EMMREM OPERATIONAL MANUAL

version 0.9.1

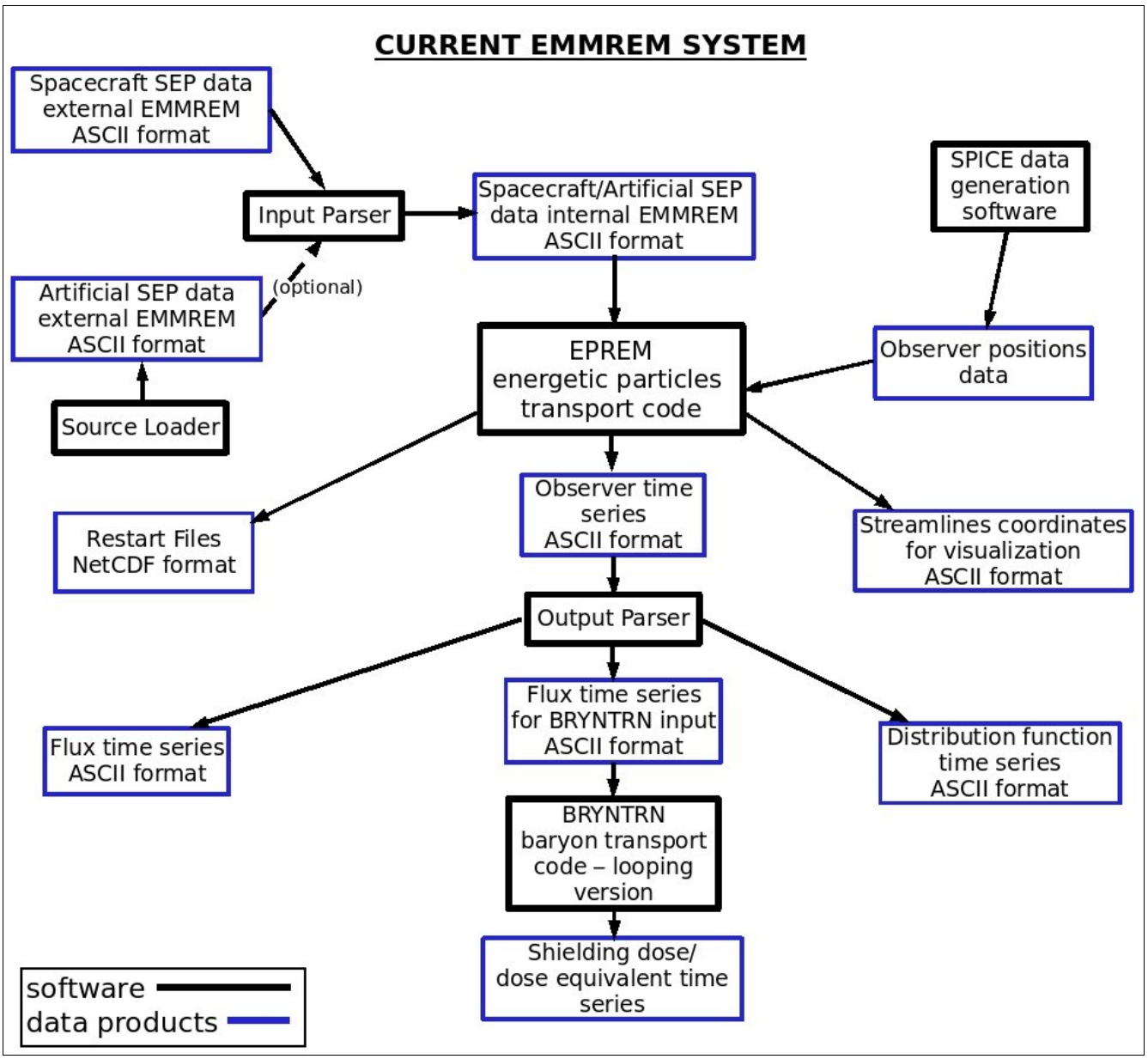
02/12/2009

prepared by Kamen Kozarev (kamen@bu.edu)

Contents:

1.OPERATIONAL OVERVIEW OF THE CURRENT EMMREM MODULE

The EMMREM system is in constant development by scientists in the various participating institutions. In this manual, we are focusing on the radiation propagation aspect of the EMMREM project - the part of it that we have developed, and that is currently in heaviest use. It was therefore required that a manual for the use of this EMMREM subsystem (which we will refer to as just EMMREM henceforth) be prepared. This manual was made as complete as possible at its writing, and efforts are made to update it regularly.



The above is a schematic of the current EMMREM operational framework, which will be discussed at length below. The blue-bordered rectangles represent input or output data products for the various subsystems. The rectangles in black borders represent the different software products, which are run in series. This whole system is controlled by a number of bash shell and Perl scripts.

1.1 INPUT TO THE SYSTEM

On the top of the diagram, the EPREM transport code takes two types of input data – EP flux time series, and positions of the various bodies of interest in the simulation in Heliocentric Inertial coordinates. The EP data can be from spacecraft in-situ observations, or can come be prepared with the sourceLoader software. This software can create flux time series for a point or wave source, moving or stationary, anywhere in the heliosphere. This software is still in development stage, and so we will defer further discussion of it for now. The reader should read its description in Appendix A.

1.1.1 THE INPUT PARSER

The external EMMREM format data (SEP flux time series observations from GOES, ACE, etc.) is converted to internal EMMREM format data (distribution function time series of SEPs) by the input Parser. These input data are then read into the EPREM executable.

1.1.2 THE SPICE POSITIONS LOADER

The other input is heliocentric positions of the relevant bodies for the time range of a particular run. Those are generated by a command utilizing the CSPICE library of the NASA SPICE Toolkit, and the data for those are generated by the NASA NAIF group in the form of kernel files.

1.2 EPREM

The EPREM submodule is a 3-D kinetic numerical simulation of Solar Energetic Particles transport throughout the inner Heliosphere. It is a parallelized code, written in C/C++, and it solves kinetic equations for the SEP time-dependent propagation. The user of the code can select several "observers" for each run - points in the computational grid, where distribution function values get dumped as output of the model continuously.

1.3 EPREM OUTPUT

The EPREM transport code currently generates two types of primary output: a restart file in the NetCDF format, which contains the entire computational grid together with various parameters important for the run, and is used to restart the runs; and observer files, which are time series of distribution function spectra for various positions in the code, in ASCII internal EMMREM format.

1.3.1 Restart File

The restart file is usually called **restart.nc**, and is written in the user's local folder. It can be can be written a different number of times throughout a run. This is set in the configure file for EPREM, by changing the following parameters:
dumpOnAbort – 1 means save a restart file when the run is interrupted and before quitting, 0 means don't;
dumpFreq – takes the number of timesteps after which the restart file will be overwritten with the current grid information;
saveRestartFile – 1 means leave the restart file, 0 – don't;
The production of a restart file is important when a run gets interrupted, especially on supercomputers with batch processing and limited wallclock time for individual runs. Currently, only the last timestep of the code is written into the restart file, but in the future we plan to include time dependence.

1.3.2 Observer Output Files

The observer output files are also written in the user's local folder. They have the names of the "observer", followed by "Observer.txt" (for example, **earthObserver.txt**.) As mentioned previously, they contain time series with values for the SEP distribution functions at the different energies of the particular simulation. The output for each observer can vary, depending on how many species, energies, or pitch angles a particular simulation uses.

1.4 THE OUTPUT PARSER

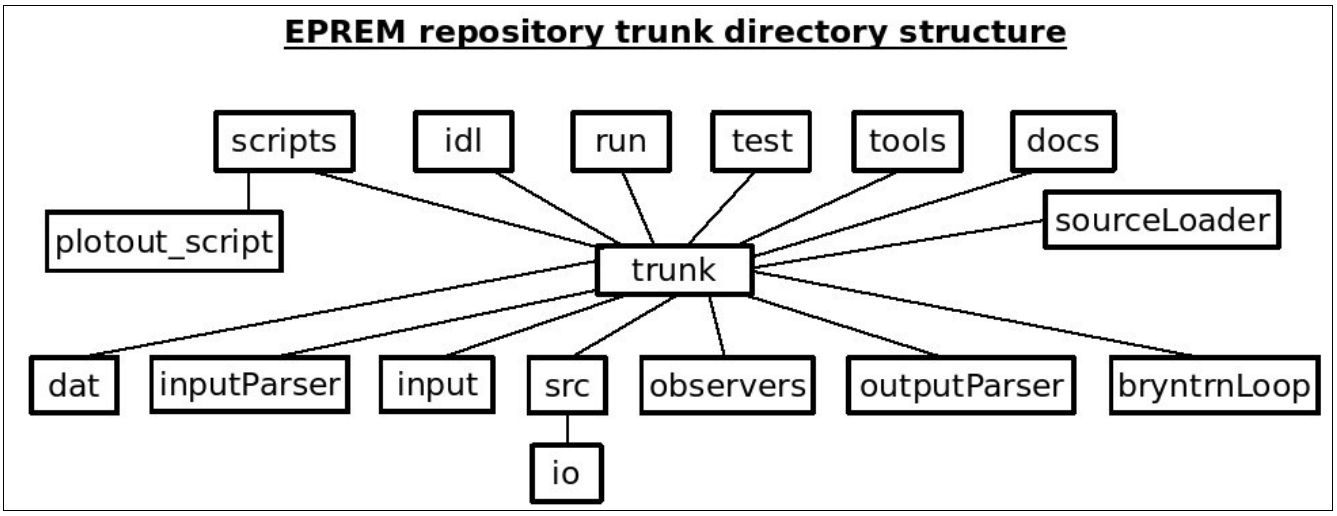
The observer output from each run is used by different applications down the EMMREM pipeline after converting to appropriate formats. This conversion is done in the output parser – a standalone piece of software written in C, which is situated in the outputParser/ folder. The distribution function values are converted to fluxes and written in a much simpler ASCII format as time series for several energies to be plotted in IDL or other software. The fluxes are also output in a format suitable for the looping version of BRYNTRN to read. The last output is a simple format of the distribution function time series, again for visualization.

1.5 BRYNTRN

EMMREM contains a version of the Baryon Transport (BRYNTRN) code, which runs over many time steps with the help of a Perl command (**loop.pl**). The BRYNTRN code, executable, and the command are located in the bryntrnLoop/ folder. The output of the code itself is a quite complicated ASCII file, but only the important bits for EMMREM (i.e., dose and dose equivalent time series for different shielding depths and shielding material combinations) are extracted from it by the Perl

command, and written into an output text file in the same folder.

2. OVERVIEW OF THE CURRENT EMMREM FOLDER STRUCTURE



The following is a description of the contents and function of the various folders in the EMMREM distribution. The folders have been ordered by alphabetical order in this description.

bryntrnLoop – This folder contains the looping version of the BRYNTRN particle transport model for EMMREM. It also contains output files, a PERL looping command, and IDL plotting procedures for the output.

dat – Contains data files with energetic particle flux time series observations for various locations and spacecraft, such as Earth (GOES), Ulysses, etc. The data is in the EMMREM self-descriptive external format.

docs – This folder contains relevant to the distribution documents, papers, articles, etc.

idl – Here are some IDL procedures for visualization and other purposes.

input – Contains the data from the dat/ folder, which has been converted to the EMMREM self-descriptive internal format.

inputParser – A folder containing software (inputParser), which converts the external format data files to the internal format for the EPREM code.

observers – This folder contains the EPREM output files with the time series for the various observers.

outputParser – Contains software (outputParser), which converts the EPREM output at various observers (i.e. energy spectrum timeseries) back to flux time series.

run – Currently unused.

commands – A repository of useful commands for auxiliary purposes.

commands->output_command – This folder contains a command and IDL procedures for visualizing the output of EPREM runs.

sourceLoader – This folder contains the sourceLoader software, which creates artificial energetic particle time series. This can be used to add EP sources to the EPREM model, or just for comparison.

src – The main folder of the distribution – it contains almost all the source code, Makefile, headers, etc.

src->io – This folder contains source code that runs the restart file capability of EPREM.

test – A folder for tests.

tools – Another repository for commands and tools. Currently unused.

3. INSTALLING AND RUNNING EMMREM

The first thing you need to do is to download the current EMMREM distribution. You will need access to the EMMREM subversion repository, where the code is kept, as well as to the ComEMMREM repository, where additional pieces might live. Contact nathanas@bu.edu or kamen@bu.edu for access.

Assuming that you do have access to the EMMREM and ComEMMREM repositories, you should be able to open a telnet/csh/bash terminal, and download a copy of the current working version of the EMMREM code.

```
% svn co https://field.bu.edu/EMMREM/svn/emmrem/trunk trunk
```

NB! Subversion is a free project versioning software, which allows the user/s to keep track of ALL changes to their documents/code/project. More on it find here: http://subversion.tigris.org/

A little subversion lingo: The subversion logic follows the tree analogy. In it, the “trunk” is a pristine version of the software code, where updates are put after we're sure the code works well with them. A “branch” is just another copy of the code, but in development phase. Thus, the software can have many branches, but developers want to keep only one trunk, which is the stable version.

3.1 REQUIRED LIBRARIES

Assuming you have already installed a Linux distribution (This manual leans heavily towards examples from the UBUNTU distribution, since it is the BU group OS of choice), the following packages are required:

	Package	(Ubuntu Synaptic name)
a)	subversion	(subversion, svn-load),
b)	autoconf	(autoconf),
c)	automake	(automake),
d)	libmpich	(libmpich-shmem1.0gf, libmpich-shmem1.0-dev, libmpich1.0gf)
e)	mpich-shmem	(mpich-shmem-bin)
f)	netcdf	(netcdf-bin, netcdfg-dev, netcdf-dbg)
g)	zlib1g	(zlib1g)
h)	bash	(bash)
g)	g77	(g77)
i)	gcc	(gcc)

You must also download the software **cspice** from the following site and follow the required steps to install it on your machine. There are several versions of the SPICE toolkit, depending on the particular system it's being installed on (which the user/admin has to determine):

http://naif.jpl.nasa.gov/naif/toolkit_C.html

Later in the manual, there will be more about installing SPICE. It is a handy toolkit allowing creation of time series of accurate positions/pointing/attitude of spacecraft and Solar System bodies in many different coordinate systems. Cspice is its distribution of code libraries written in C. Below, we might use 'SPICE' and 'cspice' interchangeably.

NB! In order to use the visualization suite of scripts we've developed for EMMREM output, the user must have the Interactive Data Language (IDL) installed.

3.2 PREPARING EMMREM TO RUN ON A PARTICULAR SYSTEM

Assuming this worked and you checked out the latest version of the trunk fine, you can go ahead and prepare the code for running. There are several steps you need to take to setup the environment for running EMMREM. **The following installation procedure assumes that you are running Linux, Unix, or Mac OSX, and have the BASH shell installed and running.** We will assume that the user has copied over the EMMREM code on their machine, in folder **/usr/share/emmrem/**, and that the SPICE toolkit has been installed in **/usr/share/cspice/**. Of course you don't HAVE to install them there, this is just an example. Let's begin!

1. EMMREM needs to 'know' where it's installed, so you must add to your **.bashrc** file - in your home folder, something like /home/Joe/ (if it doesn't exist, create it) - the following two lines:
export EMMREM_TRUNK='/usr/share/emmrem'
alias mrm_dev='. $EMMREM_TRUNK/scripts/mrm_dev.sh'

After this is done, you will need to restart bash to continue the installation. The second line is an alias to a command which makes all the EMMREM commands runnable from anywhere on your computer. This means that any time you start a terminal window in which you want to do EMMREM-related work, you should execute 'mrm_dev' from the command prompt in order to enter the EMMREM development environment (that is, to be able to execute EMMREM commands.)

2. EMMREM also needs to know where SPICE is installed, so **after** you've restarted bash and run mrm_dev, you need to run another command, which will fix directory dependencies for SPICE, with an argument the absolute path of your cspice installation, like so:
$ fixSpiceDirs.sh /usr/share/cspice

3. Next, go to the EMMREM installation folder. To prepare EPREM for running, type
$ ./prepare
If it doesn't give you any errors, next type
$ ./configure
These two commands prepare the code to run smoothly wherever you have installed it. This is only done once, the first time you install the code. You might get some errors on running './configure' having to do with NetCDF or any other required packages. For NetCDF you might have to supply its folder:
$ ./configure –with-netcdf=[netcdf_installation_folder]
If you are certain that you won't get any errors on running these programs, you can execute them from anywhere on your computer by running the command
$ prepareEPREM.sh

4. Next, go to the folder '***src/***' and open up a text file called '**observerOutput.c**'. The first defined function in it (named appropriately *defineObservers()*) serves to set up the locations of the observers inside the code. You will see that there are four observers with different x,y,z positions and different names – Earth, Moon, Mars, Ulysses. You can play around and change these, but after every change you need to recompile the code (more on that in a minute). These are the default observers.

So, EMMREM is now almost installed. The final step is to compile (also known as 'to make') all the executables. Below is a step-by-step guide on compiling and running each important executable of EMMREM. If you only need to compile the
EMMREM executables, you can use the command
$ batchMakeEMMREM.sh

NB! *Several of the programs below require the presence of text files containing information that is used by the programs, in the place where those programs are executed. From our practice, we usually call these configure files for the programs inputParser, outputParser, sourceLoader, spiceLoad, and EPREM* ***iConf***, ***oConf***, ***lConf***, ***sConf***, *and* ***eConf***, *respectively. We will use this nomenclature below.*

3.2.1 inputParser

Below are the atomic steps for running the "inputParser" program. This program takes preformatted SEP intensities data at 1 AU(Earth), and formats it, so that it can be digested by EPREM. Its output is an input to EPREM. For more on the inputParser, run
$ makeInputParser.sh --help

The inputParser requires a data file in the EMMREM self-descriptive ascii format. If you want to see an example of the format, run
$ makeInputParser.sh -s

This will copy to your local folder the file 'SEPdataFormat.txt'.

Step 1. Compile inputParser by typing
$ makeInputParser.sh

To check if it compiled properly by run a test run
$ runInputParser.sh -t

Step 2. The inputParser needs a config file to run it, so to get a local copy of the configure file, type
$ makeInputParser.sh iConf

Step 3. Examine (and edit, if necessary) the contents of the local configure file copy.

Step 4. If you have both the data file for input, and the configure file (with necessary modifications) in your local folder, run
$ runInputParser.sh iConf

Step 5. **OUTPUT:** The output of running this command is an ascii file with the time series converted from particle intensity to distribution function values, in the internal EMMREM format. It will be saved in the name and folder specified in the "**epremOutFname**" option in the ***iConf*** file.

Step 6. VISUALIZATION: The output from running this executable can be viewed by running the following commands:
plotFluxOne.sh - plots the flux from a single generic observer/spacecraft;
plotEarthDataFlux.sh - plots the flux from GOES spacecraft;
plotUlyssesDataFlux.sh - plots the flux from Ulysses spacecraft.
The user can look at input requirements, runtime options and output by running '[command] –help'. A complete decommandion of each command is available at the end of this manual (***Appendix B.***)

3.2.2 spiceLoad

Below are the atomic steps for running the "spiceLoad" program. This program creates very accurate positions for the different 'observers' inside the EPREM model. Its output is an input to EPREM. For more on spiceLoad, run
$ makeSpiceLoad.sh --help

Step 1. Compile spiceLoad by typing
$ makeSpiceLoad.sh

To check if it compiled properly by running a test, run
$ runSpiceLoad.sh -t

Step 2. The spiceLoad needs a config file to run it, so to get a local copy of the config file, type
$ makeSpiceLoad.sh sConf

Step 3. Examine (and edit, if necessary) the contents of the local configure file copy.

Step 4. If you have both the data file for input, and the configure file (with necessary modifications) in your local folder, run
$ runSpiceLoad.sh sConf

Step 5. OUTPUT: The output of running this command is an ascii file with the time series of the desired objects' positions and velocities, in the common external EMMREM ascii format. It will be saved in the name and folder specified in the "**outFilename**" option in the **sConf** file.

Step 6. VISUALIZATION: No visualization has been planned so far for the product of this executable. In the future, we might develop orbit/position plots for the different observers.

3.2.3 EPREM

Below are atomic steps for preparing and running the EPREM code. For more on eprem, run
$ runEPREM.sh --help

Step 1. This code is very big and resource-consuming, compared to the other programs in the system, so caution must be exercised in running it. The first step is making sure the required libraries and packages exist on your machine and that EPREM will compile appropriately:
$ prepareEPREM.sh

This command in general needs to be run only once, at the initial stage of setting up the EMMREM installation.

Step 2. The second step is to make sure that the code compiles properly and clean old object files and executables run:
$ makeEPREM.sh -c

to make the eprem executable, run:
$ makeEPREM.sh

Step 3. EPREM needs a config file to run it, so to get a local copy of the config file, type
$ makeEPREM.sh eConf

Step 4. Edit the ***eConf*** file for your specific run. EPREM needs the output of inputParser and spiceLoad to run. The output files from those two programs are input to EPREM, and are set in the eprem config file, in variables "**sepFilename**" and "**obsPosFilename**". Make sure you enter the correct filenames there before you run EPREM.

Step 5. If you want to start a new run, then you will first need to

remove any NetCDF restart file by running
$ runEPREM.sh -s

To run the EPREM executable with default #processors (8) and the configure file (with necessary modifications, run
$ runEPREM.sh eConf

For more running options, run
$ runEPREM.sh –help

To interrupt a run, type **Ctrl+C**. Sometimes, if you've interrupted a run the previous time, you'll get an error message. This might be because every time you run EPREM, a file is created every so often, which backs up the grid. If you interrupt in the middle of a run, it will complain the next time, since that file wasn't properly closed. To make sure that's not the issue, you need to remove this file by typing
$ rm restart.nc

and then run EPREM again.

Step 6. OUTPUT: When the code is done running, it copies to your local directory the observer files, which must end in **Observer.txt.** These names are the ones you have set in the **observerOutput.c** file. Examples are earthObserver.txt, moonObserver.txt, etc.

These contain time series of distribution function spectra over different species and pitch angles. The files correspond to specific positions inside the model grid, which in turn corresponds to the positions of bodies in the Solar system. One can add other observers by editing the **defineObservers()** function inside **observerOutput.c** in the src/ folder of the EMMREM distribution.
Additional output not connected with the observers is:

epremRunParams.dat
epremStreamNodesLocs.dat

The first file returns a log of the runtime parameters for the run. The second file returns the positions of all the nodes of the computational grid.

To continue the pipeline, you'll need to run the ***Observer.txt** files through the **outputParser**.

Step 7. VISUALIZATION: The only visualization that can be done directly after running EPREM in the current development stage of EMMREM is making a figure or a movie of the computational grid using **epremStreamNodesLocs.dat**. For this, run the commands
movieFieldLines.sh – make a movie of a rotating grid.
printFieldLineMap.sh – make a picture of the grid.
The user can look at input requirements, runtime options and output by

running '[command] –help'. A complete description of each command is available at the end of this manual (***Appendix B.***)

3.2.4 outputParser

Below are the atomic steps for running the "outputParser" program. This program takes the output of EPREM, and converts it to simple flux time series in different formats. One of its outputs is the input to BRYNTRN. For more on the outputParser, see the EMMREM manual, or run

$ makeOutputParser.sh --help

The outputParser requires a data file in the EMMREM self-decommandive ascii format. If you want to see an example of the format, run
$ makeOutputParser.sh -s

This will copy to your local folder the file '**earthObserver.good**'.

Step 1. Compile outputParser by running
$ makeOutputParser.sh

To check if it compiled properly by running a test, run
$ runOutputParser.sh -t

Step 2. The inputParser needs a config file to run it, so to get a local copy of the config file, type
$ makeOutputParser.sh oConf

Step 3. Examine (and edit, if necessary) the contents of the local configure file copy.

Step 4. If you have both the data file for input, and the configure file (with necessary modifications) in your local folder, run
$ runOutputParser.sh oConf

Step 5. OUTPUT: There are several outputs from running "outputParser". The first is a special output of energy spectra time series, generated for the looping version of BRYNTRN. It will be saved in the file and folder specified in the "**brynOutFname**" option in the configure file. The second output is a flux time series for the observer at different energies, in a format for plotting. It will be saved in the file and folder specified in the "**fluxOutFname**" option in the configure file. The third output is a distribution function time series for the observer, in a format for plotting. It will be saved in the file and folder specified in the "**distOutFname**" option in the configure file.

Step 6. VISUALIZATION: To make plots from the flux output files, the

user can run several commands (listed below). Most of these require specific input, since they've been written with a specific observer in mind. The first three are general, and take only command line input. The user can look at input requirements, runtime options and output by running '[command] –help'. A complete description of each command is available at the end of this manual (***Appendix B.***)
makeFluxPlots.sh – creates flux plots for all flux files that are present in the folder. It uses plotFluxOne.sh and plotFluxTwo.sh, and can be executed instead of all the other commands below. Recommended for batch use.
plotFluxOne.sh - plots the flux from a single generic observer/spacecraft
plotFluxTwo.sh - plots the flux from two generic observers/spacecraft
plotEarthDataFlux.sh - plots the flux from GOES spacecraft
plotEarthFlux.sh - plots the flux from the Earth observer and GOES spacecraft
plotEarthObsFlux.sh - plots the flux from the Earth observer
plotMarsFlux.sh - plots the flux from the Mars observer
plotMoonFlux.sh - plots the flux from the Moon observer
plotUlyssesDataFlux.sh - plots the flux from Ulysses spacecraft
plotUlyssesFlux.sh - plots the flux from the Ulysses observer and spacecraft
plotUlyssesObsFlux.sh - plots the flux from the Ulysses observer

3.2.5 BRYNTRN (looping version)

Below are the atomic steps for running the looping version of the "BRYNTRN" program. BRYNTRN takes as input one of the outputs from running the outputParser. For more information on BRYNTRN, run
$ makeBRYNTRN --help

Step 1. To compile BRYNTRN, run
$ makeBRYNTRN.sh

Step 2. Running BRYNTRN requires an input file, which is the output of the EPREM model – spectra of particle intensities over time in a specific format. An example can be obtained by running
$ runBRYNTRN -s

This will copy the file "**LoopInputFormat.dat**" to your local folder.

Step 3. To run the looping version of BRYNTRN with another input file, **earthInput.dat**, have the input file in your folder and run
$ runBRYNTRN earthInput.dat

Step 4. OUTPUT: When the code is done running, it copies to your local

directory the following files:

output.final - A time series of Dose/Dose Equivalent rates.
earthOutput.plot - Same as **output.final**, but in a different format. This file is used for plotting. In general, output will be in the form [observer]Output.plot, where [observer] is the observer name, such as earth.

Step 5. VISUALIZATION: To make plots from the output files(***Output.plot**), the user can run several commands (listed below). Most of these require specific input, since they've been written with a specific observer in mind. The first three are general, and take only command line input. The user can look at input requirements, runtime options and output by running '[command] –help'. A complete description of each command is available at the end of this manual (***Appendix B.***)
makeDosePlots.sh – creates all relevant dose plots, such as Skin and BFO dose, dose equivalent, accumulated dose and dose equivalent, relative GCR dose and dose equivalent, for all BRYNTRN output files that are found. The files need to have the **Output.plot** suffix in order to be ingested by this script. It uses the following two scripts to automate the plot-making process, and can substitute all other commands listed below.
plotDose.sh - plots the dose/dose equiv. rates at an observer/spacecraft
plotAccuDose.sh - plots rates and accumulated dose/dose equiv. at an observer/spacecraft
plotEarthDataAccuDose.sh - plots rates and accumulated dose/dose equiv. at GOES spacecraft
plotEarthDataDose.sh - plots the dose/dose equiv. rates at GOES spacecraft
plotEarthObsAccuDose.sh - plots rates and accumulated dose/dose equiv. at Earth Observer
plotEarthObsDose.sh - plots the dose/dose equiv. rates at Earth Observer
plotMarsObsAccuDose.sh - plots rates and accumulated dose/dose equiv. at Mars Observer
plotMarsObsDose.sh - plots the dose/dose equiv. rates at Mars Observer
plotMoonObsAccuDose.sh - plots rates and accumulated dose/dose equiv. at Moon Observer
plotMoonObsDose.sh - plots the dose/dose equiv. rates at Moon Observer
plotUlyssesDataAccuDose.sh - plots rates and accumulated dose/dose equiv. at Ulysses spacecraft
plotUlyssesDataDose.sh - plots the dose/dose equiv. rates at Ulysses spacecraft
plotUlyssesObsAccuDose.sh - plots rates and accumulated dose/dose equiv. at Ulysses Observer
plotUlyssesObsDose.sh - plots the dose/dose equiv. rates at Ulysses

Observer

3.2.6 MarsBryn (looping version)

Below are the atomic steps for running the looping version of the "BRYNTRN4_MARSREM" program, which produces dose and dose equivalent/rates at different positions in the Mars atmosphere, for different shielding depths.

Step 1. To compile MarsBryn, run
$ makeMarsBryn.sh

Step 2. Running MarsBryn requires an input file, which is the output of the EPREM model - spectra of particle intensities over time in a specific format for the Mars location - marsObserver.txt. An example can be obtained by running
$ runMarsBryn.sh -s
This will copy the file "**LoopInputFormat.dat**" to your local dir. This file is the same as the input for the regular BRYNTRN model.

Step 3. To run the looping version of MarsBRYN, have the input file in your folder and run
$ runMarsBryn.sh [input_filename]

Step 4. OUTPUT: When the code is done running, it copies to your local directory the following files:

output.final - Dose/Dose Eq. rate time series for different aterial/atmosphere depths
output.mars - Dose/Dose Equiv. at different mean heights of Martian atmosphere per timestep
outputDose.plot - Dose rate time series for different material/atmosphere depths for plotting

Step 5. VISUALIZATION:

3.2.7 MarsBrynAtm (looping version)

Below are the atomic steps for running the looping version of the "BRYNTRN4_MARSREM" program, which produces dose and dose equivalent/rates at different positions in the Mars atmosphere. For more info, run
$ makeMarsBrynAtm.sh --help

Step 1. To compile MarsBryn, run

$ makeMarsBrynAtm.sh

Step 2. Running MarsBryn requires an input file, which is the output of the EPREM model - spectra of particle intensities over time in a specific format for the Mars location - marsObserver.txt. An example can be obtained by running
$ runMarsBrynAtm.sh -s

This will copy the file "**LoopInputFormat.dat**" to your local dir. This file is the same as the input for the regular BRYNTRN model.

Step 3. To run the looping version of MarsBRYN, have the input file in your folder and run
$ runMarsBrynAtm.sh [input_filename]

Step 4. OUTPUT: When the code is done running, it copies to your local directory the following files:

output.final - Dose/Dose Eq. rate time series for different aterial/atmosphere depths
output.mars - Dose/Dose Equiv. at different mean heights of Martian atmosphere per timestep
marsOutput.plot - Dose rate time series for different atmosphere depths for plotting
marsOutputTopBottom.plot - Dose rate time series for different atmosphere elevations for plotting

Step 5. VISUALIZATION: To make plots from the dose output files, the user can run several commands (listed below). These require specific input (**marsOutput.plot**), since they've been written with the Mars observer in mind. The first one is general, and take only command line input. The last two commands take the file **marsOutputTopBottom.plot**, and produce dose/dose equivalent time series at different elevations in the Martian atmosphere. The user can look at input requirements, runtime options and output by running '[command] –help'. A complete decommandion of each command is available at the end of this manual (***Appendix B.***)

plotDose.sh - plots the dose/dose equiv. rates at an observer/spacecraft
plotMarsObsAccuDose.sh - plots rates and accumulated dose/dose equiv. at Mars Observer
plotMarsObsDose.sh - plots the dose/dose equiv. rates at Mars Observer
plotMarsTopBottomDose.sh - plots the dose/dose equiv. rates for different elevations at the Mars Atmosphere
plotMarsTopBottomAccuDose.sh - plots rates and accumulated dose/dose equiv. for different elevations at the Mars Atmosphere.

4. AN EMMREM PIPELINE EXAMPLE

This section will provide a brief overview of the simplest way to run EMMREM, and of the typical name conventions we use in order to streamline the process of running this software. We will do this by an example run of an SEP event. We assume that the user has already installed EMMREM properly, and has IDL on their machine. Let's begin.

Usually, a user would create a folder for a specific run of an event. We have identified a list of SEP events we are simulating, and identify them by a DOY and year - for example, 299_2003 would identify the Halloween events. The DOY is usually the day of biggest activity.

1. Let us use the Halloween event as an example. If the run is started on 02/12/2009, the user would create a folder named 'run_299_2003_20090212', where all input and output will reside. Assuming the user has an EP flux time series file for input (such as GOES_299_2003Small.txt), this would be copied over to the folder.

2. Then, the user would make copies of the configuration files:
$ makeInputParser.sh iConf
$ makeSpiceLoad.sh sConf
$ makeEPREM.sh eConf

3. The **iConf** file then gets modified for input (GOES_299_2003Small.txt) and output (earthSepIn.txt) files. InputParser is executed:
$ runInputParser.sh iConf

4. We will assume that the user has set three observers in the EPREM code - earth, moon, mars. Then, the **sConf** file has to be modified accordingly so a position file is created (spiceObservers.pos):
$ runSpiceLoad.sh sConf

5. The user next edits **eConf** to set the parameters for a specific run.

6. The user runs EPREM. To do this in the background, open a text editor; once in the writing area, hit ENTER, and then save this file as 'in' in the folder. Then, execute the following:
$ runEPREM.sh eConf < in > output.log &

This will run EPREM in the background, and save all output from the software into the file output.log.

7. Once the run, or 'experiment', is finished, the software produces the following files (again, assuming we have 'earth', 'moon', and 'mars' observers):
epremRunParams.txt - a summary of the most important constants and parameters.
epremStreamNodesLocs.dat - locations of the code grid cells.
earthObserver.txt
moonObserver.txt
marsObserver.txt

8. Next, we run a script to create files for plotting and for BRYNTRN input:
$ prepareFiles4plots.sh --makedirs

This command will create two folders - 'fluxPlots' to contain the flux time series plots, and 'bryn' to contain BRYNTRN input files and eventually, dose plots. In each of those, there's a folder for each observer.

9. To create flux plots, we go back to the main directory of the run (run_299_2003_20090212) and execute the following script (assuming that the user has IDL and convert installed):
$ makeFluxPlots.sh --makejpg --withdirs
It should execute successfully, and place .eps and .jpg files in each observer's folder inside the 'fluxPlots' directory.

10. To run BRYNTRN, the user has to go to each of the observer folders in the 'bryn' directory, and execute the following (here's an example for the earth observer in run_299_2003_20090212/bryn/earth):
$ runBRYNTRN.sh earthInput.dat > output.log &
This will cause BRYNTRN to be run as a background process, and will put all output into the file output.log. For the other observers, substitute with moonInput.dat and marsInput.dat. Many instances of BRYNTRN can run simultaneously. The output of each of those executions are several files, of which the most important is (for earth) earthOutput.plot (moonOutput.plot and marsOutput.plot, respectively.) These are the input files for creating the various dose plots.

12. Once BRYNTRN has been executed for all observers, the user (assumed to be situated in the main directory for the run, run_299_2003_20090212) can execute the following script to make plots of dose-related quantities in the appropriate observer folders in the 'bryn' directory:
$ makeDosePlots.sh --makejpg --withdirs

NB! The plot-making commands have many more execution options - here we have just shown their simplest use. For more insight about their

advanced use, see their full descriptions in Appendix B.

Appendix A:
A Quick Index of Shell Commands for Running EMMREM (in Alphabetical order)

All of the commands bellow can be called as they are, and have a help screen - that is, the user can call help
by running '[commandName] --help' (Example: plotDose.sh --help)

COMPILE/RUN commands
commands to compile and run the different executables in EMMREM
==========

makeBRYNTRN.sh - compiles BRYNTRN
makeEPREM.sh - compiles EPREM
makeInputParser.sh - compiles inputParser
makeMarsBryn.sh - compiles MarsBryn
makeMarsBrynAtm.sh - compiles MarsBrynAtm
makeOutputParser.sh - compiles outputParser
makeSourceLoader.sh – compiles sourceLoader
makeSpiceLoad.sh - compiles spiceLoad
prepareEPREM.sh - configures the EPREM code for the specific machine.
runBRYNTRN.sh - runs BRYNTRN
runEPREM.sh - runs EPREM
runInputParser.sh - runs inputParser
runMarsBryn.sh - runs MarsBryn
runMarsBrynAtm.sh - runs atmospheric MarsBryn code.
runOutputParser.sh - runs outputParser
runSourceLoader.sh – runs sourceLoader
runSpiceLoad.sh - runs spiceLoad

ADDITIONAL CONTROL/MAINTENANCE COMMANDS
For all these commands, a help screen is available for further information and specific input requirements, by running '[commandName] --help' (Example: plotDose.sh –help)

---

batchMakeEMMREM.sh – This script compiles all EMMREM executables.

batchRunTestCase.sh - This command runs a test case for EPREM, to check that the code is compiled and runs correctly. Again, see the help screen for more information.

fixSpiceDirs.sh - Fixes the directory dependencies inside the spiceLoad software.

getGcrDose.sh - This command produces an average GCR dose and dose equivalent rate value for a given year, radius, and Al and H2O shielding depths. It looks up tables of potentials and doses.

inspectObserver.sh - Sometimes, the user wants to see how a particular EPREM run is progressing, and examine the output in a comfortable form. This command can create encapsulated output from observer files in mid-run, producing flux files, as well as flux time series plots.

killEPREM.sh - This command kills all EPREM processes on the machine. This is very useful when running on a machine with many processors.

makeDosePlots.sh – After preparefiles4plots.sh and BRYNTRN are run, this command can create all required dose-related time series plots.

makeFluxPlots.sh – After preparefiles4plots.sh is run, this command can create all required flux-related time series plots from the output.

mrm_dev.sh - Sets the execution environment, so that all commands can be called as unix commands on the command prompt. Execute 'mrm_dev' every time you use EMMREM.

plotObsPosHelio.sh - This command creates maps of the positions of the various observers in the heliosphere for specific events, as well as a map of the boundary region for the events.

prepareFiles4Plots.sh - After EPREM is run, this command converts all observer files to flux files for plotting and for BRYNTRN input. Run 'prepareFiles4Plots.sh --help' for more info.

svnUpEMMREM.sh - Updates EMMREM to the latest version (provided you have SVN installed.)

PLOTTING COMMANDS
The following commands, divided in several categories, are used to create plots of the EPREM and BRYNTRN output.
NB! An IDL installation is required to execute these commands!
For all these, a help screen is available for further information and specific input requirements, by running '[commandName] --help' (Example: plotDose.sh --help)
==========

Commands to plot fluxes at different observers/spacecraft:
plotAccuDose.sh - plots rates and accumulated dose/dose equiv. at an observer/spacecraft

plotDose.sh - plots the dose/dose equiv. rates at an observer/spacecraft
plotEarthDataAccuDose.sh - plots rates and accumulated dose/dose equiv. at GOES spacecraft
plotEarthDataDose.sh - plots the dose/dose equiv. rates at GOES spacecraft
plotEarthDataFlux.sh - plots the flux from GOES spacecraft
plotEarthFlux.sh - plots the flux from the Earth observer and GOES spacecraft
plotEarthObsAccuDose.sh - plots rates and accumulated dose/dose equiv. at Earth Observer
plotEarthObsDose.sh - plots the dose/dose equiv. rates at Earth Observer
plotEarthObsFlux.sh - plots the flux from the Earth observer
plotFluxOne.sh - plots the flux from a single generic observer/spacecraft
plotFluxTwo.sh - plots the flux from two generic observers/spacecraft
plotMarsObsAccuDose.sh - plots rates and accumulated dose/dose equiv. at Mars Observer
plotMarsObsDose.sh - plots the dose/dose equiv. rates at Mars Observer
plotMarsObsFlux.sh - plots the flux from the Mars observer
plotMarsTopBottomAccuDose.sh - plots rates and accumulated dose/dose equiv. for different elevations at the Mars Atmosphere
plotMarsTopBottomDose.sh - plots the dose/dose equiv. rates for different elevations at the Mars Atmosphere
plotMoonObsAccuDose.sh - plots rates and accumulated dose/dose equiv. at Moon Observer
plotMoonObsDose.sh - plots the dose/dose equiv. rates at Moon Observer
plotMoonFlux.sh - plots the flux from the Moon observer
plotUlyssesDataAccuDose.sh - plots rates and accumulated dose/dose equiv. at Ulysses spacecraft
plotUlyssesDataDose.sh - plots the dose/dose equiv. rates at Ulysses spacecraft
plotUlyssesDataFlux.sh - plots the flux from Ulysses spacecraft
plotUlyssesFlux.sh - plots the flux from the Ulysses observer and spacecraft
plotUlyssesObsAccuDose.sh - plots rates and accumulated dose/dose equiv. at Ulysses Observer
plotUlyssesObsDose.sh - plots the dose/dose equiv. rates at Ulysses Observer
plotUlyssesObsFlux.sh - plots the flux from the Ulysses observer
commands to plot dose/dose equiv. at different observers/spacecraft:

Appendix B.
Complete Descriptions of Shell Commands for Running EMMREM(in alphabetical order)

COMPILE/RUN commands

commands to compile and run the different executables in EMMREM:

==========

makeBRYNTRN.sh

```
DESCRIPTION:
 This script prepares the BRYNTRN_4_EMMREM executable for running.

USAGE:
      makeBRYNTRN.sh         <Prepare BRYNTRN4_MARSREM to run.>
      makeBRYNTRN.sh -u      <Setting this option forces the script
                        to compile using the Fortran compiler
                        for IBM P4 machines, 'f77'. The default
                        command is 'g77 -fno-automatic'.>
```

==========

==========

makeEPREM.sh

```
DESCRIPTION:
 This script prepares the EPREM executable for running. It must be run
 only after a Makefile has been generated via autoconf/automake.

USAGE:
 makeEPREM.sh               <Prepare the EPREM model for running.>
 makeEPREM.sh --help        <Display this HELP screen only.>
 makeEPREM.sh -c            <Force the script to 'make clean' only.>
 makeEPREM.sh [config_file] <Make a local copy of a sample EPREM configure
                          file with user-supplied name only.>
```

==========

==========

makeInputParser.sh

```
DESCRIPTION:
This script prepares inputParser -  a program, which converts
input SEP flux time series from the external EMMREM data format
to distribution function timeseries in the internal EMMREM data
format. Running the inputParser requires a configure file as input.
This script can generate a sample configure file with default
parameters in your local folder, if supplied with a filename at runtime:
$ makeInputParser.sh [your_preferred_config_filename]

Running the inputParser also requires an SEP particle intensity
time series in the external ascii EMMREM format.
To get a sample file with this format in your local directory:
$ makeInputParser.sh -s
```

```
USAGE:
makeInputParser.sh               <Make the inputParser ready only.>
makeInputParser.sh -s            <Get a copy of sample input data file only.>
makeInputParser.sh [config_file] <Copy default params into this file only.>
makeInputParser.sh --help        <Display this HELP screen only.>

CONFIG_FILE PARAMETERS:
      -numEsteps         <number of energy steps>
      -eMin              <minimum energy value in MeV>
      -eMax              <maximum energy value in MeV>
      -debugMode         <0 - no printouts, 1 - printouts>
      -inputFilename     <input data filename to be parsed>
      -epremOutFname     <output filename for EPREM input>
==========================================================================
```

```
==========================================================================
```

makeMarsBryn.sh

```
DESCRIPTION:
 This script prepares the BRYNTRN4_MARSREM executables for running.

USAGE:
     makeMarsBryn.sh          <Prepare BRYNTRN4_MARSREM to run.>
     makeMarsBryn.sh --help   <Display this HELP screen only.>
     makeMarsBryn.sh -u       <Setting this option forces the script
                         to compile using the Fortran compiler
                         for IBM P4 machines, 'f77'. The default
                         command is 'g77 -fno-automatic'.>

==========================================================================
```

```
==========================================================================
```

makeMarsBrynAtm.sh

```
DESCRIPTION:
 This script prepares the BRYNTRN4_MARSREM (Atmosphere) executable for running.

USAGE:
     makeMarsBrynAtm.sh          <Prepare BRYNTRN4_MARSREM to run.>
     makeMarsBrynAtm.sh --help   <Display this HELP screen only.>
     makeMarsBrynAtm.sh -u       <Setting this option forces the script
                         to compile using the Fortran compiler
                         for IBM P4 machines, 'f77'. The default
                         command is 'g77 -fno-automatic'.>

==========================================================================
```

```
==========================================================================
```

makeOutputParser.sh

```
DESCRIPTION:
This script prepares outputParser -  a program, which converts
the output of the EPREM model - distribution function timeseries in the
internal EMMREM ascii format for a single observer/location - to various
outputs for plotting and further analysis.
Running the outputParser requires a configure file as input.
```

```
This script can generate a sample configure file with default
parameters in your local folder, if supplied with a filename at runtime:
$ makeOutputParser.sh [your_preferred_config_filename]

Running the outputParser also requires an "observer file" with the
SEP distribution function time series as input.
To get a sample file with this format in your local directory:
$ makeOutputParser.sh -s

USAGE:
makeOutputParser.sh               <Make the outputParser ready to run only.>
makeOutputParser.sh -s      <Get a local copy of sample observer file only.>
makeOutputParser.sh [config_file] <Copy default params into this file only.>
makeOutputParser.sh --help        <Display this HELP screen only.>

CONFIG_FILE PARAMETERS:
     -numEstepsBryn         <number of energy steps for BRYNTRN input.>
     -numEstepsVisu         <number of energy steps to use in plots.>
     -timeRes               <time resolution for the flux output, in min.>
     -debugMode             <0 - no printouts, 1 - printouts>
     -inputFilename         <input data filename>
     -energyValsBryn        <Energy values to use for BRYNTRN input [MeV]>
                               (Energies must be separated by commas.)
     -energyValsVisu        <Energy values to use for plotting [MeV]>
                               (Energies must be separated by commas.)
     -brynOutFname          <output filename for BRYNTRN input>
     -fluxOutFname          <output filename for flux timeseries>
     -distOutFname          <output filename for dist function timeseries.>
================================================================================

================================================================================
```

makeSourceLoader.sh

```
DESCRIPTION:
This script prepares sourceLoader -  a program, which creates an artificial
energetic particle (EP) source of flux in the external EMMREM data format.
Running the sourceLoader requires a configure file as input.
This script can generate a sample configure file with default
parameters in your local folder, if supplied with a filename at runtime:
$ makeSourceLoader.sh [your_preferred_config_filename]

USAGE:
makeSourceLoader.sh               <Make the sourceLoader ready only.>
makeSourceLoader.sh [config_file] <Copy default params into this file only.>
makeSourceLoader.sh --help        <Display this HELP screen only.>

CONFIG_FILE PARAMETERS:
      -sourceName        <Source name for EPREM input>
      -sourceOutFname    <output filename for EPREM input>
      -debugMode         <0 - no printouts, 1 - printouts>
      -RPos, ThetaPos,PhiPos   <Beginning and Ending positions of the source.>
      -RRange, ThetaRange, PhiRange   <Range of activity of the source. >
      -startTime, endTime   <start and end times for the source time series.>
      -dataCadence       <time series cadence.>
      -weibullBaseFlux, weibullKappa, weibullAlpha
                        <Parameters for the simulated Weibull differential flux
                         spectrum for the source, as presented in Xapsos et
al.(2000)>
```

```
      -numEsteps         <number of energy steps>
      -eMin              <minimum energy value in MeV>
      -eMax              <maximum energy value in MeV>
      -numSpecies        <number of species to be included in the time series.>
      -mass              <an array of masses for the species [proton mass]>
      -charge            <an array of charges for the species [electron charge]>
      -numMuSteps        <number of steps in pitch angle for the time series.>
      -mu                <an array of pitch angles for the time species.>
```

==========

==========

makeSpiceLoad.sh

```
DESCRIPTION:
 This script prepares spiceLoad -  a program, which creates
 time series of Solar System objects' positions and velocities
 as needed for input into the EPREM model for accurate position
 determination. The program utilizes the SPICE toolkit and kernel data,
 developed by the NASA NAIF. Running the spiceLoad requires a configure
 file as input. This script can generate a sample configure file with default
 parameters in your local folder, if supplied with a filename at runtime:
 $ makeSpiceLoad.sh [your_preferred_config_filename]

USAGE:
  makeSpiceLoad.sh               <Make the spiceLoad ready only.>
  makeSpiceLoad.sh [config_file] <Copy default params into [config_file] only.>
  makeSpiceLoad.sh --help        <Display this HELP screen only.>

CONFIG_FILE PARAMETERS:
  -startTime         <starting time for position data, in JD>
  -stopTime          <starting time for position data, in JD>
  -timeRes           <time resolution of the data, in minutes>
  -debugMode         <0 - no printouts, 1 - printouts>

  -referenceFrame    <The code for the reference frame to be used.
                      Default is HCI.>
       Currently available reference frames are:
       "J2000"(inertial) - Earth mean equator, dynamical equinox of J2000.
       "B1950"(inertial) - Earth mean equator, dynamical equinox of B1950.
       "ECLIPDATE" - Mean Ecliptic of Date.
       "HCI" - Heliocentric Inertial.
       "HEE" - Heliocentric Earth Ecliptic.
       "HEEQ" - Heliocentric Earth Equatorial.
  More information on SPICE find at http://naif.jpl.nasa.gov/naif/

  -referenceObserver    <The object, with reference to which the positions
                         are recorded.>
  -outFilename          <The name of the file to which the positions data
                         are written.>
  -numTargets           <# of objects, for which position data will be
                         produced.>
  -targetCodes          <A list of target object codes. The codes' number
                         must equal numTargets.>
       The currently available objects for input and their codes are:
                        1 - MERCURY
                        2 - VENUS
```

```
                        3 - EARTH
                        4 - MARS
                        5 - JUPITER
                        6 - SATURN
                        7 - URANUS
                        8 - NEPTUNE
                        9 - PLUTO
                        301 - MOON
                        -55 - ULYSSES
                        -23 - PIONEER 10
==========================================================================

==========================================================================
```

prepareEPREM.sh

```
DESCRIPTION:
This script loads and executes automake/autoconf on your machine for
the EPREM code. These programs check that your machine has all the
required libraries/packages/compilers to run EPREM, and creates
the appropriate Makefile to compile EPREM properly.

USAGE:
     prepareEPREM.sh           <Prepare and run automake/autoconf.>
     prepareEPREM.sh --prep    <Prepare automake/autoconf only.>
     prepareEPREM.sh --conf    <Run automake/autoconf only.>
     prepareEPREM.sh --help    <Display this HELP screen only.>

==========================================================================

==========================================================================
```

runBRYNTRN.sh

```
DESCRIPTION:
 This script runs the BRYNTRN looping code with
 some input file. The input file must be in the specific
 time series format that BRYNTRN4_EMMREM requires. It can
 be obtained by running "outputParser" on an observer
 file. For more info, try
 $ outputParser_ST2_run.sh --help

USAGE:
  runBRYNTRN.sh --help           <Runs the HELP option only.>
  runBRYNTRN.sh -s               <Copies an example input file to your local
                              folder only.>
  runBRYNTRN.sh [observer]Input.dat <Runs BRYN4_EMMREM with the specified input.
          Here [observer] is the name of an observer, such as 'earth' or 'moon'.>

OUTPUT:
 There are two output files from running this code:
 output.final       <Dose/Dose Eq. rate time series for different Al/Water
                     shielding depths>
 [observer]Output.plot        <Dose/Dose Eq. rate time series for different Al/Water
                     depths for plotting>
==========================================================================

==========================================================================
```

runEPREM.sh

```
DESCRIPTION:
 This script runs the EPREM model with multi-processor support via MPI.
 Running the  requires a configure file as input.
 To get a copy of a sample configure file in your local folder do:
 $ makeEPREM.sh [configure_file]

 Running the EMMREM model requires two input files - an SEP distribution
 function time series file, which is the output of running the "inputParser"
 program, and an observer positions time series file, which is the output of
 running the "spiceLoad" program. The locations of those two files must be
 specified in the configure file prior to running the EPREM executable.


USAGE:
   runEPREM.sh --help
   runEPREM.sh -s <Remove existing NetCDF restart file.>
   runEPREM.sh [configure_file] <Run EPREM with a configure file.>


OPTIONAL PARAMETERS:
runEPREM.sh takes several parameters, which directly modify the structure of the
model (they modify runtime constants in the EPREM Makefile.) The user is advised
caution when using these arguments.
The arguments can be called as --option [value], one after another, but they
should be placed before the [configure_file] argument.

   --procs [#_procs]     <Change the number of processors - default is 8.>
   --facecols [#_facecols] <Change the number of face columns in the model grid.>
   --facerows [#_facerows] <Change the number of face rows in the model grid.>
   --shells [#_shells]   <Change number of shells in the model grid.>
   --musteps [#_musteps] <Change number of pitch angle steps in the model.>
   --esteps [#_esteps]  <Change number of energy steps in the model.>
   --maxobs [#_maxobs]  <Change the maximum number of observers in the model.
                         Setting this also changes number of faces accordingly.>
If any of these options is evoked, this script will recompile EPREM so that
the changes can take effect, and then proceed to run it.
================================================================================

================================================================================
```

runInputParser.sh

```
DESCRIPTION:
 This script runs inputParser - a program, which converts
 input SEP flux timeseries from the external EMMREM data format
 to distribution function timeseries in the internal EMMREM data
 format. Running the inputParser requires a configure file as input.
 To get a copy of a sample configure file in your local folder do:
 $ makeInputParser.sh [your_preferred_config_filename]

 Running the inputParser also requires an SEP particle intensity
 time series in the external ascii EMMREM format.
 To get a sample file with this format in your local directory:
 $ makeInputParser.sh -s


USAGE:
     runInputParser.sh -t           <Do a test run of inputParser only.>
```

```
     runInputParser.sh [config_file] <Run inputParser with the configure file.>
     runInputParser.sh --help        <Display this HELP screen only.>

CONFIG_FILE PARAMETERS:
     -numEsteps             <number of energy steps>
     -eMin                  <minimum energy value in MeV>
     -eMax                  <maximum energy value in MeV>
     -debugMode             <0 - no printouts, 1 - printouts>
     -inputFilename         <input data filename to be parsed>
     -epremOutFname         <output filename for EPREM input>

OUTPUT:
     The output of running this script is an ascii file with the
     timeseries converted from particle intensity to distribution
     function values, in the internal EMMREM format. It will be saved
     in the name and folder specified in the "epremOutFname" option
     in the configure file.
=========================================================================
```

runMarsBryn.sh

```
DESCRIPTION:
 This script runs the BRYN4_MARSREM looping code with
 some input file. The input file must be in the specific
 time series format that BRYNTRN4_EMMREM requires. It can
 be obtained by running "outputParser" on an observer
 file. For more info, try
 $ outputParser_ST2_run.sh --help
 This also runs the Martian atmospheric angular distribution code,
 SUMMARS.

USAGE:
     runMarsBryn.sh --help             <Runs the HELP option only.>
     runMarsBryn.sh -s                 <Copies an example input file to the
                                   local folder only.>
     runMarsBryn.sh [input_filename]   <Runs BRYN4_MARSREM with the specified
                                   input.>

OUTPUT:
 There are several output files from running this code:
 output.final       <Dose/Dose Eq. rate time series for different
                     material/atmosphere depths>
 output.mars        <Dose/Dose Equiv. at different mean heights of
                     Martian atmosphere per timestep>
 outputDose.plot    <Dose rate time series for different
                     material/atmosphere depths for plotting>
 outputDoseEq.plot  <Dose rate time series for different
                     material/atmosphere depths for plotting>

=========================================================================
```

runMarsBrynAtm.sh

```
DESCRIPTION:
 This script runs the BRYN4_MARSREM looping code with
```

```
 some input file. The input file must be in the specific
 time series format that BRYNTRN4_EMMREM requires. It can
 be obtained by running "outputParser" on an observer
 file. For more info, try
 $ outputParser_ST2_run.sh --help
 This also runs the Martian atmospheric angular distribution code,
 SUMMARS.

USAGE:
     runMarsBrynAtm.sh --help             <Runs the HELP option only.>
     runMarsBrynAtm.sh -s                 <Copies an example input file to the
                                  local folder only.>
     runMarsBrynAtm.sh [input_filename]   <Runs BRYN4_MARSREM with the specified
                                  input.>

OUTPUT:
 There are several output files from running this code:
 output.final       <Dose/Dose Eq. rate time series for different
                     material/atmosphere depths.>
 output.mars        <Dose/Dose Equiv. at different mean heights of
                     Martian atmosphere per timestep.>
 output.plot    <Dose and Dose Equiv. rate time series for different
                     material/atmosphere shielding for plotting.>
 outputTopBottom.plot   <Dose and Dose Equivalent rates at different
                         Mars atmospheric elevations for plotting.>
================================================================================



================================================================================
```

runOutputParser.sh

```
DESCRIPTION:
This script runs outputParser -  a program, which converts
input SEP flux timeseries from the external EMMREM data format
to distribution function timeseries in the internal EMMREM data
format. Running the outputParser requires a configure file as input.
To get a copy of a sample configure file in your local folder do:
$ makeOutputParser.sh [your_preferred_config_filename]

Running the outputParser also requires an SEP particle intensity
time series in the external ascii EMMREM format.
To get a sample file with this format in your local directory:
$ makeOutputParser.sh -s


USAGE:
  runOutputParser.sh -t            <Do a test run of outputParser only.>
  runOutputParser.sh [config_file] <Run outputParser with the configure file.>
  runOutputParser.sh --help        <Display this HELP screen only.>

CONFIG_FILE PARAMETERS:
  -numEstepsBryn         <number of energy steps for BRYNTRN input.>
  -numEstepsVisu         <number of energy steps to use in plotting.>
  -timeRes               <time resolution for the flux output [min].>
  -debugMode             <0 - no printouts, 1 - printouts>
  -inputFilename         <input data filename>
  -energyValsBryn        <Energy values for BRYNTRN input [MeV]>
```

```
                          (Energies must be separated by commas.)
  -energyValsVisu        <Energy values to use for plotting [MeV]>
                           (Energies must be separated by commas.)
  -brynOutFname          <output filename for BRYNTRN input>
  -fluxOutFname          <output filename for flux timeseries>
  -distOutFname          <output filename for dist function timeseries.>

OUTPUT:
     There are several outputs from running "outputParser". The first
     is a special output of energy spectra time series, generated for
     the looping version of BRYNTRN. It will be saved in the file and
     folder specified in the ("brynOutFname") option in the config
     file. The second output is a flux time series for the observer at
     different energies, in a format for plotting. It will be saved in
     the file and folder specified in the ("fluxOutFname") option in
     the config file. The third output is a distribution function time
     series for the observer, in a format for plotting. It will be
     saved in the file and folder specified in the "distOutFname"
     option in the config file.
=========================================================================

=========================================================================
```

runSourceLoader.sh

```
DESCRIPTION:
This script runs sourceLoader -  a program, which creates an artificial
energetic particle (EP) source of flux in the external EMMREM data format.
Running the sourceLoader requires a configure file as input.
This script can generate a sample configure file with default
parameters in your local folder, if supplied with a filename at runtime:
$ makeSourceLoader.sh [your_preferred_config_filename]


USAGE:
     runSourceLoader.sh -t            <Do a test run of sourceLoader only.>
     runSourceLoader.sh [config_file] <Run sourceLoader with the configure file.>
     runSourceLoader.sh --help        <Display this HELP screen only.>

CONFIG_FILE PARAMETERS:
      -sourceName        <Source name for EPREM input>
      -sourceOutFname    <output filename for EPREM input>
      -debugMode         <0 - no printouts, 1 - printouts>
      -RPos, ThetaPos,PhiPos   <Beginning and Ending positions of the source.>
      -RRange, ThetaRange, PhiRange   <Range of activity of the source. >
      -startTime, endTime   <start and end times for the source time series.>
      -dataCadence       <time series cadence.>
      -weibullBaseFlux, weibullKappa, weibullAlpha
                        <Parameters for the simulated Weibull differential flux
                         spectrum for the source, as presented in Xapsos et
al.(2000)>
      -numEsteps         <number of energy steps>
      -eMin              <minimum energy value in MeV>
      -eMax              <maximum energy value in MeV>
      -numSpecies        <number of species to be included in the time series.>
      -mass              <an array of masses for the species [proton mass]>
      -charge            <an array of charges for the species [electron charge]>
      -numMuSteps        <number of steps in pitch angle for the time series.>
      -mu                <an array of pitch angles for the time species.>
```

```
OUTPUT:
     The output of running this script is an ascii file with an artificial
     time series of differential particle flux with a Weibull form spectrum
     in the external EMMREM format. It will be saved in the name and folder
     specified in the "sourceOutFname" option in the configure file.
=========================================================================
```

```
=================================================================
```

runSpiceLoad.sh

```
DESCRIPTION:
 This script prepares spiceLoad - a program, which creates
 time series of Solar System objects' positions and velocities
 as needed for input into the EPREM model for accurate position
 determination. The program utilizes the SPICE toolkit and
 kernel data, developed by the NASA NAIF. Running the spiceLoad
 requires a configure file as input. This script can generate a
 sample configure file with default parameters in your local
 folder, if supplied with a filename at runtime:
 $ makeSpiceLoad.sh [your_preferred_config_filename]


USAGE:
  runSpiceLoad.sh -t            <Do a test run of spiceLoad only.>
  runSpiceLoad.sh [config_file] <Run spiceLoad with the configure file.>
  runSpiceLoad.sh --help        <Display this HELP screen only.>

CONFIG_FILE PARAMETERS:
  -startTime         <starting time for position data, in JD>
  -stopTime          <starting time for position data, in JD>
  -timeRes           <time resolution of the data, in minutes>
  -debugMode         <0 - no printouts, 1 - printouts>
  -referenceFrame    <The code for the reference frame to be used.
                      Default is HCI.>
                Currently available reference frames are:
                "J2000"(inertial) - Earth mean equator,
                                    dynamical equinox of J2000.
                "B1950"(inertial) - Earth mean equator,
                                    dynamical equinox of B1950.
                "ECLIPDATE" - Mean Ecliptic of Date.
                "HCI" - Heliocentric Inertial.
                "HEE" - Heliocentric Earth Ecliptic.
                "HEEQ" - Heliocentric Earth Equatorial.
                More information on SPICE find at
                http://naif.jpl.nasa.gov/naif/
  -referenceObserver    <The object, with reference to which the
                         positions are recorded.>
  -outFilename          <The name of the file to which the
                         positions data are written.>
  -numTargets           <# of objects, for which position data will
                            be produced.>
  -targetCodes          <A list of target object codes. The codes'
                         number must equal numTargets>
      The currently available objects for input and their codes are:
                        1 - MERCURY
                        2 - VENUS
                        3 - EARTH
```

```
                        4 - MARS
                        5 - JUPITER
                        6 - SATURN
                        7 - URANUS
                        8 - NEPTUNE
                        9 - PLUTO
                        301 - MOON
                        -55 - ULYSSES
                        -23 - PIONEER 10

OUTPUT:
  The output of running this script is an ascii file with the
  timeseries of the desired objects' positions and velocities,
  in the common external EMMREM ascii format. It will be saved
  in the name and folder specified in the "outFilename" option
  in the configure file.

=========================================================================
```

ADDITIONAL CONTROL/MAINTENANCE COMMANDS

Commands for controlling several of the above commands, or for other administrative tasks.

```
=========================================================================
```

batchMakeEMMREM.sh

```
DESCRIPTION:
A script for batch compiling and preparing EMMREM for running.
It compiles inputParser, outputParser, spiceLoad, BRYNTRN4_EMMREM,
BRYNTRN4_MARS, and EPREM.
It also copies the configure files needed to run some of those executables
to the local folder for the user to examine.


USAGE:
     batchMakeEMMREM.sh --help    <Runs the HELP option only.>
     batchMakeEMMREM.sh -s        <Copies example input files for all EMMREM
                                  components that need them to local folder.>
     batchMakeEMMREM.sh           <Compiles the executables.>


OUTPUT:
Based on the option there are different outputs to this script in the local
folder.
If the script is executed as is, the output files are:
inConf - a configure file for the inputParser
outConf - a configure file for the outputParser
spcConf - a configure file for the spiceLoad
epremConf - a configure file for EPREM

If the script is executed with the -s option, then the output files are:
bryntrnInput.example - Input file format for BRYNTRN4_EMMREM, BRYNTRN4_MARS
earthObserver.good - Input file format for outputParser
SEPdataFormat.txt - Input file format for inputParser
=========================================================================
```

```
================================================================================
```

batchRunTestCase.sh

```
DESCRIPTION:
This script will run the default test case of EPREM.


USAGE:
   batchRunTestCase.sh --help
   batchRunTestCase.sh -s <Remove existing NetCDF restart file.>
   batchRunTestCase.sh [configure_file] <Run EPREM with a configure file.>


================================================================================

================================================================================
```

fixSpiceDirs.sh

```
DESCRIPTION:
 This script will fix the dependencies on SPICE libraries of the spiceLoad
 code. Specifically, it will change commonIncludes.h and Makefile so that
 spiceLoad can find the SPICE libraries and header files. The user must input
 the location of the SPICE distribution installation main directory. That
 folder must contain the subdirs include/ and lib/.


USAGE:
   fixSpiceDirs.sh --help  <Display this HELP screen.>
   fixSpiceDirs.sh /dir/where/spice/is/installed  <Fix SPICE directory
                                        dependencies in the spiceLoad code.>
================================================================================

============================================================
```

getGcrDose.sh

```
DESCRIPTION:
scrName:  Script for getting GCR dose and dose Equivalent.
The input to this script is a year, radial distance (AU),
Aluminum and water thickness (g/cm^2), for which the gcr dose
and dose equivalent are calculated from tables.


USAGE:
   getGcrDose.sh     <Execute the script with default input.>
   getGcrDose.sh -q  <Execute the script quietly.>
   getGcrDose.sh --help  <Display this HELP screen.>


OPTIONAL PARAMETERS:
   --year [year]  <Supply a custom year for the GCR dose calc.>
   --radius [radius]  <Supply a custom radial distance (in AU).>
   --aldepth [aldepth]  <Supply a custom Aluminum depth (g/cm^2).>
   --waterdepth [waterdepth]  <Supply a custom water depth (g/cm^2).>


OUTPUT:
The output of this script is the average dose (cGy/s) and dose
```

equivalent(cSv/s) rates from GCRs for the specified year, at the
specified radial distance, and for the specified shielding depths.
==========

==========

inspectObserver.sh

DESCRIPTION:
 This script will create a complete observer file from a partially complete
 observer file. The reason is to be able to visually inspect the progress
 of EPREM for a certain observer via plots.

USAGE:
   inspectObserver.sh --help
   inspectObserver.sh [observerFile] <Create a standalone observer file from input.>
   inspectObserver.sh --flux [observerFile] <Create a standalone observer file
                                  from input, and a flux file too.>
   inspectObserver.sh --flux --plot [observerFile] <Create a standalone observer
file
                                  from input, make a plot from it.>

(Example: inspectObserver.sh --plot earthObserver.txt)

OUTPUT:
This script outputs a standalone observer file with the name
[observerFile].inspect
If the --plot option is also selected, there will be an additional file:
[observerName]Flux.eps
For example - earthObserver.txt.inspect , earthobsFlux.eps

==========

==========

killEPREM.sh

DESCRIPTION:
 This script will kill all 'eprem' processes running on the local
 machine. It uses ps, wc, grep, and sed.

USAGE:
 killEPREM.sh   <kill all EPREM processes.>
 killEPREM.sh --help   <Display this HELP screen only.>

==========

==========

makeDosePlots.sh

DESCRIPTION:
This script creates all the required plots for BRYNTRN output for all observers.
This script should be run after the output files from BRYNTRN for all observers have
been created. It creates plots for Dose/Dose Equivalent, and their rates over the
duration of an event.

INPUT:

```
The input for this script is any BRYNTRN plot output file, i.e.
[observer]Output.plot,
where [observer] is the observer name, such as 'earth' or 'moon'.


USAGE:
      makeDosePlots.sh                    <Run the script as it is.>
      makeDosePlots.sh --help             <Show this HELP screen only.>
      makeDosePlots.sh -x 'MM/DD/YYYY HH:MM:SS' 'MM/DD/YYYY HH:MM:SS'
                  <Run the script with a specific time range.>
      The above is the complete format for the time range. This
      software accepts the short format 'MM/DD/YYYY' as well.
(Example: makeDosePlots.sh -x '05/02/1998' '05/13/1998 12:23')

      makeDosePlots.sh -t [xtimeLayerCode]
                  <plot different time units on the x-axis:>
                   1: HH:MM
                   2: MM/DD/YY
                   3: Both HH:MM and MM/DD/YY
(Example: makeDosePlots.sh -t 2)

       makeDosePlots.sh --makejpg   <Create jpg files as well. Assumes 'convert' is
installed.>

      makeDosePlots.sh --withdirs         <Assume folders for each input file
exist.>

   This last option assumes there exist in the current folder the folders 'bryn'
   and 'fluxPlots', and in them one folder for each observer, where the input files
   are stored. This structure can be automatically obtained if the user runs
   the prepareFiles4plots.sh command prior to running this script.

--withgcr [year] [radius(AU)]
    <Add GCR dose/s (and dose equivalent/s) to the SEP dose/s
     (and dose equivalent/s) and create additional plots of
     normalized  GCR (GCR/(GCR+SEP)) dose and dose equivalent rates.>
  NB! Unfortunately, since this option only takes one radius, it should
      be used on one-on-one basis for specific observers.


The above options can be combined, with the -x option preceding
the -t one, and so on:
makeDosePlots.sh -x '05/02/1998' '05/13/1998 12:23' -t 2 --withdirs --withgcr 1998
0.99


OUTPUT:
For each observer (we'll use earth as an example), the script creates the following
files:
earthobsGcrBFODose.eps
earthobsGcrBFODoseEq.eps
earthobsAccuDose.eps
earthobsAccuDoseEq.eps
earthobsGcrDose.eps
earthobsGcrDoseEq.eps
earthobsBFOAccuDose.eps
earthobsBFOAccuDoseEq.eps
earthobsGcrSkinAccuDose.eps
```

```
earthobsGcrSkinAccuDoseEq.eps
earthobsBFODose.eps
earthobsBFODoseEq.eps
earthobsGcrSkinDose.eps
earthobsGcrSkinDoseEq.eps
earthobsDose.eps
earthobsDoseEq.eps
earthobsSkinAccuDose.eps
earthobsSkinAccuDoseEq.eps
earthobsGcrAccuDose.eps
earthobsGcrAccuDoseEq.eps
earthobsSkinDose.eps
earthobsSkinDoseEq.eps
earthobsGcrBFOAccuDose.eps
earthobsGcrBFOAccuDoseEq.eps

All of these are Encapsulated PostScript files, and if the --makejpg option is
supplied, they can be converted to jpg files (provided the user has the 'convert'
program installed.)

The following suffixes are used:
Gcr - denotes including Gcr doses;
BFO - blood-forming-organ dose;
Skin - skin dose;
Accu - includes dose as well as dose rate plots;
DoseEq - Dose Equivalent;

================================================================================

================================================================================
```

makeFluxPlots.sh

```
DESCRIPTION:
This script creates all flux time series plots for EPREM output for all observers.
This script should be run after all the required flux files from EPREM have
been created.


USAGE:
      makeFluxPlots.sh                    <Run the script as it is.>
      makeFluxPlots.sh --help             <Show this HELP screen only.>
      makeFluxPlots.sh -x 'MM/DD/YYYY HH:MM:SS' 'MM/DD/YYYY HH:MM:SS'
                  <Run the script with a specific time range.>
      The above is the complete format for the time range. This
      software accepts the short format 'MM/DD/YYYY' as well.
(Example: makeFluxPlots.sh -x '05/02/1998' '05/13/1998 12:23')

      makeFluxPlots.sh -t [xtimeLayerCode]
                  <plot different time units on the x-axis:>
                   1: HH:MM
                   2: MM/DD/YY
                   3: Both HH:MM and MM/DD/YY
(Example: makeFluxPlots.sh -t 2)

      makeFluxPlots.sh --makejpg         <Make JPEG files. Assumes 'convert' is
installed.>

      makeFluxPlots.sh --withdirs         <Assume folders for each input file
```

```
exist.>

   This last option assumes there exist in the current folder the folders ''
   and 'fluxPlots', and in them one folder for each observer, where the input files
   are stored. This structure can be automatically obtained if the user runs
   the prepareFiles4plots.sh command prior to running this script.

The above options can be combined, with the -x option preceding
the -c one, and so on:
makeFluxPlots.sh -x '05/02/1998' '05/13/1998 12:23' -t 2 --makejpg --withdirs

OUTPUT:
For each observer (we'll take 'earth' as an example), the script produces the
following plot file:
earthobsFlux.eps
This is an Encapsulated PostScript file.
If there is an observer for a position in which there is also a spacecraft with
measurements,
the script will also produce a comparison plot, called
earthobsFluxCompare.eps
These two files can also be automatically converted to jpg format, if the user
selects the --makejpg
option (and has the 'convert' program installed.)
================================================================================

================================================================================
```

prepareFiles4Plots.sh

```
DESCRIPTION:
 This scrtipt prepares various flux files for plotting, as well as prepares
 flux input for BRYNTRN. The script requires that Observer files are present
 in the local folder. In particular, it accepts any of the following names:
   earthObserver.txt
   moonObserver.txt
   marsObserver.txt
   ulyssesObserver.txt
   epremSepIn.txt
   ulyssesIn.txt

OUTPUT:
 The output depend on which of the above files are present in the local
 folder. At most the script will produce the following plot-ready files
 (on the left) and BRYNTRN input files (right):
  PLOT-READY FILES                    BRYNTRN INPUT FILES
   efluxes.txt                        earthInput.dat
   mofluxes.txt                       moonInput.dat
   mfluxes.txt                        marsInput.dat
   ufluxes.txt                        ulyssesInput.dat
   sepDatafluxes.txt                  sepDataInput.dat
   ulyssesDatafluxes.txt              ulyssesDataInput.dat


USAGE:
      prepareFiles4Plots.sh                    <Run the script as it is.>
      prepareFiles4Plots.sh --help             <Show this HELP screen only.>
      prepareFiles4Plots.sh -e '[energy list]'   <Run the script, supply
                                                   energies to plot (MeV)>
      (Example: prepareFiles4Plots.sh -e '10.0 20.0 30.0 50.0')
```

```
      prepareFiles4Plots.sh --makedirs        <Make folders for the output files>

   This last option creates in the current folder the folders 'bryn'
   and 'fluxPlots', and in them one folder for each observer, where it
   stores the output files.
================================================================================
```

```
================================================================================
```

svnUpEMMREM.sh

```
DESCRIPTION:
 This script updates the version of EMMREM to the latest one from the
 SVN repository. Use this ONLY if you have SUBVERSION installed.

USAGE:
 svnUpEMMREM.sh               <Update EMMREM to latest version.>
 svnUpEMMREM.sh --help        <Display this HELP screen only.>
```

```
================================================================================
```

PLOTTING COMMANDS

The following commands, divided in several categories, are used to create plots of the EPREM and BRYNTRN output.

```
============================================================
```

plotAccuDose.sh

```
DESCRIPTION:
A script for plotting event Dose and Dose Equivalent rate and
accumulated.


USAGE:
plotAccuDose.sh --help             <Display this help screen only.>
plotAccuDose.sh [DoseFilename] [obsFileName] [obsName]  <Run this script.>
(Example: plotAccuDose.sh outputDose.plot sepDataobs 'GOES Spacecraft')

plotAccuDose.sh -x 'MM/DD/YYYY HH:MM:SS' 'MM/DD/YYYY HH:MM:SS' \
[DoseFilename] [obsFileName] [obsName]
                              <Run the script with a specific time range.>
     The above is the complete format for the time range. This
     software accepts the short format 'MM/DD/YYYY' as well.
(Example: plotAccuDose.sh -x '05/02/1998' '05/13/1998 12:23 \
outputDose.plot sepDataobs 'GOES Spacecraft')

plotAccuDose.sh -c [listOfColumnsToPlot] \
[DoseFilename] [obsFileName] [obsName]
                              <plot different data columns>
(Example: plotAccuDose.sh -c [1,2,4,5] \
outputDose.plot sepDataobs 'GOES Spacecraft')

plotAccuDose.sh -t [xtimeLayerCode] [DoseFilename] [obsFileName] [obsName]
                              <plot different time units on the x-axis:>
                               1: HH:MM
                               2: MM/DD/YY
```

```
                                3: Both HH:MM and MM/DD/YY
(Example: plotAccuDose.sh -t 2 outputDose.plot sepDataobs 'GOES Spacecraft)



plotAccuDose.sh -a [solidAngleIntegrationFactor]
                              <set the solid angle, in PI radians, over
                               which to integrate the dose. Usually, for
                               spacecraft it is 4, for planets 2.>

ADDITIONAL OPTIONAL PARAMETERS:

--withgcr [year] [radius(AU)]
    <Add GCR dose/s (and dose equivalent/s) to the SEP dose/s
     (and dose equivalent/s) and create additional plots of
     normalized  GCR (GCR/(GCR+SEP)) dose and dose equivalent rates.>

--skinlimit  <Overplot the 30-day skin limit for astronauts.>

--bfolimit  <Overplot the 30-day Blood Forming Organ limit.>

--limits   <Overplot both the skin and BFO 30-day limits.>


The above options can be combined, with the -x option preceding the -c one:
plotAccuDose.sh -x '05/02/1998' '05/13/1998 12:23' \
 -c [2,3,5] -t 2 --withgcr 1998 0.99 outputDose.plot sepDataobs 'GOES Spacecraft'


OUTPUT:
There is a maximum of 4 files that are produced from running
this script:
[obsFileName]AccuDose.eps  <Plot of Dose rate for different depths.>
[obsFileName]AccuDoseEq.eps  <Plot of Dose Equivalent rate for
                              different shielding depths.>

When the --withgcr option is used, two more files are produced:
[obsFileName]GcrAccuDose.eps  <Plot of fractional GCR dose rate
                              for different shielding depths.>
[obsFileName]GcrAccuDoseEq.eps  <Plot of fractional GCR dose equivalent
                              rate for different shielding depths.>
============================================================

============================================================
```

plotDose.sh

```
DESCRIPTION:
A script for plotting event Dose.


USAGE:
plotDose.sh --help             <Display this help screen only.>
plotDose.sh [DoseFilename] [obsFileName] [obsName]  <Run this script.>
(Example: plotDose.sh outputDose.plot sepDataobs 'GOES Spacecraft')

plotDose.sh -x 'MM/DD/YYYY HH:MM:SS' 'MM/DD/YYYY HH:MM:SS' \
[DoseFilename] [obsFileName] [obsName]
                              <Run the script with a specific time range.>
```

```
     The above is the complete format for the time range. This
     software accepts the short format 'MM/DD/YYYY' as well.
(Example: plotDose.sh -x '05/02/1998' '05/13/1998 12:23 \
outputDose.plot sepDataobs 'GOES Spacecraft')

plotDose.sh -c [listOfColumnsToPlot] \
[DoseFilename] [obsFileName] [obsName]
                              <plot different data columns>
(Example: plotDose.sh -c [1,2,4,5] \
outputDose.plot sepDataobs 'GOES Spacecraft')

plotDose.sh -t [xtimeLayerCode] [DoseFilename] [obsFileName] [obsName]
                             <plot different time units on the x-axis:>
                              1: HH:MM
                              2: MM/DD/YY
                              3: Both HH:MM and MM/DD/YY
(Example: plotDose.sh -t 2 outputDose.plot sepDataobs 'GOES Spacecraft)


plotDose.sh -a [solidAngleIntegrationFactor]
                            <set the solid angle, in PI radians, over
                             which to integrate the dose. Usually, for
                             spacecraft it is 4, for planets 2.>


plotDose.sh --withgcr [year] [radius(AU)] [DoseFilename] [obsFileName] [obsName]
    <Add GCR dose/s (and dose equivalent/s) to the SEP dose/s
     (and dose equivalent/s) and create additional plots of
     normalized  GCR (GCR/(GCR+SEP)) dose and dose equivalent rates.>

The above options can be combined, with the -x option preceding the -c one:
plotDose.sh -x '05/02/1998' '05/13/1998 12:23' \
 -c [2,3,5] -t 2 --withgcr 1998 0.99 earthDataOutput.plot earthData 'earth
Spacecraft'


OUTPUT:
There is a maximum of 4 files that are produced from running
this script:
[obsFileName]Dose.eps  <Plot of Dose rate for different depths.>
[obsFileName]DoseEq.eps  <Plot of Dose Equivalent rate for
                             different shielding depths.>

When the --withgcr option is used, two more files are produced:
[obsFileName]GcrDose.eps  <Plot of fractional GCR dose rate
                             for different shielding depths.>
[obsFileName]GcrDoseEq.eps  <Plot of fractional GCR dose equivalent
                             rate for different shielding depths.>
============================================================

============================================================
```

plotEarthDataAccuDose.sh

```
DESCRIPTION:
A script for plotting event Dose and Dose Equivalent rate and
accumulated.
```

```
USAGE:
plotEarthDataAccuDose.sh --help          <Display this help screen only.>
plotEarthDataAccuDose.sh                 <Run this script.>

plotEarthDataAccuDose.sh -x 'MM/DD/YYYY HH:MM:SS' 'MM/DD/YYYY HH:MM:SS'
                   <Run the script with a specific time range.>
     The above is the complete format for the time range. This
     software accepts the short format 'MM/DD/YYYY' as well.
(Example: plotEarthDataAccuDose.sh -x '05/02/1998' '05/13/1998 12:23')

plotEarthDataAccuDose.sh -c [listOfColumnsToPlot]
                   <plot different data columns>
(Example: plotEarthDataAccuDose.sh -c [1,2,4,5])

plotEarthDataAccuDose.sh -t [xtimeLayerCode]
                   <plot different time units on the x-axis:>
                    1: HH:MM
                    2: MM/DD/YY
                    3: Both HH:MM and MM/DD/YY
(Example: plotEarthDataAccuDose.sh -t 2)

plotEarthDataAccuDose.sh --withgcr [year] [radius(AU)]
    <Add GCR dose/s (and dose equivalent/s) to the SEP dose/s
     (and dose equivalent/s) and create additional plots of
     normalized  GCR (GCR/(GCR+SEP)) dose and dose equivalent rates.>


The above options can be combined, with the -x option preceding
the -c one, and -c the -t one:
plotEarthDataAccuDose.sh -x '05/02/1998' '05/13/1998 12:23' -c [2,3,5] -t 2 --
withgcr 1998 0.99


OUTPUT:
There is a maximum of 4 files that are produced from running
this script:
sepDataobsAccuDose.eps  <Plot of Dose rate for different depths.>
sepDataobsAccuDoseEq.eps  <Plot of Dose Equivalent rate for
                             different shielding depths.>

When the --withgcr option is used, two more files are produced:
sepDataobsGcrAccuDose.eps  <Plot of fractional GCR dose rate
                             for different shielding depths.>
sepDataobsGcrAccuDoseEq.eps  <Plot of fractional GCR dose equivalent
                             rate for different shielding depths.>
============================================================

============================================================
```

plotEarthDataDose.sh

```
DESCRIPTION:
A script for plotting event Dose and Dose Equivalent
at EMMREM/GOES Spacecraft.


INPUT:
The input to this script is the result of running BRYNTRN for
the GOES Spacecraft. This script requires that the input file
```

```
'sepDataOutput.plot' be present in the folder where the script
is executed!


USAGE:
plotEarthDataDose.sh --help    <Display this help screen only.>
plotEarthDataDose.sh           <Run this script.>
plotEarthDataDose.sh --withgcr [year] [radius(AU)]
    <Add GCR dose/s (and dose equivalent/s) to the SEP dose/s
     (and dose equivalent/s) and create additional plots of
     fractional GCR(GCR/(GCR+SEP)) dose and dose equivalent rates.>

plotEarthDataDose.sh -x 'MM/DD/YYYY HH:MM:SS' 'MM/DD/YYYY HH:MM:SS'
                   <Run the script with a specific time range.>
     The above is the complete format for the time range. This
     software accepts the short format 'MM/DD/YYYY' as well.
(Example: plotEarthDataDose.sh -x '05/02/1998' '05/13/1998 12:23')

plotEarthDataDose.sh -c [listOfColumnsToPlot]
                   <plot different data columns>
(Example: plotEarthDataDose.sh -c [1,2,4,5])

plotEarthDataDose.sh -t [xtimeLayerCode]
                   <plot different time units on the x-axis:>
                    1: HH:MM
                    2: MM/DD/YY
                    3: Both HH:MM and MM/DD/YY
(Example: plotEarthDataDose.sh -t 2)

The above options can be combined, with the -x option preceding
the -c one, and -c the -t one:
plotEarthDataDose.sh -x '05/02/1998' '05/13/1998 12:23' -c [2,3,5] -t 2 --withgcr
1998 0.99


OUTPUT:
There is a maximum of 4 files that are produced from running
this script:
sepDataDose.eps  <Plot of Dose rate for different depths.>
sepDataDoseEq.eps  <Plot of Dose Equivalent rate for
                            different shielding depths.>

When the --withgcr option is used, two more files are produced:
sepDataGcrDose.eps  <Plot of fractional GCR dose rate
                            for different shielding depths.>
sepDataGcrDoseEq.eps  <Plot of fractional GCR dose equivalent
                            rate for different shielding depths.>
============================================================

============================================================
```

plotEarthDataFlux.sh

```
DESCRIPTION:
A script for plotting output particle flux from one source.


USAGE:
plotEarthDataFlux.sh --help                    <Display this help screen only.>
```

```
plotEarthDataFlux.sh                            <Run this script.>

plotEarthDataFlux.sh -x 'MM/DD/YYYY HH:MM:SS' 'MM/DD/YYYY HH:MM:SS'
                         <Run the script with a specific time range.>
     The above is the complete format for the time range. This
     software accepts the short format 'MM/DD/YYYY' as well.
(Example: plotEarthDataFlux.sh -x '05/02/1998' '05/13/1998 12:23')

plotEarthDataFlux.sh -c [listOfColumnsToPlot]  <plot different data columns>
(Example: plotEarthDataFlux.sh -c [1,2,4,5])

plotEarthDataFlux.sh -t [xtimeLayerCode] <plot different time units on the x-axis:>
                               1: HH:MM
                               2: MM/DD/YY
                               3: Both HH:MM and MM/DD/YY
(Example: plotEarthDataFlux.sh -t 2)

The above options can be combined, with the -x option preceding the -c one:
plotEarthDataFlux.sh -x '05/02/1998' '05/13/1998 12:23' -c [2,3,5] -t 2
============================================================
```

```
============================================================
```

plotEarthFlux.sh

```
DESCRIPTION:
A script for comparing EPREM Earth observer output particle
flux with flux from the GOES spacecraft.


USAGE:
plotEarthFlux.sh --help              <Display this help screen only.>
plotEarthFlux.sh                     <Run this script.>
Example:

plotEarthFlux.sh -x 'MM/DD/YYYY HH:MM:SS' 'MM/DD/YYYY HH:MM:SS'
                            <Run the script with a specific time range.>
     The above is the complete format for the time range. This
     software accepts the short format 'MM/DD/YYYY' as well.
(Example: plotEarthFlux.sh -x '05/02/1998' '05/13/1998 12:23')

plotEarthFlux.sh -c [listOfColumnsToPlot]  <plot different data columns>
(Example: plotEarthFlux.sh -c [1,2,4,5])

plotEarthFlux.sh -t [xtimeLayerCode]  <plot different time units on the x-axis:>
                               1: HH:MM
                               2: MM/DD/YY
                               3: Both HH:MM and MM/DD/YY
(Example: plotEarthFlux.sh -t 2)

The above options can be combined, with the -x option preceding the -c one,
and the -c the -t one:
plotEarthFlux.sh -x '05/02/1998' '05/13/1998 12:23' -c [2,3,5] -t 2
============================================================
```

```
============================================================
```

plotEarthObsAccuDose.sh

```
DESCRIPTION:
A script for plotting event Dose and Dose Equivalent rate and
accumulated.


USAGE:
plotEarthObsAccuDose.sh --help          <Display this help screen only.>
plotEarthObsAccuDose.sh                 <Run this script.>

plotEarthObsAccuDose.sh -x 'MM/DD/YYYY HH:MM:SS' 'MM/DD/YYYY HH:MM:SS'
                  <Run the script with a specific time range.>
     The above is the complete format for the time range. This
     software accepts the short format 'MM/DD/YYYY' as well.
(Example: plotEarthObsAccuDose.sh -x '05/02/1998' '05/13/1998 12:23')

plotEarthObsAccuDose.sh -c [listOfColumnsToPlot]
                  <plot different data columns>
(Example: plotEarthObsAccuDose.sh -c [1,2,4,5])

plotEarthObsAccuDose.sh -t [xtimeLayerCode]
                  <plot different time units on the x-axis:>
                   1: HH:MM
                   2: MM/DD/YY
                   3: Both HH:MM and MM/DD/YY
(Example: plotEarthObsAccuDose.sh -t 2)

plotEarthObsAccuDose.sh --withgcr [year] [radius(AU)]
    <Add GCR dose/s (and dose equivalent/s) to the SEP dose/s
     (and dose equivalent/s) and create additional plots of
     normalized  GCR (GCR/(GCR+SEP)) dose and dose equivalent rates.>


The above options can be combined, with the -x option preceding
the -c one, and -c the -t one:
plotEarthObsAccuDose.sh -x '05/02/1998' '05/13/1998 12:23' -c [2,3,5] -t 2 --withgcr
1998 0.99


OUTPUT:
There is a maximum of 4 files that are produced from running
this script:
earthobsAccuDose.eps  <Plot of Dose rate for different depths.>
earthobsAccuDoseEq.eps  <Plot of Dose Equivalent rate for
                           different shielding depths.>

When the --withgcr option is used, two more files are produced:
earthobsGcrAccuDose.eps  <Plot of fractional GCR dose rate
                           for different shielding depths.>
earthobsGcrAccuDoseEq.eps  <Plot of fractional GCR dose equivalent
                           rate for different shielding depths.>
============================================================

============================================================
```

plotEarthObsDose.sh

```
DESCRIPTION:
A script for plotting event Dose and Dose Equivalent
at EMMREM/Earth Observer.
```

```
INPUT:
The input to this script is the result of running BRYNTRN for
the Earth Observer. This script requires that the input file
'earthOutput.plot' be present in the folder where the script
is executed!


USAGE:
plotEarthObsDose.sh --help    <Display this help screen only.>
plotEarthObsDose.sh           <Run this script.>

plotEarthObsDose.sh -x 'MM/DD/YYYY HH:MM:SS' 'MM/DD/YYYY HH:MM:SS'
                    <Run the script with a specific time range.>
     The above is the complete format for the time range. This
     software accepts the short format 'MM/DD/YYYY' as well.
(Example: plotEarthObsDose.sh -x '05/02/1998' '05/13/1998 12:23')

plotEarthObsDose.sh -c [listOfColumnsToPlot]
                    <plot different data columns>
(Example: plotEarthObsDose.sh -c [1,2,4,5])

plotEarthObsDose.sh -t [xtimeLayerCode]
                    <plot different time units on the x-axis:>
                     1: HH:MM
                     2: MM/DD/YY
                     3: Both HH:MM and MM/DD/YY
(Example: plotEarthObsDose.sh -t 2)

plotEarthObsDose.sh --withgcr [year] [radius(AU)]
    <Add GCR dose/s (and dose equivalent/s) to the SEP dose/s
     (and dose equivalent/s) and create additional plots of
     normalized  GCR (GCR/(GCR+SEP)) dose and dose equivalent rates.>

The above options can be combined, with the -x option preceding
the -c one, and so on:
plotEarthObsDose.sh -x '05/02/1998' '05/13/1998 12:23' -c [2,3,5] -t 2 --withgcr
1998 0.99


OUTPUT:
There is a maximum of 4 files that are produced from running
this script:
earthobsDose.eps  <Plot of Dose rate for different depths.>
earthobsDoseEq.eps  <Plot of Dose Equivalent rate for
                             different shielding depths.>

When the --withgcr option is used, two more files are produced:
earthobsGcrDose.eps  <Plot of fractional GCR dose rate
                             for different shielding depths.>
earthobsGcrDoseEq.eps  <Plot of fractional GCR dose equivalent
                             rate for different shielding depths.>
============================================================

============================================================
```

plotEarthObsFlux.sh

```
DESCRIPTION:
A script for plotting output particle flux from one source.


USAGE:
plotEarthObsFlux.sh --help                    <Display this help screen only.>
plotEarthObsFlux.sh [fluxFilename] [obsFileName] [obsName]  <Run this script.>
(Example: plotEarthObsFlux.sh sepDatafluxes.txt sepDataobs 'GOES Spacecraft')

plotEarthObsFlux.sh -x 'MM/DD/YYYY HH:MM:SS' 'MM/DD/YYYY HH:MM:SS'
                              <Run the script with a specific time range.>
     The above is the complete format for the time range. This
     software accepts the short format 'MM/DD/YYYY' as well.
(Example: plotEarthObsFlux.sh -x '05/02/1998' '05/13/1998 12:23')

plotEarthObsFlux.sh -c [listOfColumnsToPlot]
                              <plot different data columns>
(Example: plotEarthObsFlux.sh -c [1,2,4,5])

plotEarthObsFlux.sh -t [xtimeLayerCode] <plot different time units on the x-axis:>
                               1: HH:MM
                               2: MM/DD/YY
                               3: Both HH:MM and MM/DD/YY
(Example: plotEarthObsFlux.sh -t 2)

The above options can be combined, with the -x option preceding the -c one:
plotEarthObsFlux.sh -x '05/02/1998' '05/13/1998 12:23' -c [2,3,5] -t 2
============================================================

============================================================
```

plotFluxOne.sh

```
DESCRIPTION:
A script for plotting output particle flux from one source.


USAGE:
plotFluxOne.sh --help                      <Display this help screen only.>
plotFluxOne.sh [fluxFilename] [obsFileName] [obsName]  <Run this script.>
(Example: plotFluxOne.sh sepDatafluxes.txt sepDataobs 'GOES Spacecraft')

plotFluxOne.sh -x 'MM/DD/YYYY HH:MM:SS' 'MM/DD/YYYY HH:MM:SS' \
[fluxFilename] [obsFileName] [obsName]
                              <Run the script with a specific time range.>
     The above is the complete format for the time range. This
     software accepts the short format 'MM/DD/YYYY' as well.
(Example: plotFluxOne.sh -x '05/02/1998' '05/13/1998 12:23')

plotFluxOne.sh -c [listOfColumnsToPlot] \
[fluxFilename] [obsFileName] [obsName]
                              <plot different data columns>
(Example: plotFluxOne.sh -c [1,2,4,5] \
sepDatafluxes.txt sepDataobs 'GOES Spacecraft')

plotFluxOne.sh -t [xtimeLayerCode] [fluxFilename] [obsFileName] [obsName]
                              <plot different time units on the x-axis:>
                               1: HH:MM
                               2: MM/DD/YY
```

```
                                3: Both HH:MM and MM/DD/YY
(Example: plotFluxOne.sh -t 2 sepDatafluxes.txt sepDataobs 'GOES Spacecraft)

The above options can be combined, with the -x option preceding the -c one:
plotFluxOne.sh -x '05/02/1998' '05/13/1998 12:23' \
 -c [2,3,5] -t 2 sepDatafluxes.txt sepDataobs 'GOES Spacecraft'
============================================================

============================================================
```

plotFluxTwo.sh

```
DESCRIPTION:
A script for comparing EPREM Earth observer output particle
flux with flux from the GOES spacecraft.


USAGE:
plotFluxTwo.sh --help                       <Display this help screen only.>
plotFluxTwo.sh [fluxFile1] [fluxFile2] [obsFile1] [obsFile2] [obsName1] [obsName2]
                                     <Run this script.>
Example:
plotFluxTwo.sh efluxes.txt sepDatafluxes.txt earthobs sepDataobs  \
'Earth Observer' 'GOES Spacecraft'

plotFluxTwo.sh -x 'MM/DD/YYYY HH:MM:SS' 'MM/DD/YYYY HH:MM:SS' \
[fluxFile1] [fluxFile2] [obsFile1] [obsFile2] [obsName1] [obsName2]
                             <Run the script with a specific time range.>
     The above is the complete format for the time range. This
     software accepts the short format 'MM/DD/YYYY' as well.
(Example: plotFluxTwo.sh -x '05/02/1998' '05/13/1998 12:23' \
efluxes.txt sepDatafluxes.txt earthobs sepDataobs \
'Earth Observer' 'GOES Spacecraft')

plotFluxTwo.sh -c [listOfColumnsToPlot] \
[fluxFile1] [fluxFile2] [obsFile1] [obsFile2] [obsName1] [obsName2]
                             <plot different data columns>
(Example: plotFluxTwo.sh -c [1,2,4,5] \
efluxes.txt sepDatafluxes.txt earthobs sepDataobs \
'Earth Observer' 'GOES Spacecraft')

plotFluxTwo.sh -t [xtimeLayerCode] \
[fluxFile1] [fluxFile2] [obsFile1] [obsFile2] [obsName1] [obsName2]
                             <plot different time units on the x-axis:>
                              1: HH:MM
                              2: MM/DD/YY
                              3: Both HH:MM and MM/DD/YY
(Example: plotFluxTwo.sh -t 2 \
efluxes.txt sepDatafluxes.txt earthobs sepDataobs \
'Earth Observer' 'GOES Spacecraft')

The above options can be combined, with the -x option preceding the -c one:
plotFluxTwo.sh -x '05/02/1998' '05/13/1998 12:23' \
 -c [2,3,5] -t 2 efluxes.txt sepDatafluxes.txt earthobs sepDataobs \
'Earth Observer' 'GOES Spacecraft'
============================================================

============================================================
```

plotMarsObsAccuDose.sh

```
DESCRIPTION:
A script for plotting event Dose and Dose Equivalent rate and
accumulated.


USAGE:
plotMarsObsAccuDose.sh --help          <Display this help screen only.>
plotMarsObsAccuDose.sh                 <Run this script.>

plotMarsObsAccuDose.sh -x 'MM/DD/YYYY HH:MM:SS' 'MM/DD/YYYY HH:MM:SS'
                    <Run the script with a specific time range.>
     The above is the complete format for the time range. This
     software accepts the short format 'MM/DD/YYYY' as well.
(Example: plotMarsObsAccuDose.sh -x '05/02/1998' '05/13/1998 12:23')

plotMarsObsAccuDose.sh -c [listOfColumnsToPlot]
                    <plot different data columns>
(Example: plotMarsObsAccuDose.sh -c [1,2,4,5])

plotMarsObsAccuDose.sh -t [xtimeLayerCode]
                    <plot different time units on the x-axis:>
                     1: HH:MM
                     2: MM/DD/YY
                     3: Both HH:MM and MM/DD/YY
(Example: plotMarsObsAccuDose.sh -t 2)

plotMarsObsAccuDose.sh --withgcr [year] [radius(AU)]
    <Add GCR dose/s (and dose equivalent/s) to the SEP dose/s
     (and dose equivalent/s) and create additional plots of
     normalized  GCR (GCR/(GCR+SEP)) dose and dose equivalent rates.>


The above options can be combined, with the -x option preceding
the -c one, and -c the -t one:
plotMarsObsAccuDose.sh -x '05/02/1998' '05/13/1998 12:23' -c [2,3,5] -t 2 --withgcr
1998 0.99


OUTPUT:
There is a maximum of 4 files that are produced from running
this script:
marsobsAccuDose.eps  <Plot of Dose rate for different depths.>
marsobsAccuDoseEq.eps  <Plot of Dose Equivalent rate for
                            different shielding depths.>

When the --withgcr option is used, two more files are produced:
marsobsGcrAccuDose.eps  <Plot of fractional GCR dose rate
                            for different shielding depths.>
marsobsGcrAccuDoseEq.eps  <Plot of fractional GCR dose equivalent
                            rate for different shielding depths.>
============================================================

============================================================
```

plotMarsObsDose.sh

```
DESCRIPTION:
A script for plotting event Dose and Dose Equivalent
at EMMREM/Mars Observer.


INPUT:
The input to this script is the result of running BRYNTRN for
the Mars Observer. This script requires that the input file
'marsOutput.plot' be present in the folder where the script
is executed!


USAGE:
plotMarsObsDose.sh --help    <Display this help screen only.>
plotMarsObsDose.sh           <Run this script.>

plotMarsObsDose.sh -x 'MM/DD/YYYY HH:MM:SS' 'MM/DD/YYYY HH:MM:SS'
                   <Run the script with a specific time range.>
     The above is the complete format for the time range. This
     software accepts the short format 'MM/DD/YYYY' as well.
(Example: plotMarsObsDose.sh -x '05/02/1998' '05/13/1998 12:23')

plotMarsObsDose.sh -c [listOfColumnsToPlot]
                   <plot different data columns>
(Example: plotMarsObsDose.sh -c [1,2,4,5])

plotMarsObsDose.sh -t [xtimeLayerCode]
                   <plot different time units on the x-axis:>
                    1: HH:MM
                    2: MM/DD/YY
                    3: Both HH:MM and MM/DD/YY
(Example: plotMarsObsDose.sh -t 2)

plotMarsObsDose.sh --withgcr [year] [radius(AU)]
    <Add GCR dose/s (and dose equivalent/s) to the SEP dose/s
     (and dose equivalent/s) and create additional plots of
     normalized  GCR (GCR/(GCR+SEP)) dose and dose equivalent rates.>

The above options can be combined, with the -x option preceding
the -c one, and so on:
plotMarsObsDose.sh -x '05/02/1998' '05/13/1998 12:23' -c [2,3,5] -t 2 --withgcr 1998
0.99


OUTPUT:
There is a maximum of 4 files that are produced from running
this script:
marsobsDose.eps  <Plot of Dose rate for different depths.>
marsobsDoseEq.eps  <Plot of Dose Equivalent rate for
                            different shielding depths.>

When the --withgcr option is used, two more files are produced:
marsobsGcrDose.eps  <Plot of fractional GCR dose rate
                            for different shielding depths.>
marsobsGcrDoseEq.eps  <Plot of fractional GCR dose equivalent
                            rate for different shielding depths.>
============================================================
```

```
============================================================
```

plotMarsObsFlux.sh

```
DESCRIPTION:
A script for plotting output particle flux from one source.


USAGE:
plotMarsObsFlux.sh --help                    <Display this help screen only.>
plotMarsObsFlux.sh                           <Run this script.>

plotMarsObsFlux.sh -x 'MM/DD/YYYY HH:MM:SS' 'MM/DD/YYYY HH:MM:SS'
                           <Run the script with a specific time range.>
     The above is the complete format for the time range. This
     software accepts the short format 'MM/DD/YYYY' as well.
(Example: plotMarsObsFlux.sh -x '05/02/1998' '05/13/1998 12:23')

plotMarsObsFlux.sh -c [listOfColumnsToPlot]    <plot different data columns>
(Example: plotMarsObsFlux.sh -c [1,2,4,5])

plotMarsObsFlux.sh -t [xtimeLayerCode]  <plot different time units on the x-axis:>
                                1: HH:MM
                                2: MM/DD/YY
                                3: Both HH:MM and MM/DD/YY
(Example: plotMarsObsFlux.sh -t 2)

The above options can be combined, with the -x option preceding the -c one:
plotMarsObsFlux.sh -x '05/02/1998' '05/13/1998 12:23' -c [2,3,5] -t 2
============================================================

============================================================
```

plotMarsTopBottomAccuDose.sh

```
DESCRIPTION:
A script for plotting event Dose and Dose Equivalent
at EMMREM/Mars Atmosphere.

This script requires that the file marsOutputTopBottom.plot be present
in the folder where the script is executed!


USAGE:
plotMarsTopBottomAccuDose.sh --help    <Display this help screen only.>
plotMarsTopBottomAccuDose.sh           <Run this script.>

plotMarsTopBottomAccuDose.sh -x 'MM/DD/YYYY HH:MM:SS' 'MM/DD/YYYY HH:MM:SS'
                   <Run the script with a specific time range.>
     The above is the complete format for the time range. This
     software accepts the short format 'MM/DD/YYYY' as well.
(Example: plotMarsTopBottomAccuDose.sh -x '05/02/1998' '05/13/1998 12:23')

plotMarsTopBottomAccuDose.sh -c [listOfColumnsToPlot]
                   <plot different data columns>
(Example: plotMarsTopBottomAccuDose.sh -c [1,2,4,5])

plotMarsTopBottomAccuDose.sh -t [xtimeLayerCode]
                   <plot different time units on the x-axis:>
```

```
                    1: HH:MM
                    2: MM/DD/YY
                    3: Both HH:MM and MM/DD/YY
(Example: plotMarsTopBottomAccuDose.sh -t 2)

The above options can be combined, with the -x option preceding
the -c one, and -c the -t one:
plotMarsTopBottomAccuDose.sh -x '05/02/1998' '05/13/1998 12:23' -c [2,3,5] -t 2
============================================================

============================================================
```

plotMarsTopBottomDose.sh

```
DESCRIPTION:
A script for plotting event Dose and Dose Equivalent
at EMMREM/Mars Atmosphere.

This script requires that the file marsOutputTopBottom.plot be present
in the folder where the script is executed!


USAGE:
plotMarsTopBottomDose.sh --help    <Display this help screen only.>
plotMarsTopBottomDose.sh           <Run this script.>

plotMarsTopBottomDose.sh -x 'MM/DD/YYYY HH:MM:SS' 'MM/DD/YYYY HH:MM:SS'
                   <Run the script with a specific time range.>
     The above is the complete format for the time range. This
     software accepts the short format 'MM/DD/YYYY' as well.
(Example: plotMarsTopBottomDose.sh -x '05/02/1998' '05/13/1998 12:23')

plotMarsTopBottomDose.sh -c [listOfColumnsToPlot]
                   <plot different data columns>
(Example: plotMarsTopBottomDose.sh -c [1,2,4,5])

plotMarsTopBottomDose.sh -t [xtimeLayerCode]
                   <plot different time units on the x-axis:>
                    1: HH:MM
                    2: MM/DD/YY
                    3: Both HH:MM and MM/DD/YY
(Example: plotMarsTopBottomDose.sh -t 2)

The above options can be combined, with the -x option preceding
the -c one, and -c the -t one:
plotMarsTopBottomDose.sh -x '05/02/1998' '05/13/1998 12:23' -c [2,3,5] -t 2
============================================================

============================================================
```

plotMoonObsAccuDose.sh

```
DESCRIPTION:
A script for plotting event Dose and Dose Equivalent rate and
accumulated.


USAGE:
plotMoonObsAccuDose.sh --help          <Display this help screen only.>
```

```
plotMoonObsAccuDose.sh                  <Run this script.>

plotMoonObsAccuDose.sh -x 'MM/DD/YYYY HH:MM:SS' 'MM/DD/YYYY HH:MM:SS'
                   <Run the script with a specific time range.>
     The above is the complete format for the time range. This
     software accepts the short format 'MM/DD/YYYY' as well.
(Example: plotMoonObsAccuDose.sh -x '05/02/1998' '05/13/1998 12:23')

plotMoonObsAccuDose.sh -c [listOfColumnsToPlot]
                   <plot different data columns>
(Example: plotMoonObsAccuDose.sh -c [1,2,4,5])

plotMoonObsAccuDose.sh -t [xtimeLayerCode]
                   <plot different time units on the x-axis:>
                    1: HH:MM
                    2: MM/DD/YY
                    3: Both HH:MM and MM/DD/YY
(Example: plotMoonObsAccuDose.sh -t 2)

plotMoonObsAccuDose.sh --withgcr [year] [radius(AU)]
    <Add GCR dose/s (and dose equivalent/s) to the SEP dose/s
     (and dose equivalent/s) and create additional plots of
     normalized  GCR (GCR/(GCR+SEP)) dose and dose equivalent rates.>


The above options can be combined, with the -x option preceding
the -c one, and -c the -t one:
plotMoonObsAccuDose.sh -x '05/02/1998' '05/13/1998 12:23' -c [2,3,5] -t 2 --withgcr
1998 0.99


OUTPUT:
There is a maximum of 4 files that are produced from running
this script:
moonobsAccuDose.eps  <Plot of Dose rate for different depths.>
moonobsAccuDoseEq.eps  <Plot of Dose Equivalent rate for
                           different shielding depths.>

When the --withgcr option is used, two more files are produced:
moonobsGcrAccuDose.eps  <Plot of fractional GCR dose rate
                           for different shielding depths.>
moonobsGcrAccuDoseEq.eps  <Plot of fractional GCR dose equivalent
                           rate for different shielding depths.>
============================================================

============================================================
```

plotMoonObsDose.sh

```
DESCRIPTION:
A script for plotting event Dose and Dose Equivalent
at EMMREM/Moon Observer.


INPUT:
The input to this script is the result of running BRYNTRN for
the Moon Observer. This script requires that the input file
'moonOutput.plot' be present in the folder where the script
is executed!
```

```
USAGE:
plotMoonObsDose.sh --help    <Display this help screen only.>
plotMoonObsDose.sh           <Run this script.>

plotMoonObsDose.sh -x 'MM/DD/YYYY HH:MM:SS' 'MM/DD/YYYY HH:MM:SS'
                   <Run the script with a specific time range.>
     The above is the complete format for the time range. This
     software accepts the short format 'MM/DD/YYYY' as well.
(Example: plotMoonObsDose.sh -x '05/02/1998' '05/13/1998 12:23')

plotMoonObsDose.sh -c [listOfColumnsToPlot]
                   <plot different data columns>
(Example: plotMoonObsDose.sh -c [1,2,4,5])

plotMoonObsDose.sh -t [xtimeLayerCode]
                   <plot different time units on the x-axis:>
                    1: HH:MM
                    2: MM/DD/YY
                    3: Both HH:MM and MM/DD/YY
(Example: plotMoonObsDose.sh -t 2)

plotMoonObsDose.sh --withgcr [year] [radius(AU)]
    <Add GCR dose/s (and dose equivalent/s) to the SEP dose/s
     (and dose equivalent/s) and create additional plots of
     normalized  GCR (GCR/(GCR+SEP)) dose and dose equivalent rates.>

The above options can be combined, with the -x option preceding
the -c one, and -c the -t one:
plotMoonObsDose.sh -x '05/02/1998' '05/13/1998 12:23' -c [2,3,5] -t 2 --withgcr 1998
0.99


OUTPUT:
There is a maximum of 4 files that are produced from running
this script:
moonobsDose.eps  <Plot of Dose rate for different depths.>
moonobsDoseEq.eps  <Plot of Dose Equivalent rate for
                            different shielding depths.>

When the --withgcr option is used, two more files are produced:
moonobsGcrDose.eps  <Plot of fractional GCR dose rate
                            for different shielding depths.>
moonobsGcrDoseEq.eps  <Plot of fractional GCR dose equivalent
                            rate for different shielding depths.>
============================================================

============================================================
```

plotMoonObsFlux.sh

```
DESCRIPTION:
A script for plotting output particle flux from one source.


USAGE:
plotMoonObsFlux.sh --help                    <Display this help screen only.>
plotMoonObsFlux.sh                           <Run this script.>
```

```
plotMoonObsFlux.sh -x 'MM/DD/YYYY HH:MM:SS' 'MM/DD/YYYY HH:MM:SS'
                          <Run the script with a specific time range.>
     The above is the complete format for the time range. This
     software accepts the short format 'MM/DD/YYYY' as well.
(Example: plotMoonObsFlux.sh -x '05/02/1998' '05/13/1998 12:23')

plotMoonObsFlux.sh -c [listOfColumnsToPlot]  <plot different data columns>
(Example: plotMoonObsFlux.sh -c [1,2,4,5])

plotMoonObsFlux.sh -t [xtimeLayerCode]  <plot different time units on the x-axis:>
                               1: HH:MM
                               2: MM/DD/YY
                               3: Both HH:MM and MM/DD/YY
(Example: plotMoonObsFlux.sh -t 2 sepDatafluxes.txt sepDataobs 'GOES Spacecraft)

The above options can be combined, with the -x option preceding the -c one:
plotMoonObsFlux.sh -x '05/02/1998' '05/13/1998 12:23' -c [2,3,5] -t 2
============================================================

============================================================
```

plotUlyssesDataAccuDose.sh

```
DESCRIPTION:
A script for plotting event Dose and Dose Equivalent rate and
accumulated.


USAGE:
plotUlyssesDataAccuDose.sh --help          <Display this help screen only.>
plotUlyssesDataAccuDose.sh                 <Run this script.>

plotUlyssesDataAccuDose.sh -x 'MM/DD/YYYY HH:MM:SS' 'MM/DD/YYYY HH:MM:SS'
                    <Run the script with a specific time range.>
     The above is the complete format for the time range. This
     software accepts the short format 'MM/DD/YYYY' as well.
(Example: plotUlyssesDataAccuDose.sh -x '05/02/1998' '05/13/1998 12:23')

plotUlyssesDataAccuDose.sh -c [listOfColumnsToPlot]
                    <plot different data columns>
(Example: plotUlyssesDataAccuDose.sh -c [1,2,4,5])

plotUlyssesDataAccuDose.sh -t [xtimeLayerCode]
                    <plot different time units on the x-axis:>
                     1: HH:MM
                     2: MM/DD/YY
                     3: Both HH:MM and MM/DD/YY
(Example: plotUlyssesDataAccuDose.sh -t 2)

plotUlyssesDataAccuDose.sh --withgcr [year] [radius(AU)]
    <Add GCR dose/s (and dose equivalent/s) to the SEP dose/s
     (and dose equivalent/s) and create additional plots of
     normalized  GCR (GCR/(GCR+SEP)) dose and dose equivalent rates.>


The above options can be combined, with the -x option preceding
the -c one, and -c the -t one:
plotUlyssesDataAccuDose.sh -x '05/02/1998' '05/13/1998 12:23' -c [2,3,5] -t 2 --
```

```
withgcr 1998 0.99


OUTPUT:
There is a maximum of 4 files that are produced from running
this script:
ulyssesDataobsAccuDose.eps  <Plot of Dose rate for different depths.>
ulyssesDataobsAccuDoseEq.eps  <Plot of Dose Equivalent rate for
                             different shielding depths.>

When the --withgcr option is used, two more files are produced:
ulyssesDataobsGcrAccuDose.eps  <Plot of fractional GCR dose rate
                             for different shielding depths.>
ulyssesDataobsGcrAccuDoseEq.eps  <Plot of fractional GCR dose equivalent
                             rate for different shielding depths.>
============================================================

============================================================
```

plotUlyssesDataDose.sh

```
DESCRIPTION:
A script for plotting event Dose and Dose Equivalent
at EMMREM/Ulysses Spacecraft.


INPUT:
The input to this script is the result of running BRYNTRN for
the Ulysses Spacecraft. This script requires that the input file
'ulyssesDataOutput.plot' be present in the folder where the script
is executed!


USAGE:
plotUlyssesDataDose.sh --help    <Display this help screen only.>
plotUlyssesDataDose.sh           <Run this script.>

plotUlyssesDataDose.sh -x 'MM/DD/YYYY HH:MM:SS' 'MM/DD/YYYY HH:MM:SS'
                   <Run the script with a specific time range.>
     The above is the complete format for the time range. This
     software accepts the short format 'MM/DD/YYYY' as well.
(Example: plotUlyssesDataDose.sh -x '05/02/1998' '05/13/1998 12:23')

plotUlyssesDataDose.sh -c [listOfColumnsToPlot]
                   <plot different data columns>
(Example: plotUlyssesDataDose.sh -c [1,2,4,5])

plotUlyssesDataDose.sh -t [xtimeLayerCode]
                   <plot different time units on the x-axis:>
                    1: HH:MM
                    2: MM/DD/YY
                    3: Both HH:MM and MM/DD/YY
(Example: plotUlyssesDataDose.sh -t 2)

plotUlyssesDataDose.sh --withgcr [year] [radius(AU)]
    <Add GCR dose/s (and dose equivalent/s) to the SEP dose/s
     (and dose equivalent/s) and create additional plots of
     normalized  GCR (GCR/(GCR+SEP)) dose and dose equivalent rates.>
```

```
The above options can be combined, with the -x option preceding
the -c one, and -c the -t one:
plotUlyssesDataDose.sh -x '05/02/1998' '05/13/1998 12:23' -c [2,3,5] -t 2 --withgcr
1998 0.99


OUTPUT:
There is a maximum of 4 files that are produced from running
this script:
ulyssesDataDose.eps  <Plot of Dose rate for different depths.>
ulyssesDataDoseEq.eps  <Plot of Dose Equivalent rate for
                           different shielding depths.>

When the --withgcr option is used, two more files are produced:
ulyssesDataGcrDose.eps  <Plot of fractional GCR dose rate
                           for different shielding depths.>
ulyssesDataGcrDoseEq.eps  <Plot of fractional GCR dose equivalent
                           rate for different shielding depths.>
============================================================

============================================================
```

plotUlyssesDataFlux.sh

```
DESCRIPTION:
A script for plotting output particle flux from one source.


USAGE:
plotUlyssesDataFlux.sh --help                    <Display this help screen only.>
plotUlyssesDataFlux.sh                           <Run this script.>

plotUlyssesDataFlux.sh -x 'MM/DD/YYYY HH:MM:SS' 'MM/DD/YYYY HH:MM:SS'
                          <Run the script with a specific time range.>
     The above is the complete format for the time range. This
     software accepts the short format 'MM/DD/YYYY' as well.
(Example: plotUlyssesDataFlux.sh -x '05/02/1998' '05/13/1998 12:23')

plotUlyssesDataFlux.sh -c [listOfColumnsToPlot]   <plot different data columns>
(Example: plotUlyssesDataFlux.sh -c [1,2,4,5])

plotUlyssesDataFlux.sh -t [xtimeLayerCode]  <plot different time units on the x-
axis:>
                               1: HH:MM
                               2: MM/DD/YY
                               3: Both HH:MM and MM/DD/YY
(Example: plotUlyssesDataFlux.sh -t 2)

The above options can be combined, with the -x option preceding the -c one:
plotUlyssesDataFlux.sh -x '05/02/1998' '05/13/1998 12:23' -c [2,3,5] -t 2
============================================================

============================================================
```

plotUlyssesFlux.sh

```
DESCRIPTION:
A script for comparing EPREM Earth observer output particle
flux with flux from the GOES spacecraft.
```

```
USAGE:
plotUlyssesFlux.sh --help              <Display this help screen only.>
plotUlyssesFlux.sh                     <Run this script.>
Example:

plotUlyssesFlux.sh -x 'MM/DD/YYYY HH:MM:SS' 'MM/DD/YYYY HH:MM:SS'
                            <Run the script with a specific time range.>
     The above is the complete format for the time range. This
     software accepts the short format 'MM/DD/YYYY' as well.
(Example: plotUlyssesFlux.sh -x '05/02/1998' '05/13/1998 12:23')

plotUlyssesFlux.sh -c [listOfColumnsToPlot]  <plot different data columns>
(Example: plotUlyssesFlux.sh -c [1,2,4,5])

plotUlyssesFlux.sh -t [xtimeLayerCode]  <plot different time units on the x-axis:>
                              1: HH:MM
                              2: MM/DD/YY
                              3: Both HH:MM and MM/DD/YY
(Example: plotUlyssesFlux.sh -t 2)

The above options can be combined, with the -x option preceding the -c one,
and the -c the -t one:
plotUlyssesFlux.sh -x '05/02/1998' '05/13/1998 12:23' -c [2,3,5] -t 2
============================================================

============================================================
```

plotUlyssesObsAccuDose.sh

```
DESCRIPTION:
A script for plotting event Dose and Dose Equivalent rate and
accumulated.


USAGE:
plotUlyssesObsAccuDose.sh --help          <Display this help screen only.>
plotUlyssesObsAccuDose.sh                 <Run this script.>

plotUlyssesObsAccuDose.sh -x 'MM/DD/YYYY HH:MM:SS' 'MM/DD/YYYY HH:MM:SS'
                  <Run the script with a specific time range.>
     The above is the complete format for the time range. This
     software accepts the short format 'MM/DD/YYYY' as well.
(Example: plotUlyssesObsAccuDose.sh -x '05/02/1998' '05/13/1998 12:23')

plotUlyssesObsAccuDose.sh -c [listOfColumnsToPlot]
                  <plot different data columns>
(Example: plotUlyssesObsAccuDose.sh -c [1,2,4,5])

plotUlyssesObsAccuDose.sh -t [xtimeLayerCode]
                  <plot different time units on the x-axis:>
                   1: HH:MM
                   2: MM/DD/YY
                   3: Both HH:MM and MM/DD/YY
(Example: plotUlyssesObsAccuDose.sh -t 2)

plotUlyssesObsAccuDose.sh --withgcr [year] [radius(AU)]
    <Add GCR dose/s (and dose equivalent/s) to the SEP dose/s
```

```
     (and dose equivalent/s) and create additional plots of
     normalized  GCR (GCR/(GCR+SEP)) dose and dose equivalent rates.>


The above options can be combined, with the -x option preceding
the -c one, and -c the -t one:
plotUlyssesObsAccuDose.sh -x '05/02/1998' '05/13/1998 12:23' -c [2,3,5] -t 2 --
withgcr 1998 0.99


OUTPUT:
There is a maximum of 4 files that are produced from running
this script:
ulyssesobsAccuDose.eps  <Plot of Dose rate for different depths.>
ulyssesobsAccuDoseEq.eps  <Plot of Dose Equivalent rate for
                           different shielding depths.>

When the --withgcr option is used, two more files are produced:
ulyssesobsGcrAccuDose.eps  <Plot of fractional GCR dose rate
                           for different shielding depths.>
ulyssesobsGcrAccuDoseEq.eps  <Plot of fractional GCR dose equivalent
                           rate for different shielding depths.>
============================================================

============================================================
```

plotUlyssesObsDose.sh

```
DESCRIPTION:
A script for plotting event Dose and Dose Equivalent
at EMMREM/Ulysses Observer.


INPUT:
The input to this script is the result of running BRYNTRN for
the Ulysses Observer. This script requires that the input file
'ulyssesOutput.plot' be present in the folder where the script
is executed!


USAGE:
plotUlyssesObsDose.sh --help    <Display this help screen only.>
plotUlyssesObsDose.sh           <Run this script.>

plotUlyssesObsDose.sh -x 'MM/DD/YYYY HH:MM:SS' 'MM/DD/YYYY HH:MM:SS'
                   <Run the script with a specific time range.>
     The above is the complete format for the time range. This
     software accepts the short format 'MM/DD/YYYY' as well.
(Example: plotUlyssesObsDose.sh -x '05/02/1998' '05/13/1998 12:23')

plotUlyssesObsDose.sh -c [listOfColumnsToPlot]
                   <plot different data columns>
(Example: plotUlyssesObsDose.sh -c [1,2,4,5])

plotUlyssesObsDose.sh -t [xtimeLayerCode]
                   <plot different time units on the x-axis:>
                    1: HH:MM
                    2: MM/DD/YY
                    3: Both HH:MM and MM/DD/YY
```

```
(Example: plotUlyssesObsDose.sh -t 2)

plotUlyssesObsDose.sh --withgcr [year] [radius(AU)]
    <Add GCR dose/s (and dose equivalent/s) to the SEP dose/s
     (and dose equivalent/s) and create additional plots of
     normalized  GCR (GCR/(GCR+SEP)) dose and dose equivalent rates.>

The above options can be combined, with the -x option preceding
the -c one, and -c the -t one:
plotUlyssesObsDose.sh -x '05/02/1998' '05/13/1998 12:23' -c [2,3,5] -t 2 --withgcr
1998 0.99


OUTPUT:
There is a maximum of 4 files that are produced from running
this script:
ulyssesobsDose.eps  <Plot of Dose rate for different depths.>
ulyssesobsDoseEq.eps  <Plot of Dose Equivalent rate for
                           different shielding depths.>

When the --withgcr option is used, two more files are produced:
ulyssesobsGcrDose.eps  <Plot of fractional GCR dose rate
                           for different shielding depths.>
ulyssesobsGcrDoseEq.eps  <Plot of fractional GCR dose equivalent
                           rate for different shielding depths.>
============================================================

============================================================
```

plotUlyssesObsFlux.sh

```
DESCRIPTION:
A script for plotting output particle flux from one source.


USAGE:
plotUlyssesObsFlux.sh --help                    <Display this help screen only.>
plotUlyssesObsFlux.sh                           <Run this script.>

plotUlyssesObsFlux.sh -x 'MM/DD/YYYY HH:MM:SS' 'MM/DD/YYYY HH:MM:SS'
                          <Run the script with a specific time range.>
     The above is the complete format for the time range. This
     software accepts the short format 'MM/DD/YYYY' as well.
(Example: plotUlyssesObsFlux.sh -x '05/02/1998' '05/13/1998 12:23')

plotUlyssesObsFlux.sh -c [listOfColumnsToPlot]      <plot different data columns>
(Example: plotUlyssesObsFlux.sh -c [1,2,4,5])

plotUlyssesObsFlux.sh -t [xtimeLayerCode]   <plot different time units on the x-
axis:>
                               1: HH:MM
                               2: MM/DD/YY
                               3: Both HH:MM and MM/DD/YY
(Example: plotUlyssesObsFlux.sh -t 2)

The above options can be combined, with the -x option preceding the -c one:
plotUlyssesObsFlux.sh -x '05/02/1998' '05/13/1998 12:23' -c [2,3,5] -t 2
============================================================
```