. (mod(i,100) .ne. 0)) .or.
     +        (mod(i,400)) .eq. 0) idx = idx + 1
          enddo
        endif
C remaining time for year iyyy
        do i = 1,12
```

```
          if(mon .eq. month(i)) day = float(mm(i) + ida)
        enddo
        day = day + float(idx)
        sec = sec + dble(day-1)*86400.D0 + dble(ihr)*3600.D0 + dble(imi)
     +        *60.D0 + dble(sss)
C is iyyy a leapyear?
        if(day .gt. 59.) then
          if(((mod(iyyy,4) .eq. 0) .and. (mod(iyyy,100) .ne. 0))
     +        .or.(mod(iyyy,400)) .eq. 0) sec = sec + 86400.D0
        endif
C finally leap seconds !!!!! Last leap 2012 June 30 !!!! add more as needed
        do i = 1,3
          if (sec .ge. pl(i)) sec = sec + 1
        enddo
      endif

C 2000 JAN 1 00:00:00.000 is et  -43135.82
C count full years between 2000 and year of date - include date year
      if(iyyy .le. 1999) then
        idx = 0
        iyrs = iyyy - 2000
        sec = dble(iyrs)*86400.D0*365.D0
C starting from 2000 Jan 1 00:00:00.000
        sec = sec - 43135.81607D0
C take care of leap years
        do i = iyyy,1999
          if(((mod(i,4) .eq. 0) .and. (mod(i,100) .ne. 0)) .or.
     +    (mod(i,400)) .eq. 0) idx = idx - 1
        enddo
C backtrack to date
        do i = 1,12
          if(mon .eq. month(i)) day = float(mm(i))
        enddo
        day = day + float(ida -1)
        sec = sec + dble(day)*86400.D0 + dble(ihr)*3600.D0 + dble(imi)*
     +        60.D0 + dble(sss)
C is iyyy a leapyear?
        if(day .gt. 59.) then
          if(((mod(iyyy,4) .eq. 0) .and. (mod(iyyy,100) .ne. 0))
     +        .or.(mod(iyyy,400)) .eq. 0) sec = sec + 86400.D0
        endif
C finally leap seconds !!!!! Last leap 2012 June 30 !!!! add more as needed
        do i = 1,25
          if (sec .le. nl(i)) sec = sec - 1.D0
        enddo
      endif
 102  format(i2)
 104  format(i4)
 203  format(a3)
 306  format(f6.3)
      return
      end
```

C. Sample Record in jfluence

For those who want to implement the Grid2 into their own programs, the above program listing has the algorithm to find records that have positions that surround the spacecraft position for a specific local time (Locations < 15 R_j ignore local time) highlighted in yellow.

For graphics, the grid is probably too course in range to use as is. You can define your own grid and use it as "spacecraft positions" below. Interpolate between grid points using the same algorithm. In this case you will have to switch shells as you go around Jupiter in local time.

Below is an explanation of the highlighted lines, not in order of the program listing.

Find an index (iri) associated with range (database range < range).

```
      irj = 0
      do i =  1,100
        if(rj .gt. rdb(i)) irj = i
      enddo
```

The program listing has a data statement that list rdb near the top of the program.

Or skip point

```
C if irj out of range read another record.
      if(irj .eq. 0. .or. irj .eq. 100) go to 1
```

Find index associated with latitude (ial) and West longitude (iwl)

```
      if(alat .eq. 90.) alat = alat - 0.0001 ! for proper integer below
      ial = int(alat + 90.)/2
      iwl = int(wlong)/3
```

Calculate initial records for 8 bounding positions, the "cuboid"

```
C irec for bounding 'cuboid'
      irec(1) = iwl   + ial*120     + (irj-1)*10920 ! minr minal minwl
      irec(2) = iwl   + (ial+1)*120 + (irj-1)*10920 ! minr maxal minwl
      irec(3) = iwl+1 + ial*120     + (irj-1)*10920 ! minr minal maxwl
      irec(4) = iwl+1 + (ial+1)*120 + (irj-1)*10920 ! minr maxal maxwl
      irec(5) = iwl   + ial*120     +  irj*10920    ! maxr minal minwl
      irec(6) = iwl   + (ial+1)*120 +  irj*10920    ! maxr minal maxwl
      irec(7) = iwl+1 + ial*120     +  irj*10920    ! maxr maxal minwl
      irec(8) = iwl+1 + (ial+1)*120 +  irj*10920    ! maxr maxal maxwl
```

include 0 in 330-360 shell

```
      if (iwl .eq. 0) then
        irec(1) = 120 + ial*120     + (irj-1)*10920 ! 360 --> 0
        irec(2) = 120 + (ial+1)*120 + (irj-1)*10920 ! 360 --> 0
        irec(5) = 120 + ial*120     +  irj   *10920 ! 360 --> 0
        irec(6) = 120 + (ial+1)*120 +  irj   *10920 ! 360 --> 0
      endif
```

Use et (calculated) or etx (supplied) to find out which shell

First, is et supplied?

```
C find et (don't skip dates where Range is out of bounds to avoid LARGE
C delta T, use etx if available
      if(etx .eq. 0.D0) call xtime(date,et)
      if(etx .ne. 0.D0) et = etx
      if(n .eq. 1)etold = et
```

Find spacecraft local time (wlt0)

```
      if(rj .gt. rdb(80)) then
        call jup(et,wlong,wlt0)
```

Find index of record

The program listing has a data statement that lists wlx near the top of the program:

```
C -15 to 15, 15-45, etc.
        do i = 2,14
          if( wlt0 .ge. wlx(i-1) .and. wlt0 .lt. wlx(i))
     +        irecx= ireca(i) - 79 * 10920
        enddo
        do i = 1,8
          irec(i) = irec(i) + irecx
        enddo
      endif
```

```
                    The core grid is in records 1-873600
                 Shell grids for Spacecraft Local times are
                    Local time range            records
                    015-045              873601-1102920
                    045-075             1102921-1332240
                    075-105             1332241-1561560
                    105-135             1561561-1790880
                    135-165             1790881-2020200
                    165-195             2020201-2249520
                    195-225             2249521-2478840
                    225-255             2478841-2708160
                    255-285             2708161-2937480
                    285-315             2937481-3166800
                    315-345             3166801-3396120
                    345-015             3396121-3625440
```

What is in a record?

A record contains record #, range (R_j), latitude (degrees), West-longitude(degrees), L, Bspacecraft (Gauss), Bequator (Gauss), trash and 17 electron fluxes (fe#), 17 proton fluxes (fp#) for energies. The 17 energies are in bins at mega electron volt levels of 0.10, 0.20, 0.30, 0.50, 1.00, 2.00, 3.00, 5.00, 10.00, 20.00, 30.00, 50.00, 100.00, 200.00, 300.00, 500.00, and 1000.00).

The quantity labeled "trash" was supposed to be zmap (from Khurana, Ref. 3), but the wrong column was selected—zmap is not used by Grid2, just used to produce the local time shells. It is worthless for including in the grid, and trash can be later eliminated later – saving 36 MB in the file. Range, Latitude, Wlong, L, Bsc, Beq are also not needed for fluences calculations, the first three can be found by the program and the last were left in for other work. Eliminating these would also save an additional 250 MB.

```
Record # Range(RJ) Latitude(d) Wlong(d) L        Bsc       Beq
Trash     fe1       fe2       fe3       fe4       fe5       fe6
fe7       fe8       fe9       fe10      fe11      fe12      fe13
fe14      fe15      fe16      fe17      fp1       fp2       fp3
fp4       fp5       fp6       fp7       fp8       fp9       fp10
fp11      fp12      fp13      fp14      fp15      fp16      fp17

Read a record with (RECL=418)
       read(12,REC=irec(i))line

Then read off quantities within a record with
       read(line,11) ict(i),ry(i),alay(i),wly(i),by(i),beqy(i),aly(i)
     +               ,blaty(i),(ay(i,j),j=1,34)
11   format(i8,41(1x,e9.3))


    1058 0.103E+01 -.740E+02 0.294E+03 0.828E+01 0.235E-02 0.103E+02
0.900E+02 0.179E+09 0.111E+09 0.812E+08 0.522E+08 0.257E+08 0.107E+08
0.588E+07 0.248E+07 0.640E+06 0.139E+06 0.535E+05 0.154E+05 0.271E+04
0.462E+03 0.162E+03 0.424E+02 0.604E+01 0.144E+09 0.114E+08 0.260E+07
0.405E+06 0.337E+05 0.318E+04 0.921E+03 0.244E+03 0.594E+02 0.177E+02
0.829E+01 0.280E+01 0.553E+00 0.105E+00 0.397E-01 0.116E-01 0.219E-02
```

D. Acronyms and Abbreviations

ΔT	difference in time
Alat	spacecraft latitude (deg)
ASCII	American Standard Code for Information Interchange
B	Magnetic field parameter in McIlwain B-L magnetic coordinate system
Date	date in NAIF/SPICE format YYYY MMM DD hh:mm:ss Example: 2024 JAN 02 00:00:00.0000
et	ephemeris time based on J2000 (in seconds)
etx	ephemeris time J2000 (in seconds)
GIRE	Galileo Interim Radiation Electron model [original version]
GIRE2	Galileo Interim Radiation Electron model 2 [latest version]
Grid	original grid array model (there was no Grid1)
Grid2	current upgraded grid array model
ial	index associated with latitude
J2000	international standard for starting time referencing noon, Jan. 1, 2000
jfluence	Grid2 output file for Jupiter fluence
JPL	Jet Propulsion Laboratory
L	dipole shell parameter for the McIlwain B-L magnetic coordinate system
MeV	mega electron volt
NAIF	(NASA) Navigation and Ancillary Information Facility (established at the Jet Propulsion Laboratory to lead the design and implementation of the "SPICE" ancillary information system)
NASA	National Aeronautics and Space Administration
NOVICE	(radiation transport shielding code from Experimental and Mathematical Physics Consultants)
R	spacecraft range from Jupiter
R_j	spacecraft range in Jovian diameters
SPICE	Spacecraft Planets Instrument C-Matrix Events
Wlong	spacecraft West longitude (deg)