ESTEC/Contract No. 12854/98/NL/NB
Space Environment Database (SEDAT)
WP302. Radiation environment analysis

Report on radiation environment analysis for the cruise phase of an interplanetary mission

RAL-SED-RP-0302
Issue 1.0, 3 January 2005

Author: Mike Hapgood, CLRC Rutherford Appleton Laboratory

ESA technical Officer: A Hilgers D/TOS, TOS-EMA

CCLRC
Rutherford Appleton Laboratory

Table of contents

1 Preface

1.1 Document change record

Issue	Date	Notes/remarks
0.5	10 Oct 2002	Partial draft released to report progress to and seek advice from ESA
0.6	25 Jan 2003	Add sections on trapped protons, trapped electrons, merging tool and application to the Bepi-Colombo trajectory supplied by ESA
1.0	03 Jan 2005	Update section 3 to address recent changes: • removal of strings from CALL_EXTERNAL interface • codes changes advised by Daniel Heynderickx • suppression of routine information messages Editorial changes throughout to clarify use of tools Add Annex B to list data codes Add Annex C to describe tools and Annex D to show compliance with test procedure

1.2 Purpose of the document

This document is the technical note reporting the results of the application of SEDAT to the radiation environment analysis for the cruise phase of an interplanetary mission task (WP302).

1.3 Definitions, acronyms and abbreviations

ESA	European Space Agency
GOES	Geosynchronous Orbiting Environment Satellite
GSE	Geocentric solar ecliptic
HAE	Heliospheric Aries ecliptic
HEE	Heliospheric Earth ecliptic
IDL	Interactive Data Language. Commercial product with good mathematical and graphics functionality used as the scripting language in SEDAT.
IMP	Interplanetary Monitoring Platform
ISEE	International Sun Earth Explorer
MeV	Mega electron-volt
NASA	National Aeronautics and Space Administration
NOAA	National Oceanic and Atmospheric Administration
RAL	Rutherford Appleton Laboratory
RSI	Research System Inc
SEDAT	Space Environment Database
SI	Système International, the international system of units.
TBC	To be confirmed
TBD	To be done

1.4 Important Documents

We list here the various documents used as source material for this report. These include both hardcopy and web sources. Documents may be referenced in the test and this is indicated by a sequential code of the form Xn, where n is an integer and X = A or R (for applicable and reference documents respectively). The series of integers are separate for applicable and reference documents.

1.4.1 Applicable documents

A1 SEDAT Statement of Work. Appendix 1 to AO/1-3306/97/NL/NB

A2 Space Environment Database and Analysis Tools. Proposal in response to ESA ITT AO/1-3306/97/NL/NB. RAL/RRS/201/97. January 1998.

1.4.2 Reference documents

R1 SEDAT report on solar proton model, RAL-SED-RP-0301

R2 Comments on Using SHIELDOSE-2, Readme file supplied with Shieldose-2

R3 Tables of Physical and Chemical Constants, G.W.C. Kaye and T.H. Laby, 14th edition, p. 279, 1979.

R4 SEDAT report on particle-induced background analysis, RAL-SED-RP-0304

R5 ISTP/IACG Global Attributes, http://spdf.gsfc.nasa.gov/istp_guide/gattributes.htm

R6 ISTP/IACG Variable Attributes, http://spdf.gsfc.nasa.gov/istp_guide/vattributes.htm

R7 http://spdf.gsfc.nasa.gov/istp_guide/variables.htm#Epoch

R8 An introduction to space physics coordinate systems, http://sspg1.bnsc.rl.ac.uk/Share/Coordinates/ct_home.htm

R9 Explanatory supplement to the astronomical almanac. Revised edition. 1992. Seidelmann P K ed.

R10 Reference Document for CSDS CDF Implementation, DS-QMW-TN-0003 http://www.space-plasma.qmw.ac.uk/DOC/DS-QMW-TN-0003.ps

R11 Feynman, J., Spitale,G., Wang, J. And Gabriel, S. (1993) Interplanetary proton fluence model: JPL 1991, *J.Geophys.Res*. **98**, 13281-13294.

R12 Feynman, J., Armstrong, T.P., Dao-Gibner, L. and Silverman, S. (1990) New interplanetary proton fluence model, *J. Spacecraft and Rockets* **27**, 403-410.

2 Introduction and structure

This document presents the results of SEDAT work package 302 which is a demonstration of the application of SEDAT to perform a radiation environment analysis for the cruise phase of an interplanetary missions. There are several aspects of this application and these are reflected in the subsequent sections of the document.

A key issue is the implementation of the Shieldose model as a SEDAT tool. This model is widely used to calculate the radiation doses in various materials due to energetic particle fluxes incident on various depths of aluminium shielding protecting those materials. Thus this tool, together with the energetic particle data in the SEDAT database, is the practical basis of this work package. The Shieldose model is available as a Fortran programme and had to be adapted slightly into to make it work within the SEDAT environment. Section 3 outlines that adaption with sufficient detail to demonstrate the major issues and show that the adaption is well-documented and properly verified. In addition, Section 4 discusses some important features of Shieldose that have been identified during the adaption and preserved in the SEDAT tool; it is important that users are aware of these features.

The next three sections (5, 6 and 7) describe the separate application of Shieldose to different particles types – first solar protons, then trapped protons and finally electrons. These sections outline any special processing needed to handle data on these different particles. In section 8 we describe a simple tool to merge and display the dose-depth data for these different particle types.

In section 9 we bring all this work together and show how SEDAT apply these applications to a particular mission scenario selected by ESA.

At the end of the report we have a series of annexes that provide details of the SEDAT data fields used to run Shieldose.

3 Use of the Shieldose model

3.1 Background

The main aim of WP302 is to generate the dose-depth curves expected in response to the energetic charged particle environment in the cruise phase of an interplanetary mission. Thus we need a SEDAT tool to generate dose-depth curves given energetic particle fluxes as input, e.g. as generated in SEDAT WP301 [R1]. To do this we have adapted Shieldose, the standard dose-depth program, to run as a SEDAT system tool. The more recent version 2 of Shieldose was used.

However, this was not a trival task given the constraints imposed by SEDAT requirements, in particular the lack of user access to the system level. Shieldose-2 is a standalone Fortran program which calculates dose-depth curves given the incident fluxes of energetic protons and electrons. It has four inputs:

- it prompts the user to enter the name of a control file via stdin,
- it then reads that control file to obtain parameters as described below,
- it reads a fixed data file which supplies parameters relevant to the calculation of proton-induced doses, and
- it reads the equivalent data file for electrons.

Shieldose-2 also outputs an extensive set of data to two files; outputs include reportings of control parameter values, of intermediate parameter values and of the dose-depth curves themselves.

Thus our aim was to modify Shieldose so that it could be invoked from inside SEDAT, i.e. from IDL. It was originally thought best to implement this using the IDL SPAWM command, i.e. to write an IDL script that would prepare the control file using particle flux spectra from SEDAT as an input, then invoke Shieldose as a standalone program and finally read the output files to ingest the dose-depth curves back into SEDAT. After further analysis it was decided to take an alternative approach in which Shieldose was invoked via the IDL CALL_EXTERNAL interface. This approach was selected as it offers more scope for tuning the efficiency of Shieldose-2 in the SEDAT environment. However, it requires us to convert Shieldose from a standalone program to a subroutine library. The rest of this section is concerned with the design, implementation and verification of that conversion.

We also note that examination of the Shieldose code showed many features of poor quality coding (e.g. widespread use of obsolescent features, poor file handling, limited use of comments, poor breakdown into reasonably size modules, convoluted logic). Thus this exercise was limited to the minimum changes need to make the code work as a subroutine and to generate test results consistent with reference data supplied with the source code.

3.2 Architecture

In this section we present an overview of the architecture used to implement Shieldose within SEDAT. A detailed description of the implementation, including the user interface, is given in subsequent sections. Figure 1 shows the key elements of the architecture. The main program is converted into a Fortran subroutine (shieldose_two.f) and the supporting subroutines are split out into separate files – as usual when using Fortran under Unix. It is called from SEDAT using an IDL script (shieldose_two.pro) via IDL's CALL_EXTERNAL interface; the script is implemented as a SEDAT system tool in order to allow this level of access to the operating system. In between the IDL script script and the main subroutine (as it now is), we have a Fortran wrapper routine (shieldose_idl.f), which converts input and output variables between the forms used by IDL and Fortran format. This uses some supporting subroutines to encapsulate machine-specific detail and to handle conversion of strings. The whole set of Fortran routines (main program, wrapper and all supporting subroutines) is compiled into a shared-object library as required by the CALL_EXTERNAL interface.

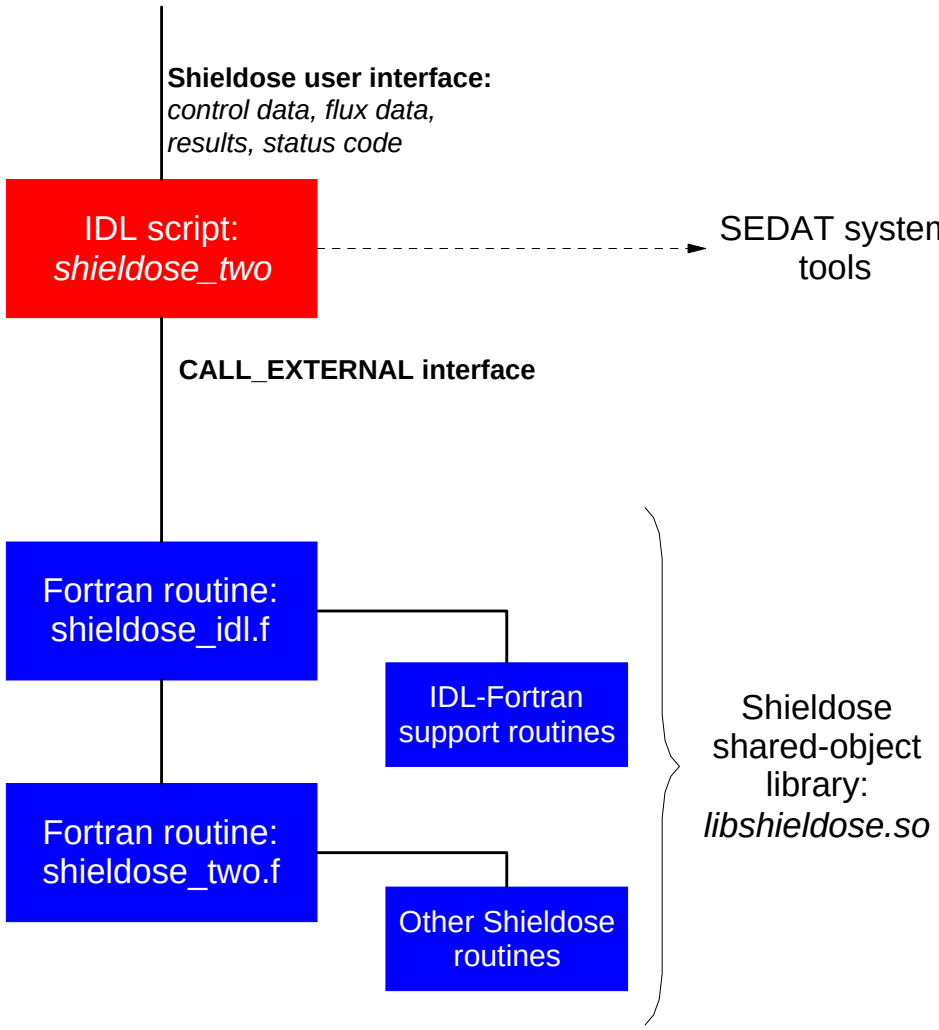

Figure 1. Architecture of Shieldose implementation in SEDAT

3.3 Test environment

We first verified that the original Fortran code compiled and executed correctly on the SEDAT host machine. The input for this test was the reference data supplied with the original code. The output was checked inspected to verify that it matched the results also supplied with that code. The results from this stage were then used as a benchmark for checking results from subsequent tests. This ensures that all results were generated on the same computer, thus facilitating automatic comparison with tools such as diff and vdiff. If the results file supplied with the original code were used as the benchmark automatic comparison tools would generate many spurious warnings, e.g. because of rounding differences (see Section 3.4.2).

We then established a set of related test environments to verify the correct implementation of Shieldose within the architecture shown in Figure 1. These are:

- a Fortran environment to test the main Shieldose-2 subroutine,
- an IDL environment to test the CALL_EXTERNAL interface to the Fortran wrapper and hence to Shieldose-2,
- an IDL environment to test the SEDAT interface to Shieldose-2.

Each environment supports output of results in two forms: (a) the usual Shieldose output files (SAMPLE.OUT and SAMPLE.ARR) and (b) a listing of the dose-depth data. The former can be compared directly with the reference output files supplied with the source code. The latter can be compared with an edited subset of SAMPLE.OUT. Note that output of SAMPLE.OUT and SAMPLE.ARR can be suppressed by changing these file names (which are control parameters) to the null device (/dev/null).

The environments can be used to check the relevant code after each major change. All tests have confirmed that the code outputs are in good agreement with the reference output supplied with the original code.

An important feature of the SEDAT implementation of Shieldose is that the software has been divided into two parts: (a) an initialisation that reads the external data files and derives various parameters and (b) a part that processes the input particle spectra to derive dose-depth curves. The initialisation need only be executed on first use; so this division improves execution speed when there are repeated sets of spectra to be processed. To verify this feature the test environments were enhanced to initialise Shieldose on the first call and then to make repeated calls (typically 5) to process the reference input data. Test outputs were identified by a file name suffix of the form –nnn, where nnn is the sequential number of the call to Shieldose (padded with leading zeroes).

3.4 Implementation of the Fortran code

3.4.1 Source code

The Shieldose source code is "SHIELDOSE-2, VERSION 2.10, 28 APR 94". It was downloaded from ftp://ftp.nist.gov/pub/shieldose/. It comes together with (a) the database files needed by the program, (b) a reference set of input and output files that we use for testing, and (c) a small amount of documentation.

3.4.2 Creating the Shieldose subroutine library

A copy of the source code was broken down into a set of files each containing one routine using the Unix utility fsplit (when using Fortran on Unix systems it is standard practice to use one file per routine). A makefile to build this code was then written. The makefile targets include:

- share – compiles all the subordinate routines to object code and assembles that into a shared-object library (we use this approach throughout for compatibility with the IDL CALL_EXTERNAL interface). The library file is libshieldose.so.
- sd2 – compiles and links the main program to produce a standalone executable sd2

To verify the correctness of this work, we executed the standalone executable sd2 using as input the reference control file supplied with the code. The resulting output files were compared with the reference output files supplied with the Fortran code. Very good agreement was obtained. Differences were only seen occasionally (perhaps 2% of data records) and then only in the least significant digit. These are attributed to differences in machine rounding and are not considered significant.

3.4.3 The main subroutine

The main steps required to convert the Shieldose main program from standalone form to subroutine library are outlined in Table 1. In addition, various minor changes were made as appropriate, e.g. disabling the output of screen control characters, closing output files. All changes are documented in the revised source code; please consult that for further details.

Whilst performing this conversion, we discovered that there was an undocumented control logic that allows the main programme to process multiple sets of flux data using the same initial target data. This has been disabled to support only a single pass through the software each time it is called. But it has also been used as the basis for supporting multiple calls to the main subroutine using different flux data but the same initialisation as discussed in Table 1.

Table 1. Conversion of the Shieldose main program to a subroutine.

N	Task	Notes
1	Encapsulate top level program as a subroutine	Building the call interface and disabling the STOP statement
2	Disable read from stdin	
3	Disable all reads from input file and replace by arguments in subroutine call	In some cases, new variables had to be specified to transport values from the call interface to the relevant point in the code. This step also revealed a control logic that allowed processing of multiple sets of particle spectra. This was changed to allow a single pass only.
4	Capture dose-depth curves in new arrays and output these via the call interface	
5	Add counter to track number of calls to subroutine.	Counter is integer variable N_CALLS and is included in the call interface. User must set to zero to initialise.
6	Report this to stdout	
7	Encapsulate initialisation (logo output, read of two large data files) in IF block that is only executed on user command (and if first call).	Separate IF blocks for logo and initialisation
8	Add SAVE attribute to routine to ensure retention of local variables set by above initialisation	Substantial comments added to explain this.
9	Read proton and electron data files from path specified in environment variable SEDAT_SUPPORT_PATH. Update OPEN statements to add STATUS=”OLD”, so the code searches for existing file and generates an error if this is not found (otherwise the code creates a new file and then fails when it finds no content).	Trap possible errors: • Environment variable not set • Error opening data files
10	Make code changes advised by Daniel Heynderickx to correct error in bremsstrahlung data tables for SHIELDOSE-2	Substantial comments added to explain this.
11	Use new variable MESSAGE_CONTROL to control printing of routine information messages to stdout. If its value is 0 routine messages are suppressed, If set to 1, messages are output, e.g. for diagnostic purposes.	Added in response to ESTEC request to suppress these messages. Substantial comments added to explain this. Keep option to switch messages on for diagnostic purposes – value set in code.

The main subroutine was validated in the Fortran test environment discussed above. This is a Fortran test harness call_sd.f that: (a) contains the reference input data as hard-coded values, and (b) calls shieldose_two.f using those data. It is compiled by makefile target call_sd and uses the Shieldose shared object library to resolve subroutine references. Thus, when running the test harness, the environment variable LD_LIBRARY_PATH must point to the directory containing that library.

3.4.4 The wrapper routines

The Fortran wrapper routine shieldose_idl.f is used to manipulate input and output between the IDL form used by SEDAT and the Fortran form used by shieldose_two.f.

The main manipulations concern the mapping of addresses to variables. IDL's CALL_EXTERNAL interface follows C language conventions; it passes subroutine arguments as (a) the number of distinct variables (argc) and (b) an array of addresses at which these variables are stored (argv). Thus the wrapper routine must extract these addresses and convert them to values as required by Fortran. This is done using the function %VAL, which is available on many Fortran compilers, e.g. %VAL(argv(3)) returns the value of the third variable.

The Fortran variables used to manipulate addresses, e.g. argv, must be integers whose size matches that used by the operating system – typically 32 or 64 bit. This operating-system-dependent size is coded in a single place, the Fortran module def_idl_types.f, by setting a parameter ADDRESS_KIND, which is the KIND parameter to be used when declaring address variables as integers. (Note the KIND parameter is a Fortran-90 feature used to set variable sizes; see the compiler documentation for the appropriate value.)

The wrapper routine also generates a 32-bit integer status code, e.g. if the number of distinct variables (argc) is incorrect, the wrapper exits without calling Shielddose and sets a non-zero status code. To return this code to IDL through the CALL_EXTERNAL interface, it must be copied to the address that IDL has assigned for the status code. This is done by subroutine set_int32.f.

The first implementation of the wrapper routine also supported conversion of strings between IDL and Fortran. However, this was abandoned when it was agreed that the effort needed to maintain that conversion was not worthwhile given that string conversion is not critical to Shielddose operation. The need for maintenance was driven by changes[1] in the way IDL stores strings. Instead the strings used in the main Shielddose routine are hard-coded in wrapper:

- The two strings defining output files PRTFIL and ARRFIL are set to '/dev/null' to suppress output
- The operation description string TAG_INPUT is simply set to 'Shieldose run from IDL'

The wrapper and its supporting routines are also included in the shared object library (libshieldose.so). Thus the whole Fortran code is available in this library and so can be called via CALL_EXTERNAL.

The wrapper routine must be tested from an IDL environment. This is an IDL test harness call_sd.pro, which is similar in content to the Fortran harness discussed in section 3.4.3. It contains the reference input data as hard-coded values, and calls the wrapper (shieldose_idl.f) using those data and the CALL_EXTERNAL interface. It uses the Shieldose shared object library to resolve subroutine references. Thus the IDL session must be started with LD_LIBRARY_PATH pointing to the directory containing that library.

[1] An IDL string is stored as a structure with three elements – the length of the string, a 16-bit type code (not for user manipulation) and the address (as an integer of appropriate length) at which its value is stored. The key change is that string length was 16-bit up to IDL 5.4, 32-bit from IDL 5.5.

3.5 SEDAT user interface to Shieldose

The user interface must allow a SEDAT user to provide the large amount of information needed by Shieldose and also return the detailed results back to that user. This has been implemented through three data structures. The input data is divided into two structures – one for flux data and one for control data; there is a single output data structure. The division of the input is done purely for convenience; many SEDAT applications (such as that described later in this report) will use a fixed set of control data but many different sets of flux data. Thus it is appropriate to separate out that more frequently changed data into a separate structure.

3.5.1 Control data.

This data structure contains the data that controls the operation of Shieldose, e.g. the detector type and the depths for which dose are to be calculated. The structure definition is as follows:

```
sd2_control_data={ IDET:     1L,             $
                   INUC:     1L,             $
                   IUNT:     -1L,            $
                   EMINS:    0.100D0,        $
                   EMAXS:    10000.000D0,    $
                   EMINP:    0.100D0,        $
                   EMAXP:    10000.000D0,    $
                   NPTSP:    1001L,          $
                   EMINE:    0.050D0,        $
                   EMAXE:    10.000D0,       $
                   NPTSE:    1001L,          $
                   N_CALLS:  0L,             $
                   ZM_ARRAY: dblarr(n_depth) }
```

Most of the tag names match those of control variables in the Shieldose program and are unchanged from the original code. The exceptions are:

- N_CALLS, which has been added in SEDAT in order to count the number of times that the Shieldose tool has been executed since the last initialisation. A new initialisation (which reads the proton and electron data files) can be forced by setting N_CALLS to zero. This must be done when first using the Shieldose tool.
- ZM_ARRAY is the array of depth values (which has various names in Shieldose). Note that n_depth is the number of depths.
- The Shieldose variable IMAX is not explicitly included as a tag as it is implicit in the array size.

The structure definition above sets sensible default values for all tags except the depth-related tags: IUNT and ZM_ARRAY. It can be executed by invoking the auxiliary tool make_sd_control_structure as follows:

```
make_sd_control_structure,n_depth,sd2_control_data
```

Use of this tool ensures that the data values are expressed in IDL formats that are consistent with those used in the CALL_EXTERNAL interface used to execute the Shieldose-2 Fortran program. These formats are typically that all reals must be 64-bit (IDL type DOUBLE) and integers must be 32-bit (IDL type LONG). This consistency is essential for safe operation. An error in the formats passed through the CALL_EXTERNAL interface may not just cause a simple SEDAT program error but has the potential to crash the underlying IDL session. Thus the SEDAT job may terminate abruptly without providing diagnostic information.

3.5.2 Flux data

This data structure contains the flux data from which Shieldose will derive doses. The structure definition is as follows:

```
sd2_flux_data={JSMAX:       -1L          , $
               JPMAX:       -1L          , $
               JEMAX:       -1L          , $
               EUNIT:       -1.0D0       , $
               DURATN:      -1.0D0       , $
               TAG:         ''           , $
               sp_energies: dblarr(n_sp), $
               sp_fluxes:   dblarr(n_sp), $
               tp_energies: dblarr(n_tp), $
               tp_fluxes:   dblarr(n_tp), $
               el_energies: dblarr(n_el), $
               el_fluxes:   dblarr(n_el)  }
```

The tag names match those of variables in the Shieldose program and are unchanged from the original code. n_sp, n_tp and n_el are the number of energy-flux data pairs for solar protons, trapped protons and electrons respectively. The scalar fields are initialised to unphysical values, so they must be explicitly set by the user. It can be created by invoking the auxiliary tool make_sd_flux_structure as follows:

```
make_sd_flux_structure,n_sp,n_tp,n_el,sd2_flux_data
```

As with make_sd_control_structure, use of this tool ensures that the data values are expressed in IDL formats that are consistent with those used in the CALL_EXTERNAL interface used to execute the Shieldose-2 Fortran program. Thus the tool is protected from the risk of crashing the underlying IDL session. The make_sd_flux_structure tool also initialises the energy and flux arrays to unphysical values, so they must be explicitly set by the user.

3.5.3 Results

This data structure contains the results from Shieldose. The structure definition is as follows:

```
sd2_dose_depth = {depth:   dblarr(n_depth,3), $
                  dose_el: dblarr(n_depth,3), $
                  dose_br: dblarr(n_depth,3), $
                  dose_tp: dblarr(n_depth,3), $
                  dose_sp: dblarr(n_depth,3)  }
```

The tags are:

- Array depth gives the depth values at which doses have been calculated. The first index selects different depths and the second different units (0=mils, 1= g cm^{-2}, 2 = mm).
- Array dose_el gives the doses due to electrons. The first index selects different depths (matching the first index of array depth) and the second different shielding configurations (0 = semi-infinite medium, 1 = transmission surface of finite slab shields, and 2 = half dose at the centre of a sphere.).
- Arrays dose_br, dose_tp and dose_sp give the doses due to bremstrahlung, trapped protons and solar protons respectively. Their indices operate in the same way as for dose_el.

This data structure is automatically build as part of the SEDAT Shieldose tool. The only user action needed is to extract results from the structure as needed to meet the user's aims.

3.5.4 Testing

The SEDAT user interface to Shieldose must be tested in an IDL environment. A script make_sd2_test_structures is used to initialise the input data structures used by shieldose_two.pro. with the reference input data. To do this it calls the auxiliary routines make_sd2_control_structure and make_sd2_flux_structure to build the data structures; it then uses hard-coded values to populate those structures. Having run make_sd2_test_structures, the user must then invoke shieldose_two.pro to run Shieldose and a tool such as print_test_dose_depth.pro to list the dose-depth data. This test uses the Shieldose shared object library to resolve subroutine references. Thus the IDL session must be started with LD_LIBRARY_PATH pointing to the directory containing that library.

3.6 Implementation of the IDL script

Shieldose is invoked from SEDAT via a IDL script shieldose_two.pro, which is implemented as a system tool so that it can invoke the CALL_EXTERNAL interface. The main functionality of the script is to:

- extract control and flux data from the structures supplied by the user as described above
- carry out various checks on these data as described in Table 2. If a problem is found, this is reported to the log and the script exits without running Shieldose.
- check that N_CALLS is ≥ 0 and reset it to zero if set negative,
- feed the validated data into Shieldose via the CALL_EXTERNAL interface
- pick up the Shieldose results returned via that interface,
- construct and populate the results data structure as described above
- generate a status code which is non-zero if a problem was encountered
- increment N_CALLS by one if execution was successful.

Table 2. Checks performed by the Shieldose front-end script

Input structures	Checks that sd2_control_data and sd2_flux_data are structures.
Detector type	Check that IDET has valid value
Nuclear attenuation	Check that INUC has valid value
Depth units	Check that IUNT has valid value
Depth values	Extract depths > 0, check that we have at least two valid depths.
Energy units	Check that EUNIT > 0
Duration	Check that DURATN > 0
Particle channels	Check that numbers of channels is either 0 (i.e. suppressed) or >2 (valid data). Do separately for solar protons, trapped protons and electrons.
Flux values	If number of channels > 2, check that number of valid fluxes (>0) matches the number of channels. Do separately for solar protons, trapped protons and electrons.

The display of information messages from the front-end script is controlled by a variable message_control, similar to that used in the main Fortran subroutine. Message_control is usually set to zero, which suppresses information messages. If set (in the code) to 1, the information messages will be output (e.g. for diagnostic purposes). Error messages are not controlled by message_control and will always be output if generated.

As noted above, the script invokes the Shieldose Fortran code via the CALL_EXTERNAL interface. The shared object library referenced by CALL_EXTERNAL is libshieldose.so and must be in a directory listed in environment variable LD_LIBRARY_PATH. The interface invokes the Fortran wrapper routine shieldose_idl in the shared object library but this must be referenced by a C-language style name of shieldose_idl_ (i.e. add a trailing underscore).

4 Shieldose features

This section notes some important features of Shieldose that were identified during the testing of the SEDAT implementation and that have been retained in that implementation.

4.1 Cut-off depth

The Shieldose Fortran code stops the calculation of the dose due to particles of a particular energy when the depth exceeds the range of the particles in aluminium at that energy. This became apparent when we used Shieldose to derive dose-depth curves using proton fluences from a limited range of energy channels. This is illustrated in Figure 2, which shows dose-depth curves derived using an early version of the tool mah!fluence_levels2. The input fluences (IMP proton data) have h a maximum energy of 60 MeV and the curves are cut off at a depth between 14 and 15 mm.

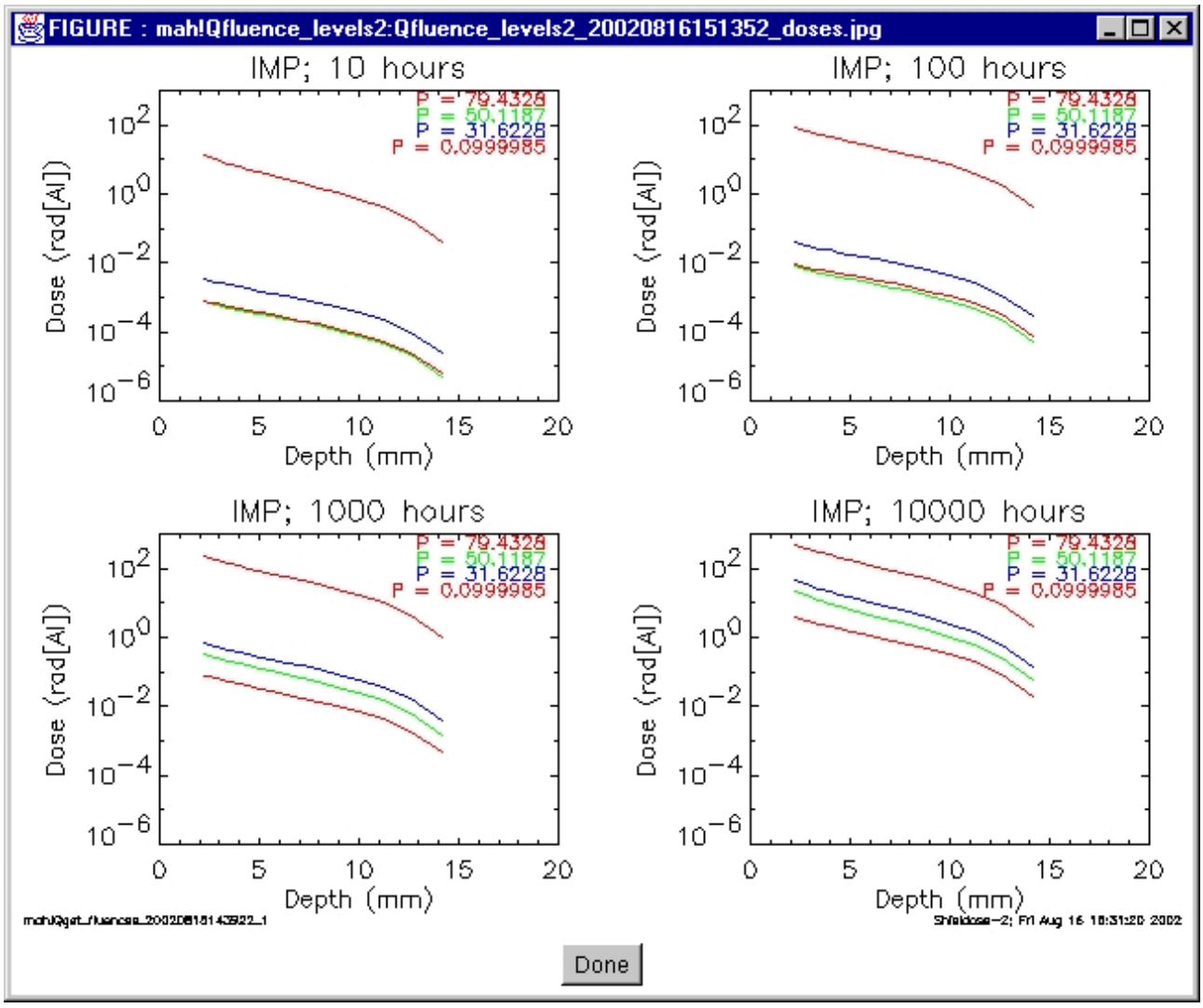

Figure 2. Dose-depth curves derived for proton data with a maximum energy of 60 MeV.

This cut-off is a fundamental property of the Shieldose program. It remained constant at 14 to 15 mm when the range of depths was doubled. Manual inspection of the code revealed a cut-off logic

(see Table 3 for an example), which suggested that the dose was set to zero at depths beyond some limit.

Table 3. Example of cut-off logic from the main Shieldose routine

```
ZRIN=Z(I)/RINP
      IF (ZRIN.LT.ZRP(LMAXP)) GO TO 640
      GP(NP,I)=0.0
      GO TO 645
  640 CALL BSPOL (ZRIN,LMAXP,ZRP,DIN,DINB,DINC,DIND,ANS)
      ANS=EXP(ANS)
      GP(NP,I)=TPP(NP)*ANS/RINP
  645 ……
```

This logic suggests the dose calculation is cut-off at the nominal range of energetic particles in a material. This was confirmed by plotting the range of protons in aluminium [R3] as shown in Figure 3. At a maximum energy of 60 MeV (as used in Figure 2), the range is about 14.7 mm – in excellent agreement with the cut off seen in that same figure.

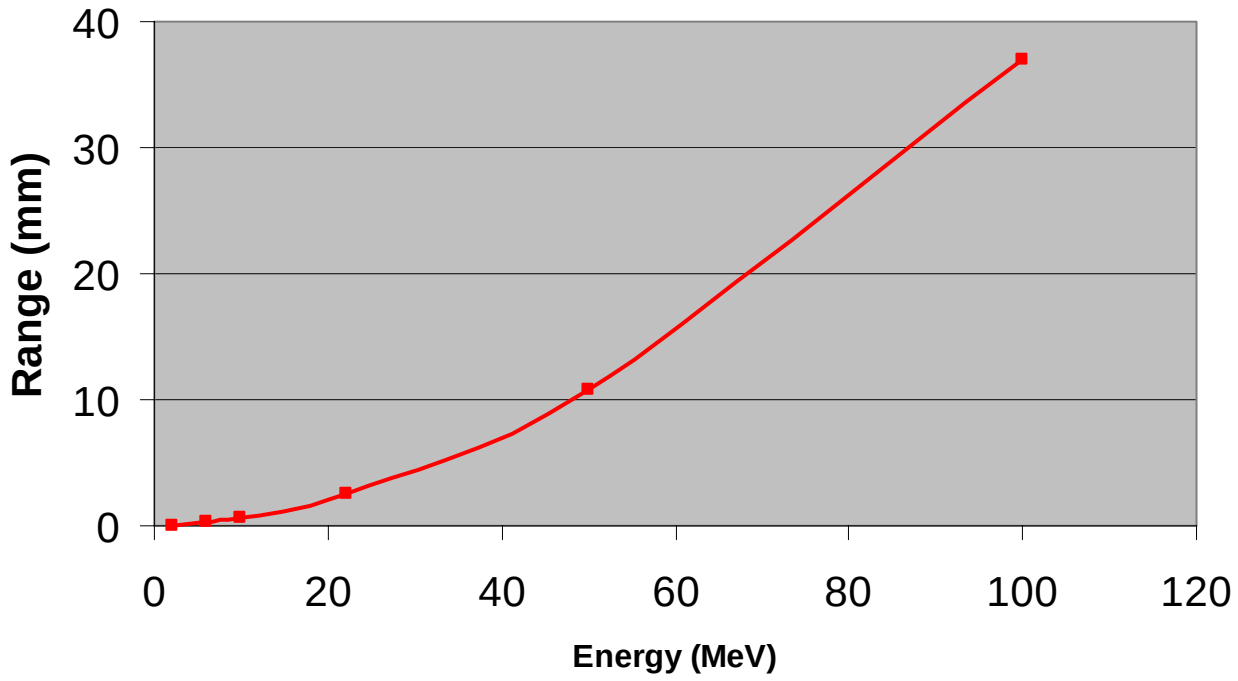

Figure 3. Range of protons in aluminium.

Thus we conclude that when using measured proton fluences to drive Shieldose (which is the key objective of WP302) it is important to consider if doses are required at depths close to or greater than the range of protons at maximum energy threshold of the dataset. If so, when running Shieldose, it will be necessary to extrapolate the measurements to higher energies.

5 Application to solar protons

5.1 Preparing the particle spectra

To calculate the mission doses due to solar protons, the Shieldose tool must be supplied with an estimate of the mission-integrated solar-proton fluences over a range of energies. In this demonstration, these fluences are those derived by statistical analysis of solar proton data in WP301 [R1]. That work package provides estimates of the percentiles of the solar proton distribution for a variety of periods and energies. Thus we can derive energy spectra of the fluence for any percentile level and duration. Figure 4 shows an example derived using all available IMP proton data.

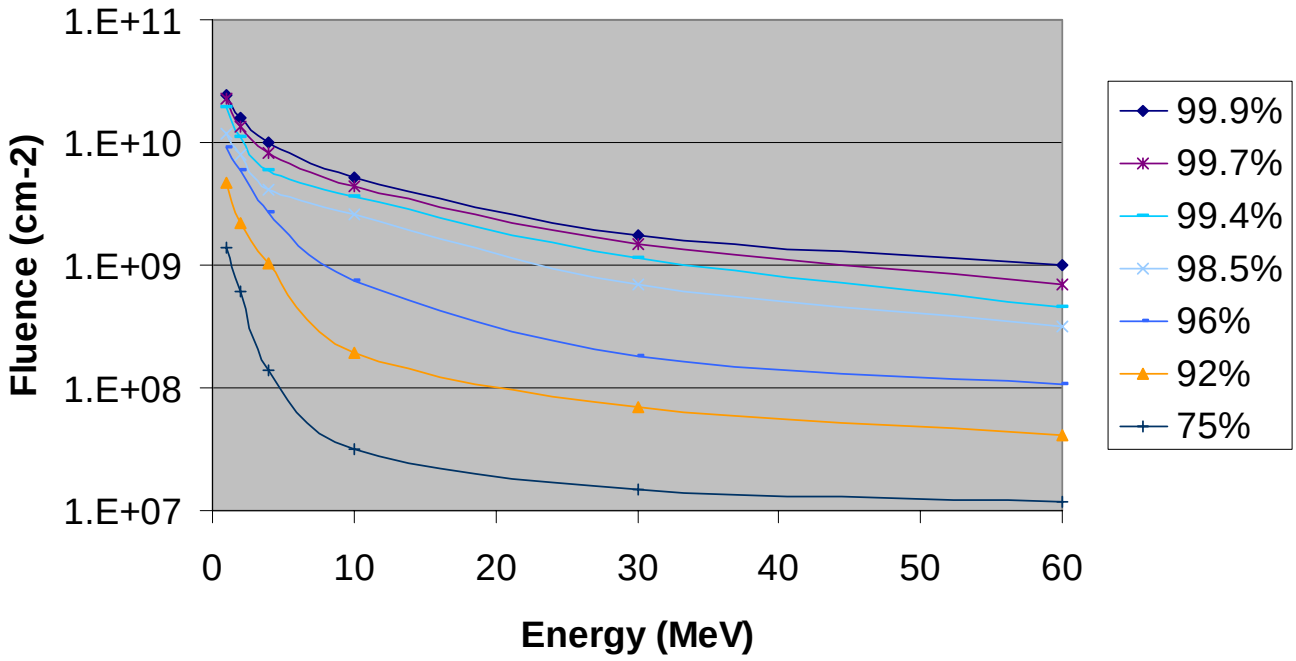

Figure 4. Energy spectra of solar proton integral fluence for a period of 1000 hours and for various probabilities (that the fluence will be ≤ the plotted level).

However, these fluence spectra cannot be used directly by Shieldose:

- the fluences used in WP301 are integral fluences whereas Shieldose requires differential fluences as input,
- the fluences may have to be extrapolated to higher energies in order to calculate complete dose-depth curves, as discussed in section 4.1.

Thus it is necessary to manipulate the fluence spectra before they can be used as input to the Shieldose tool. In principle, we could derive the differential fluences simply by numerical differentiation but this will also emphasise any noise in the data. It also fails to address the need for extrapolation. Thus we have fitted the integral fluence spectra to a suitable mathematical form that can give us a robust set of differential fluences and can extrapolate these to higher energies. The best mathematical form appears to be a power law, i.e. $F = F_0E^Z$, where F is the integral fluence and E is the energy. The coefficients F_0 and Z can be obtained by fitting a straight line to log F and log E ($\log F = \log F_0 + Z \log E$) so the exponent Z is given by the slope of that line and F_0 is derived from the intercept.

A power law gives a very good fit to the integral spectra with R^2, the square of the correlation coefficient, often better than 99%. In contrast, an exponential gives a much poorer fit with R^2 around 80%. This latter point is obvious when one looks at Figure 4. This figure plots fluence on a logarithmic scale and energy on a linear scale, so an exponential spectrum would appear here as a straight line – and the proton spectra are very clearly not straight lines in this figure.

This fitting is illustrated in Figure 5. The red points and curve show the results obtained from the IMP proton data. The blue line shows an excellent power law fit giving $F_0 = 6\times10^{10}$ and Z = -0.958, whereas the green line shows the poor fit of an exponential form.

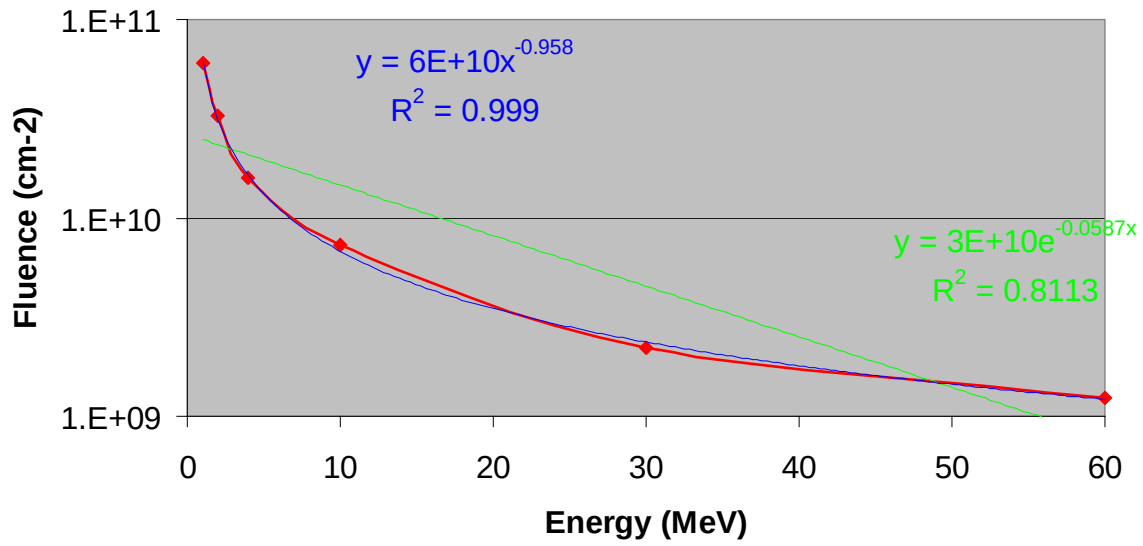

Figure 5. IMP proton integral fluence spectrum for a period of 10000 hours and a confidence level of 98%

The power law fit can easily be converted to give f, the differential fluence spectrum – simply by differentiating the power law function, i.e. $f = -dF/dE = -ZF_0E^{Z-1}$. Note that we must apply a negative sign to the differential to obtain a positive value for f. Given the fitted parameters and this equation, it is straightforward to generate differential fluence spectra over whatever range of energies is required for the dose-depth calculation.

The approach rests on the assumption that the spectral form can be extrapolated to energies higher than those measured. In the absence of other information, this seems the best solution.

5.2 Results

Figure 6 shows dose-depths curves derived from solar proton fluence spectra using the tool mah!fluence_levels2. This tool performed the following steps:

- determined the percentiles of the integral fluence distribution for period of 10, 100, 1000 and 10000 hours – on the lines for scalar (one energy) proton data developed in WP301 [R1], but here applied to vector (range of energies) proton data.
- converted these percentiles to differential fluence spectra as described in section 5.1, including extrapolation to higher energies to overcome the cut-off problem discussed in section 4.1.
- calculated the dose-depth curve for each percentile using the SEDAT tool for Shielddose
- plotted the results as shown; the percentile levels P are colour-coded and each value of P, in this case, is the probability that the fluence exceeds the plotted curve (i.e. P=1% indicates that in 99% of cases, the fluence will be less than that plotted).

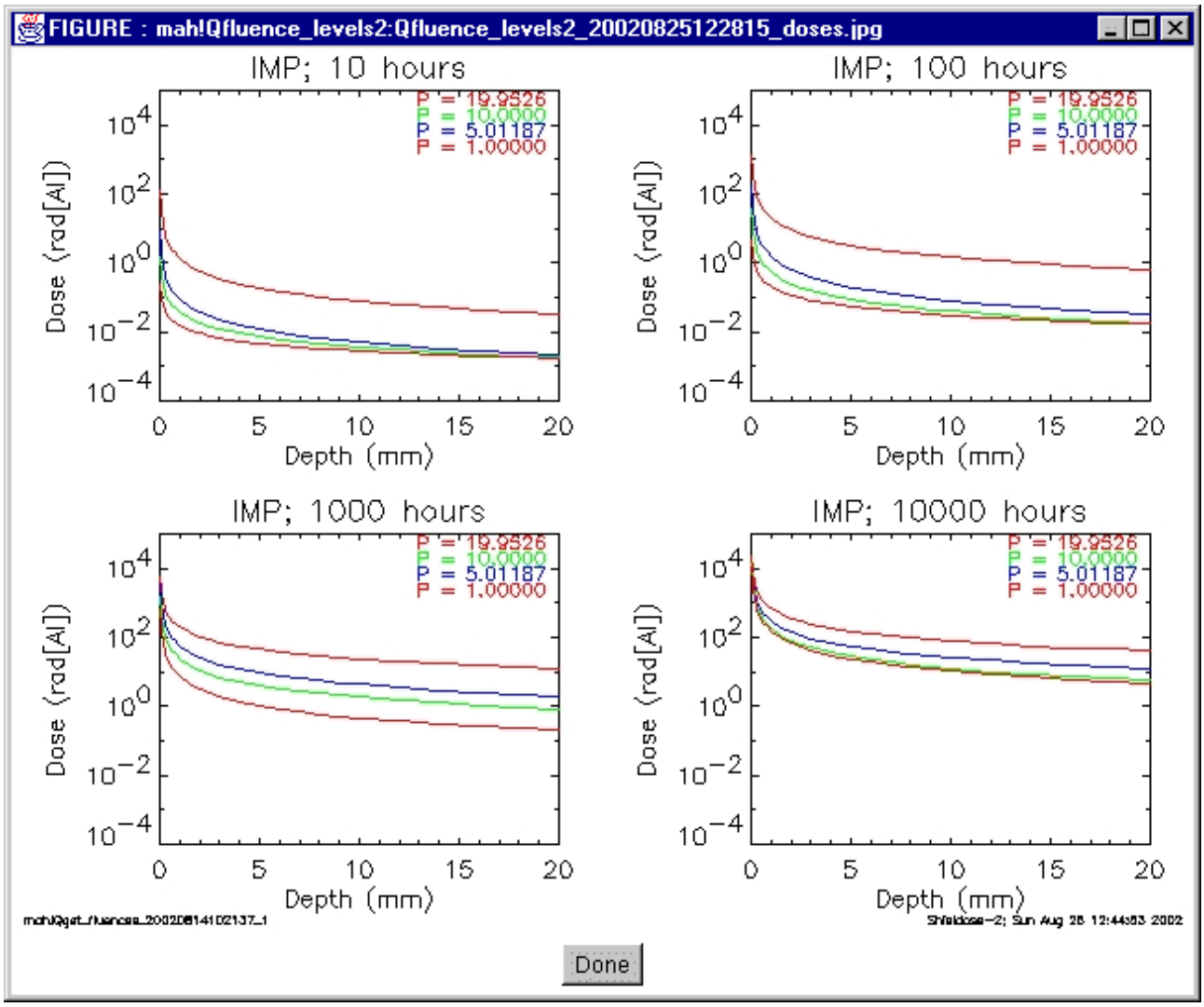

Figure 6. Dose-depth curves derived from the percentiles of IMP proton data.

Note again that these curves are based on extrapolation of fluences to energies above the highest energy channel (in this case 60 MeV). This extrapolation is done independently for each percentile

of each period. Thus it is possible that the curves may cross at depths greater than the proton range of the highest energy (14.7 mm for 60 MeV). Examples may be seen in the two upper panels of the figure.

5.3 File output

The dose-depth curves are written out to a SEDAT user dataset so that they are available for use by other tools. The fields in this dataset are:

- The depth value
- The various doses estimated at that depth – one for each quantile derived from the fluence distribution
- The probability levels associated with these quantiles
- The particle type (1=solar protons, 2=trapped protons, 3=trapped electrons and 4=bremsstrahlung)
- The code number for the data source (as in Table 2 of the WP301 report [R1])

The second and third fields (doses and quantiles) are 4-byte real arrays of equal length. The particle type and data source are 4-byte integers (IDL type LONG), while the depth is a 4-byte scalar real. The dataset is populated with appropriate metadata as supported by SEDAT. In particular, the units of depth and dose are stored together with the relevant SI relationship string [R10].

6 Application to trapped protons

6.1 Preparing the particle spectra

To calculate the doses due to trapped protons, the Shieldose tool must be supplied with an estimate of the trapped-proton fluxes over a range of energies and at various locations on the trajectory of interest. In this demonstration, these fluxes are those derived by statistical analysis of trapped proton data taken at these locations by previous missions. To do this we have re-used some techniques and code developed for other demonstrations:

1. The trapped proton flux dataset relevant to each point on the trajectory is selected as for the equivalent selection of electron fluxes in WP304. We scan through an appropriate dataset and find all flux measurements that were taken close to each point. In this case, closeness is specified in terms of magnetic position, e.g. that the McIlwain L value for the measurement and that at the point on the trajectory differ by less than one unit in L.
2. The trapped proton flux dataset is characterised statistically by deriving the quantiles of the distribution of flux values, i.e. the 60% quantile is the flux level such that 60% of the dataset has a flux lower than the quantile. The quantile approach was developed in WP301 [R1] and so we have re-used the function developed there for determining quantiles.

This approach gives the quantiles of the differential proton flux spectrum at each point on the trajectory and at whatever range of energies is available in the source dataset. Figure 7 below shows two examples of quantile spectra derived by applying this analysis to the Azur proton data for January 1970. The fitted curves are best fits to a power law ($y=ax^b$) and also show the square of the correlation coefficient. Thus, as for solar protons, we can use a power law fit to interpolate and extrapolate these spectra so that they are better conditioned for processing by Shieldose.

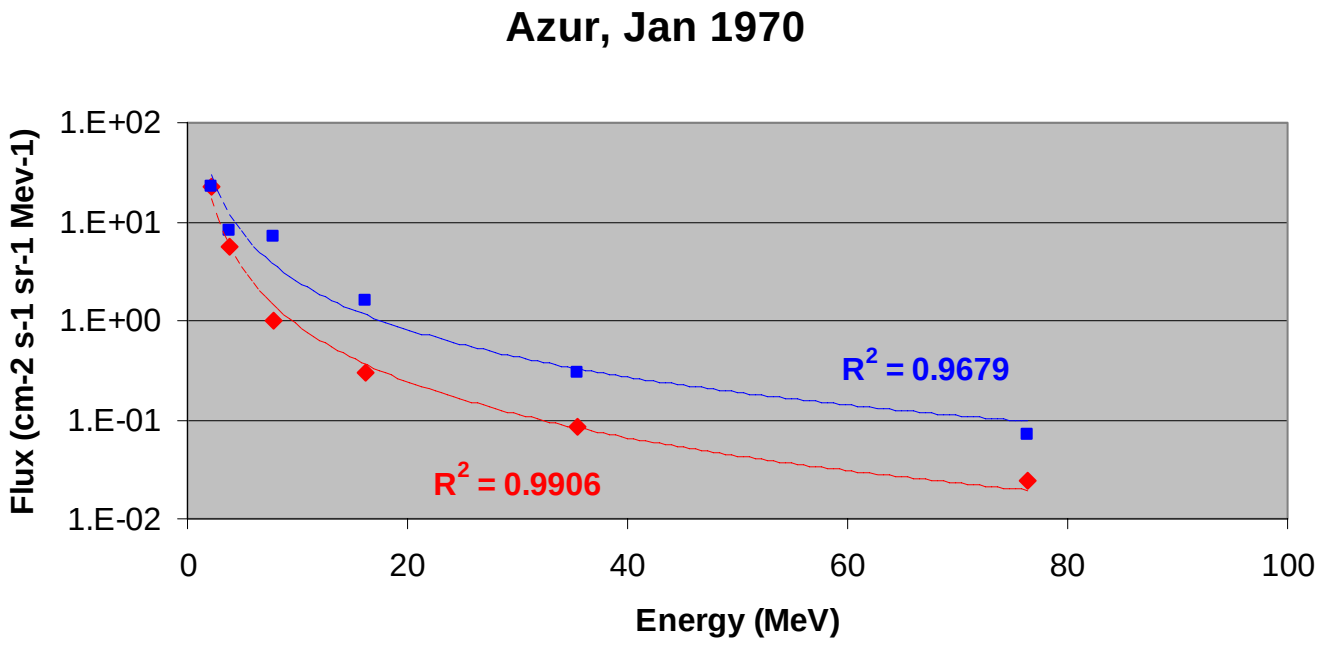

Figure 7. 95% quantiles for sample differential spectra derived from Azur proton data.

6.2 Results

This procedure yields a set of differential proton spectra with one spectrum at each point on the trajectory (and for each quantile level specified by the user). We then take each of these spectra and use Shieldose to convert it to a dose-depth curve – with the duration given to Shieldose being the time between that point and the next point on the trajectory. To derive a dose-depth curve for the whole trajectory we simply sum the dose-depth curves derived along the trajectory. This process is repeated for each quantile level so that we can associate each quantile with a mission dose-depth curve. This approach is implemented in tool mah!trapped_particle_doses2. Its output uses the same format as Figure 6 and an example based on Azur proton data is shown in Figure 8 below. For convenience of testing, this example takes the required trajectory through the radiation belt by re-using the XMM-Newton trajectory already prepared in WP304 – see section 4 of [R4].

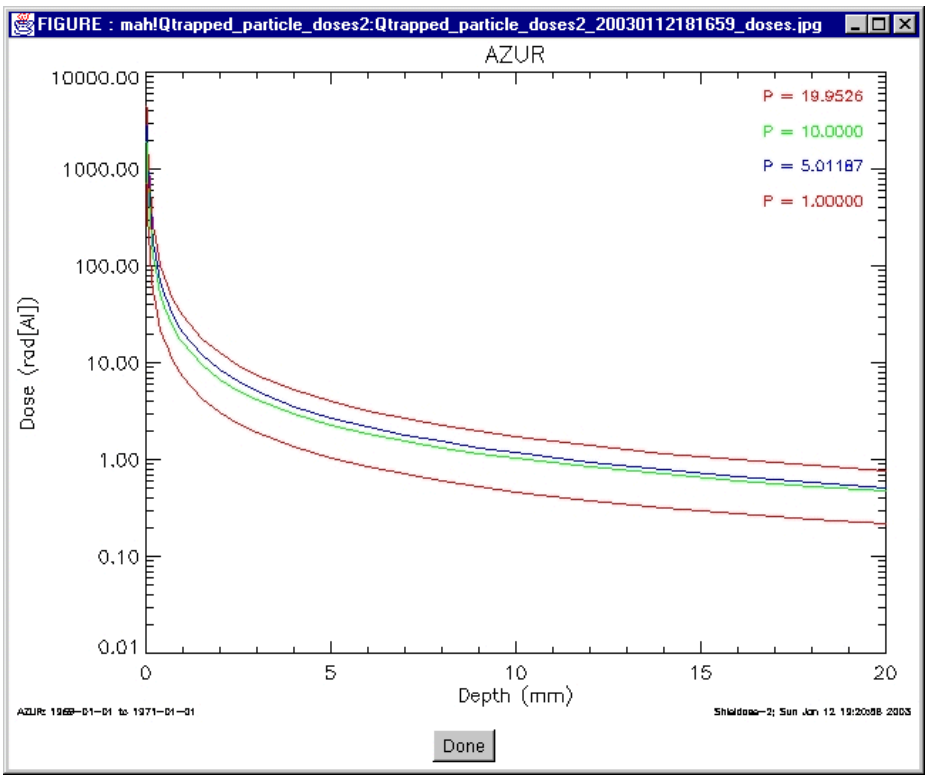

Figure 8. Dose-depth curve due to trapped protons for an orbit of XMM-Newton.

The dose-depth curves due to protons are written out to a SEDAT dataset using the format described in section 5.3.

The use of a power law for interpolation and extrapolation has another important benefit in that it allows us to check the quality of each quantile spectrum before processing it through Shieldose. We check:

- The sign of the power law exponent, i.e. b in $y=ax^b$. This must be negative in a physically realistic spectrum. Thus we flag a spectrum as bad if the exponent is positive.
- The square of the correlation coefficient. This is a measure of goodness-of-fit. We flag a spectrum as bad if $R^2 < 0.9$.

We only process spectra through Shieldose if both these criteria indicate that the spectrum is good. This has proved to an important mechanism to select good data. This is illustrated in Figure 9 and

Figure 10, which show dose-depth curves obtained by analysis of the same trapped proton data. If no quality checks are applied we obtain the bizarre result shown in Figure 9; the curve for 5% quantile is out of quantile order at most depths. If quality checks are applied we obtain the correct order as shown in Figure 10.

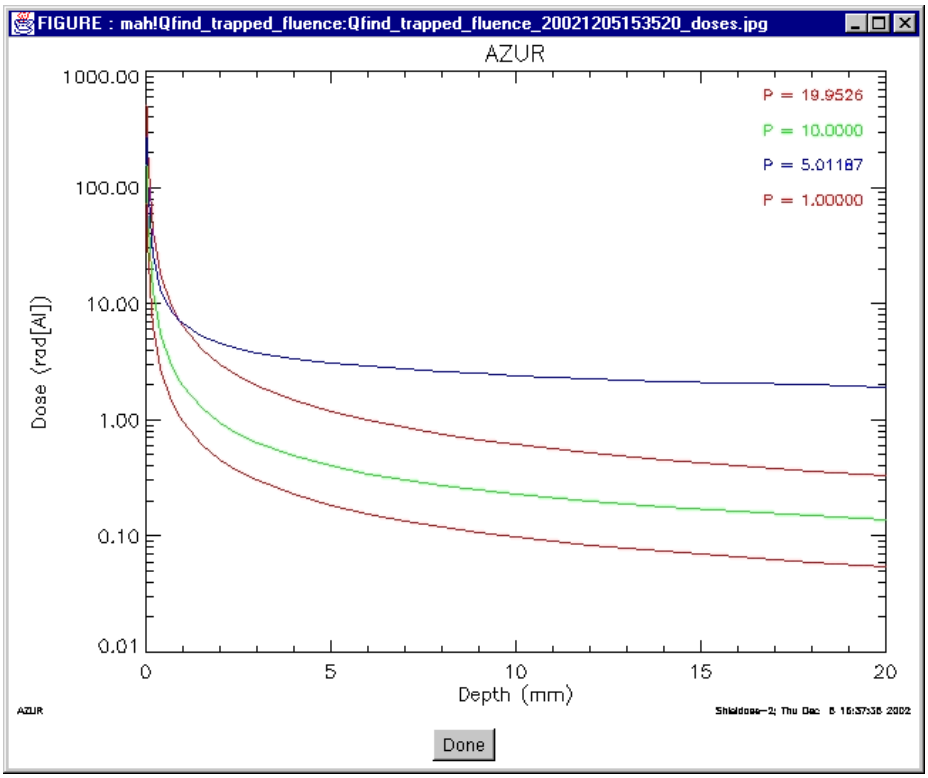

Figure 9. Dose-depth curves obtained without checking quality of spectra.

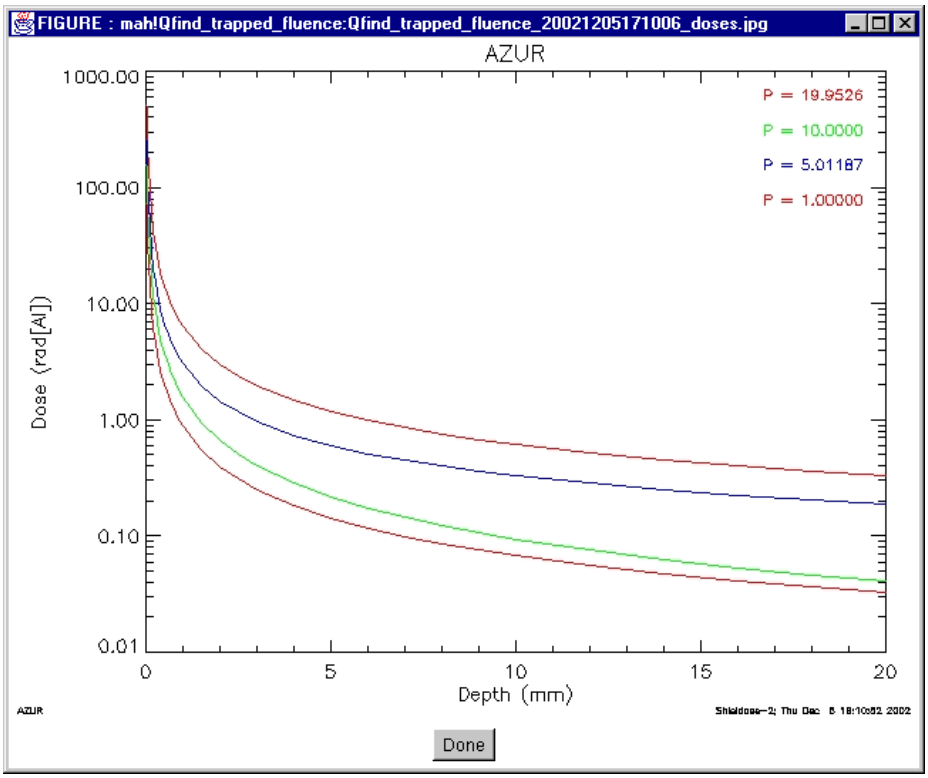

Figure 10. Dose-depth curves obtained after checking quality of spectra.

7 Application to electrons

7.1 Preparing the particle spectra

The procedure for calculating the doses due to trapped electrons is very similar to that for trapped protons, as described in section 6. The Shieldose tool must be supplied with an estimate of the trapped-electron fluxes over a range of energies and at various locations on the trajectory of interest. These fluxes derived by statistical analysis of trapped electron data taken by previous missions using the same techniques as described in section 6, i.e. re-use of the orbit generator developed for WP304 [R4], re-use of the same closeness criterion (magnetic position differs by less than one unit in L value) and re-use of the quantile approach was developed in WP301 [R1]. We also re-use the technique of fitting a power law to interpolate and extrapolate these spectra so that they are better conditioned for processing by Shieldose. Figure 11 shows an example of electron spectra derived using the quantile approach. These are well-fitted by power laws - with the square of the correlation coefficient, R^2 ,well above 0.9 in each case.

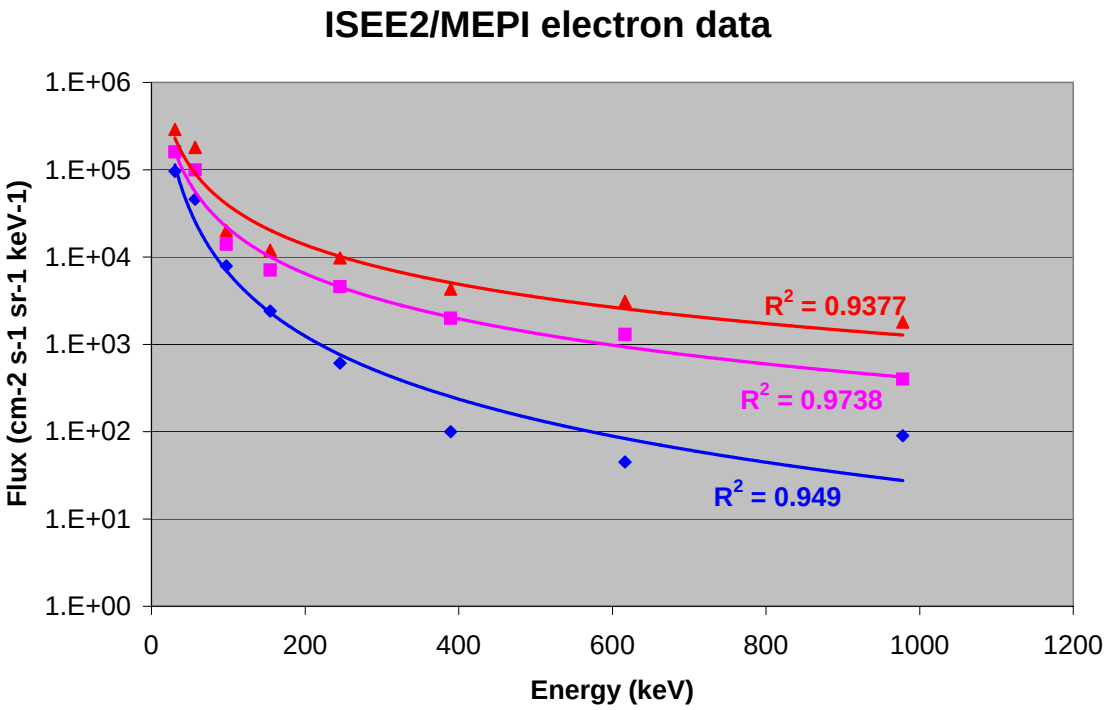

Figure 11. Power law fits to electron spectra

7.2 Results

This procedure yields a set of differential electron spectra with one spectrum at each point on the trajectory (and for each quantile level specified by the user). We then take each of these spectra and use Shieldose to convert it to a pair of dose-depth curves (one for doses due directly to the electrons and one for doses due to the bremstrahlung generated by the electrons). The duration required by Shieldose is taken as the time between that point and the next point on the trajectory. To derive the

electron and bremsstrahlung dose-depth curves for the whole trajectory we simply sum the dose-depth curves derived at each point along the trajectory. This process is repeated for each quantile level so that we can associate each quantile with a mission dose-depth curve for electrons and for bremsstrahlung. The output can be displayed in the usual format as Figure 6. An example based on processing the ISEE1/WIM electron data with tool mah!trapped_particle_doses2 is shown in Figure 12 below.

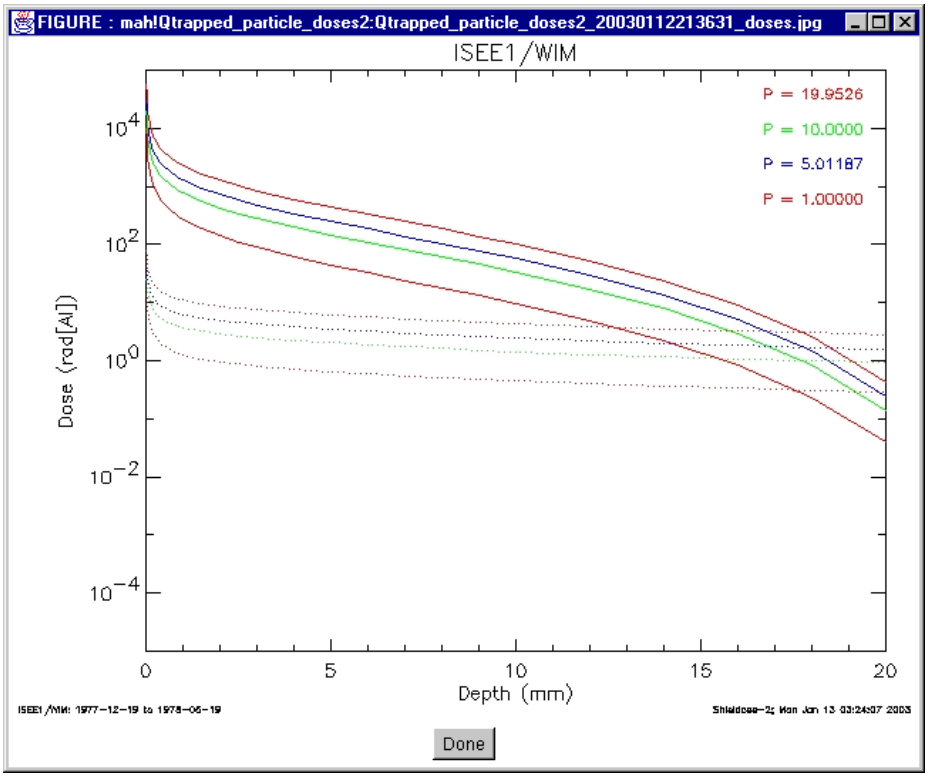

Figure 12. Electron (solid lines) and bremsstrahlung (dotted lines) doses for an orbit of XMM-Newton.

As when calculating the doses due to protons, it is important to extrapolate the electron spectra up to a sufficient energy that the electrons penetrate the full range of depths. This is easily done using the power law fit. The effect of truncating the extrapolate is shown in Figure 13 and Figure 14. They are produced using the same dataset and control parameters but with different extrapolation limits on the electron energy. The effect of truncating that limit can be clearly seen in the reduced doses at depths > 10 mm in Figure 14.

The dose-depth curves due to electrons are written out to a SEDAT dataset using the format described in section 5.3. Separate datasets are produced for the doses due to directly to electrons and those due to bremsstrahlung.

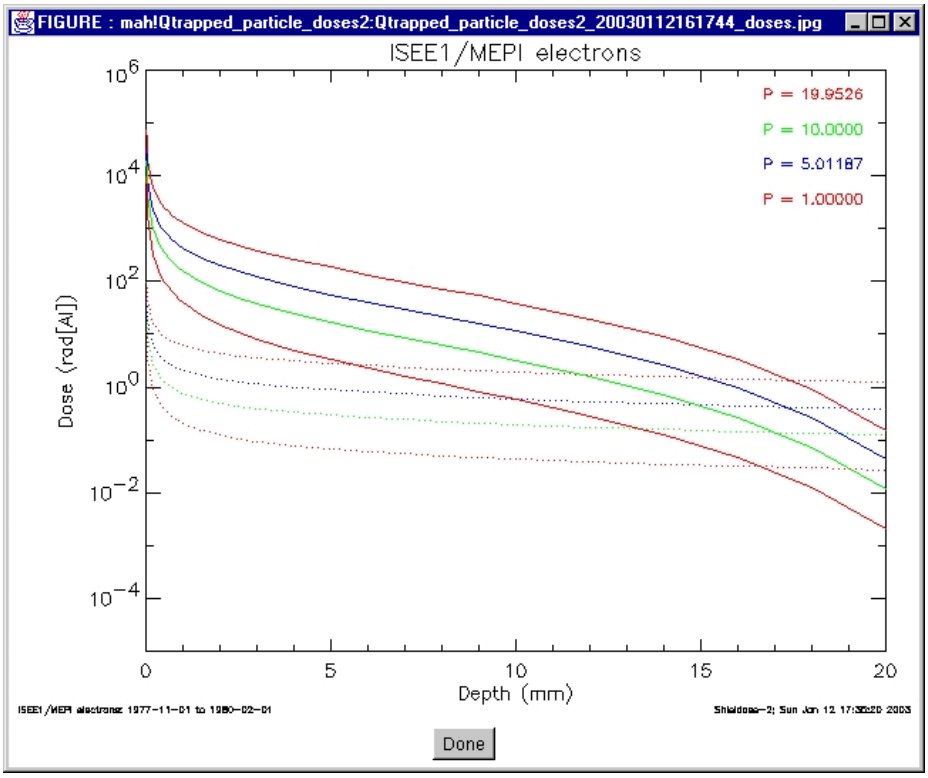

Figure 13. Dose-depth curves for electron spectra expolated to 10 MeV

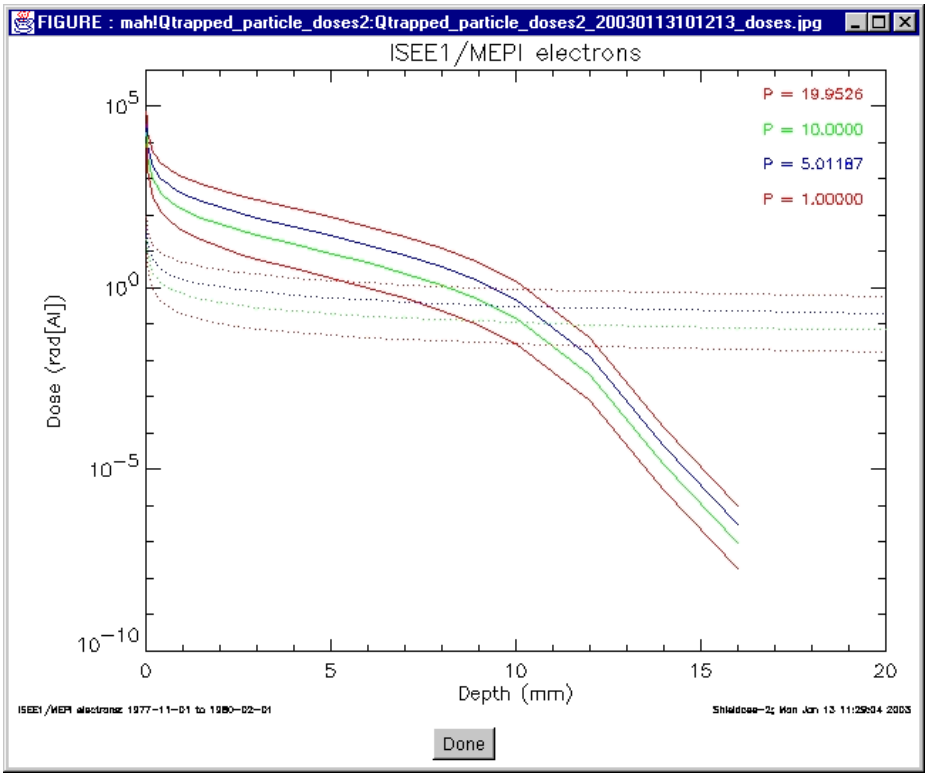

Figure 14. Dose-depth curves for electron spectra expolated to 5 MeV

8 Merging tool

The tool to merge and plot radiation dose-depth datasets, mah!merge_doses requires four datasets as input – one for each particle type. However, as a check the particle type is extracted from the appropriate field in each dose-depth dataset (see section 5.3). If the tool is given two or more datasets of the same particle type, it will plot the last dataset only - and the radiation doses for the missing particle types will be omitted.

The input datasets each contain several dose-depth curves corresponding to different probability levels (or quantiles of the flux distribution). However, to avoid an overly-busy figure, the tool only plots a single dose-depth curve for each particle type. These are selected, separately for each type, by finding the probability level that is closest to a user-supplied value. Figure 15 shows an example of the resulting plot. The quantile value shown at the lower level is the percentage probability that doses will exceed the displayed dose-depth curve, i.e. a quantile level of 1 is equivalent to 99% confidence level.

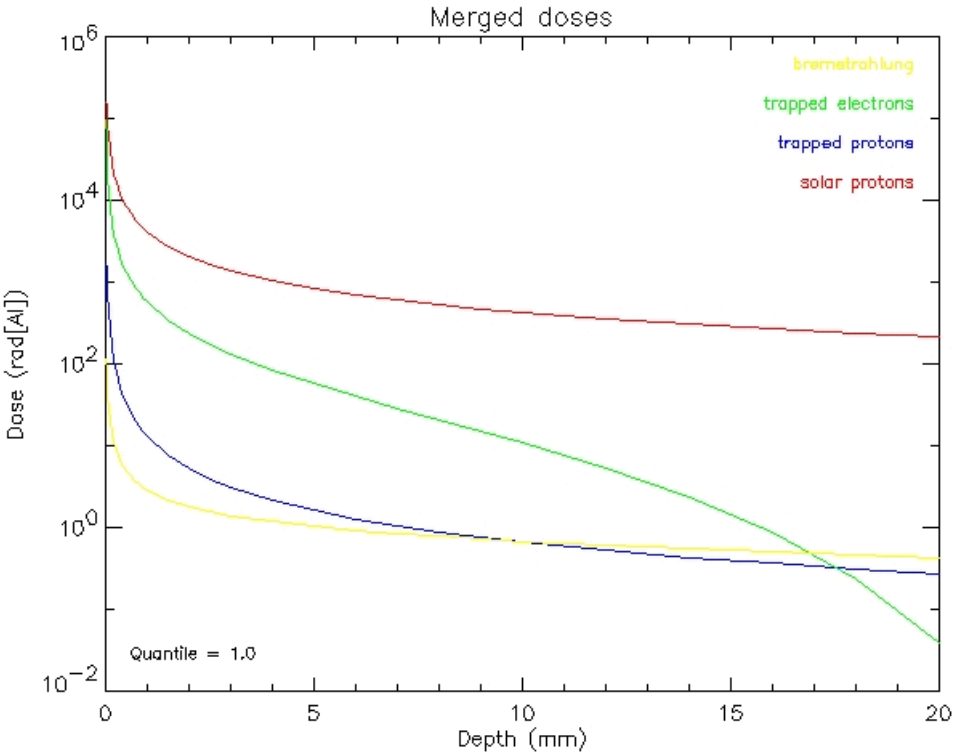

Figure 15. Example of merged radiation doses.

9 Combining the applications on a real orbit

9.1 Mission orbit

The mission orbit selected by ESA is a possible transfer orbit for ESA's Bepi-Colombo mission to Mercury[2]. This was supplied as an ASCII flat file in which each record gives the position and velocity of the spacecraft at a particular time. The fields of interest for this study are:

1. The time in days from 00:00 UTC on 1 January 2000.
2. The cartesian coordinates of the spacecraft in heliospheric aries ecliptic (HAE) coordinates [R8] for the epoch J2000. In this system X points towards the First Point of Aries and Z to the north ecliptic pole.

Other fields in the file include the cartesian components of the velocity at each time, the mass of the spacecraft (descreasing during the misison as fuel is used up) and the thrust angles.

The ASCII file ingested into an Excel spreadsheet so that each time field could easily be converted into year, month, day, hour, minutes and seconds. The times and cartesian coordinates were then written out to a comma-separated value file, which was used to create a CDF file that could easily be ingested into SEDAT. The conversion to CDF was done using a standalone IDL program. This program first created the empty CDF file with appropriate global and variable attributes [R5, R6] as shown in Table 4 below. The program then read each record of the CSV file, converted the time to CDF Epoch format [R7] and then wrote these times and the cartesian coordinates to the CDF file.

Table 4. CDF attributes for the early event data

Global attributes	Variable attributes
Acknowledgement	Fieldnam
Data_version	Fillval
Generated_by	Lablaxis
Generation_date	SI_conversion
Title	Units
Central_body	
Frame	

The CDF file was then ingested into SEDAT as a user dataset (mah!bepi_orbit_v2). To check the correctness of this ingestion, a tool (mah!plot_bepi) was written to extract the cartesian coordinates of the spacecraft and to plot the project of the trajectory on to the ecliptic plane (i.e. plot X versus Y) and to compare this with a plot supplied by ESA along with the data. This is shown in Figure 16 below which compares well with the reference plot supplied by ESA (see Figure 17). Note that the ESA plot also shows the orbits of Mercury, Venus and Earth – and uses colour to distinguish those parts of the trajectory when the spacecraft will be under thrust.

[2] Note that the orbit used here assumes a 2009 launch date for Bepi-Colombo; this changed to 2012 subsequent to delivery of the orbit data.

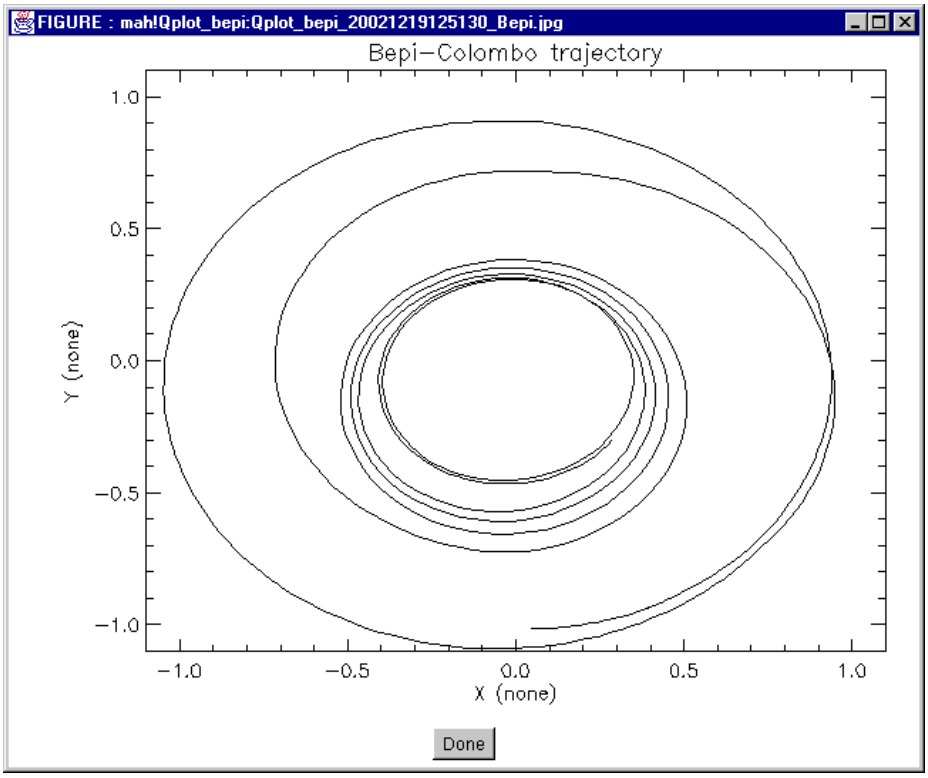

Figure 16. Trajectory of Bepi-Colombo produced by SEDAT.

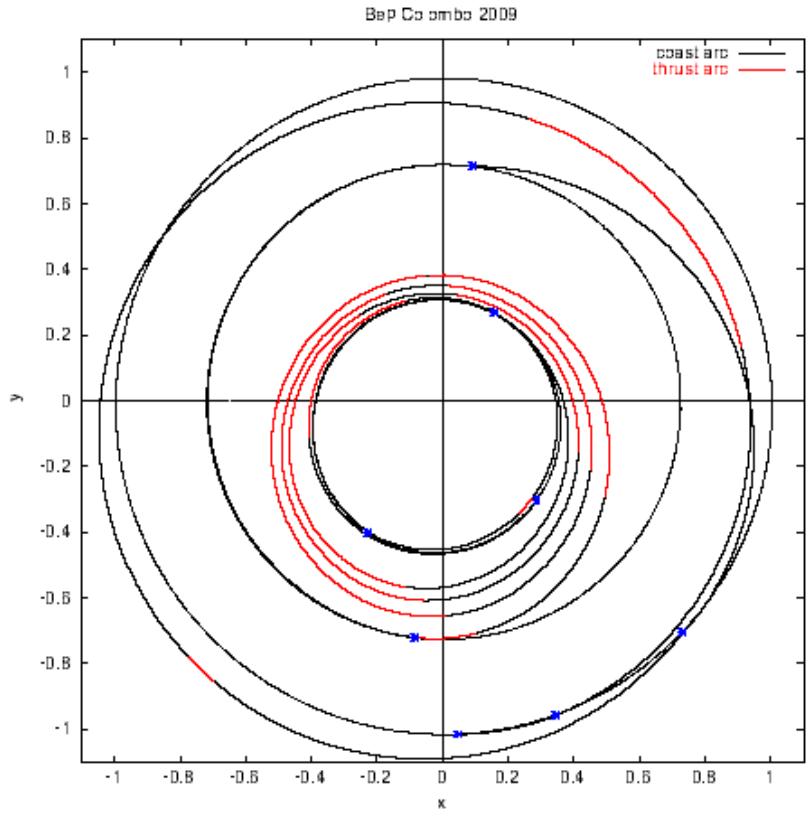

Figure 17. Trajectory of Bepi-Colombo supplied by ESA.

9.2 Position with respect to Earth

The trajectory described above is close to Earth at launch (24 June 2009)and during a fly-by on 8 August 2010. To determine the spacecraft position with respect to Earth on each of these two occasions, we must convert the spacecraft position from HAE to GSE (geocentric solar ecliptic) co-ordinates. This was done using the method outlined by Hapgood [R8]:

1. The spacecraft position is first converted from HAE to HEE (heliospheric earth ecliptic) coordinates. This conversion is a rotation about the north ecliptic pole, which is the common Z axis of the two systems. The angle of rotation is the Sun's ecliptic longitude, which was derived using the equations given in [R8].
2. This angle must be adjusted for precession to convert from the mean ecliptic of date form (as given by the equations in [R8]) to the ecliptic of J2000 (in order to match the Bepi-Colombo coordinates supplied by ESA). This correction was done using the equations given on page 105 of [R9]. It amounts to a shift of 300000 km along the Earth's orbit track (equivalent to a precession angle of 0.1 degrees).
3. The spacecraft position is then converted from HEE to GSE co-ordinates. This is a shift of origin from the Sun to the Earth and a 180 degree rotation about the Z axis – as described in [R8].

A tool (mah!plot_bepi_earth2) was then written to implement this co-ordinate transformation and display the results graphically. The latter are shown in Figure 18. The track at the lower left shows the initial highly elliptical orbit with apogee around 100 Earth radii and then the movement of the spacecraft away from the Earth following lunar fly-by (moving to the right in the figure). The track at the middle left shows the later Earth encounter – with the spacecraft approaching the Earth to a distance of a few tens of thousands of kilometres (the nominal figure is 20000 km) and then receding in a sunward direction. These plots show that we can reproduce the general features of the Earth fly-bys.

However, in both cases there is only a single trajectory point within the Earth's magnetosphere and its radiation belts. Thus in order to assess the fluxes of trapped protons and electrons encountered by the spacecraft, we interpolate the trajectory to obtain finer resolution.

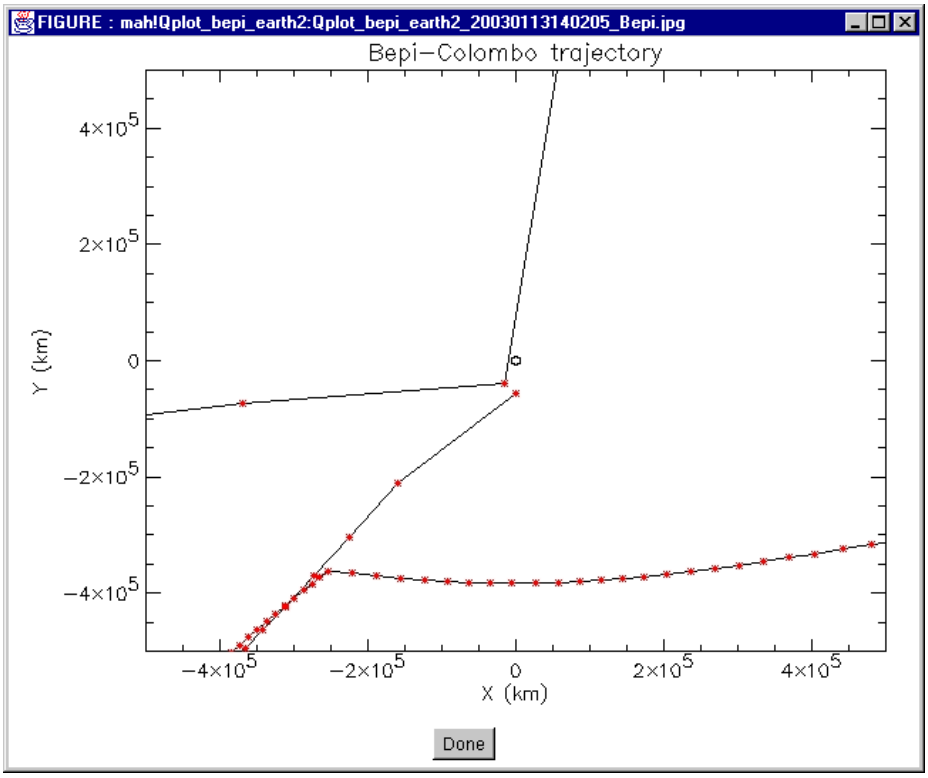

Figure 18. Bepi-Colombo trajectory with respect to Earth and in GSE co–ordinates. The red dots are the data points derived from the ESA data and are typically one day apart. The small circle at the origin shows the size and location of the Earth.

9.3 Doses due to solar protons

The fluences (and hence doses) due to solar protons may be estimated as follows using the tool described in section 5. However, the solar proton fluence must be adjusted to heliospheric distance of Bepi-Colombo. We assume that the solar proton flux varies as the inverse square of the distance from the Sun. Thus the mission fluence may be estimated as $\int f(t)/r(t)^2 \, dt$, where f(t) is the expected flux at the Earth, r(t) is the heliocentric distance of the spacecraft at time t and in expressed in Astronomical Units (i.e. r = 1 at the Earth) and the time integral is taken over the mission duration. Now, if we follow the approach used by Feynman [R11, R12], we may consider f(t) is zero during solar minimum years and has a constant positive value f_0 in the years around maximum (between 2.5 years before peak sunspot number and 4.5 years after peak sunspot number). Thus the mission fluence becomes $f_0 \int 1/r(t)^2 \, dt$, where the time integral is taken over that part of the mission duration in solar maximum years. Thus to derive the mission fluence we simply take that at the Earth and multiply by a correction factor $C = \int 1/r(t)^2 \, dt$. The tool mah!plot_bepi was extended to calculate the mission duration and this correction factor.

Examination of the Bepi-Colombo trajectory data using mah!plot_bepi shows that the mission duration from launch to arrival at Mercury is 3.34 years (starting June 2009 and ending October 2012). We use a simple tool to integrate over the trajectory and calculate the correction factor $C = \int 1/r(t)^2 \, dt$ assuming that the next solar cycle will peak in April 2011 (11 years after the peak of the present cycle). This yields a correction factor of 3.01.

Finally we modified the mah!fluence_levels2 tool (as described in section 5) to apply this correction factor to the mission fluences. The modified tool (mah!solar_proton_doses2) was run for the mission duration and with the above correction factor and using solar proton fluences derived from IMP-J measurements taken between 1973 and 2000. This yields the mission fluences show in Figure 19 and the dose-depth curves shown in Figure 20. Note the near-vertical shape of the fluence curves at probabilities < 20%. This arises because there are very few independent samples contributing to the fluence curves. Despite the 28-year run of IMP data, we can extract only a few independent samples to match the mission duration (especially given the constraint to sample solar maximum years only). A second consequence of the limited sampling is the overlay of the dose-depth curves for different probability levels as shown in Figure 20.

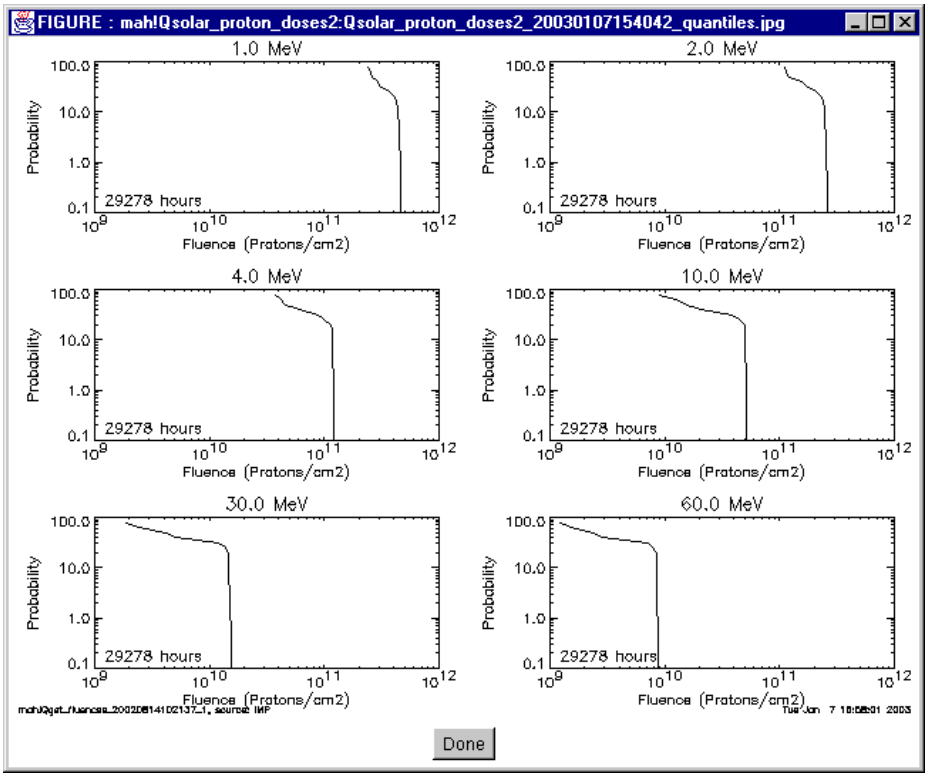

Figure 19. Solar proton fluence versus probability curves for the Bepi-Colombo trajectory.

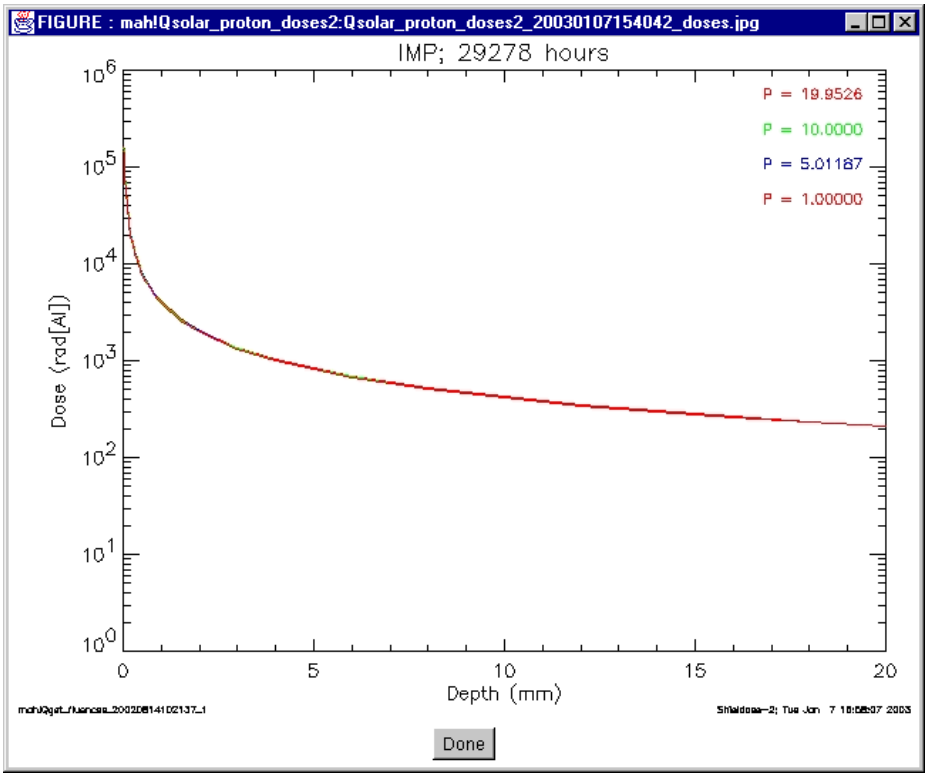

Figure 20. Solar proton dose-depth curves for the Bepi-Colombo trajectory

9.4 Doses due to trapped particles

The doses due to trapped particles (protons, electrons, bremsstrahlung) were estimated using the tool described in sections 6 and 7. To combine the effects of the two radiation belt passages (launch and the 8 August 2010 fly-by), we modified the mah!plot_bepi_earth2 tool so that it constructed and output a pseudo-trajectory that concisely combines the two encounter trajectories shown in Figure 18. It links them with a segment that is entirely outside the radiation belts; this is shown in Figure 21 below with the link segment marked in blue.

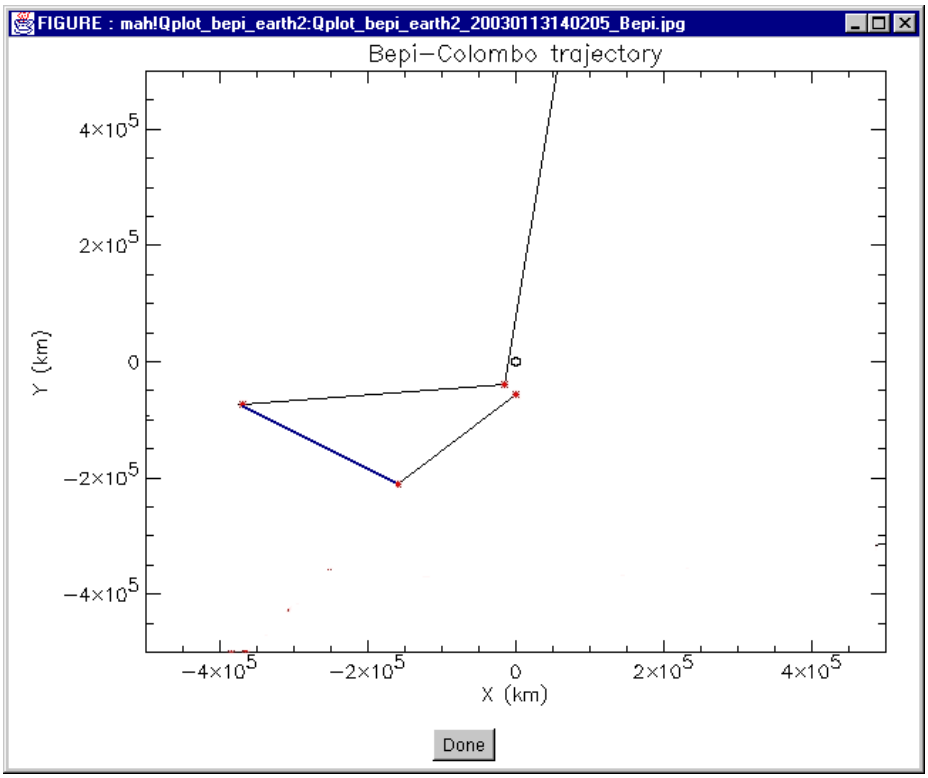

Figure 21. Pseudo trajectory for radiation belt encounters.

Given this pseudo-trajectory it is straightforward to use the trapped particle tool, mah!trapped_particle_doses2, to calculate dose depths curves. The results are shown in Figure 22 to Figure 25. The first pair of figures show the dose-depth curves due to protons using two different proton datasets (Azur and ISEE/MEPI), while the second pair show dose-depth curves due to two electron datasets (ISEE/WIM and ISEE/MEPI). There is a pleasing degree of consistency between each pair.

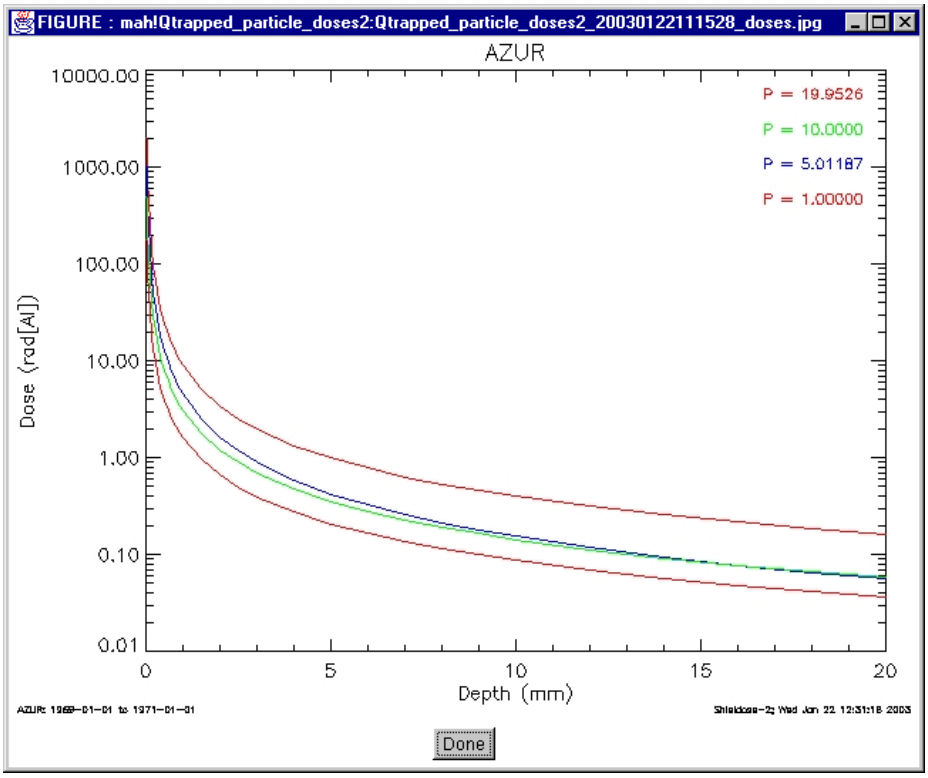

Figure 22. Dose-depth curves due to trapped protons for the Bepi-Colombo radiation belt crossings and using the Azur dataset.

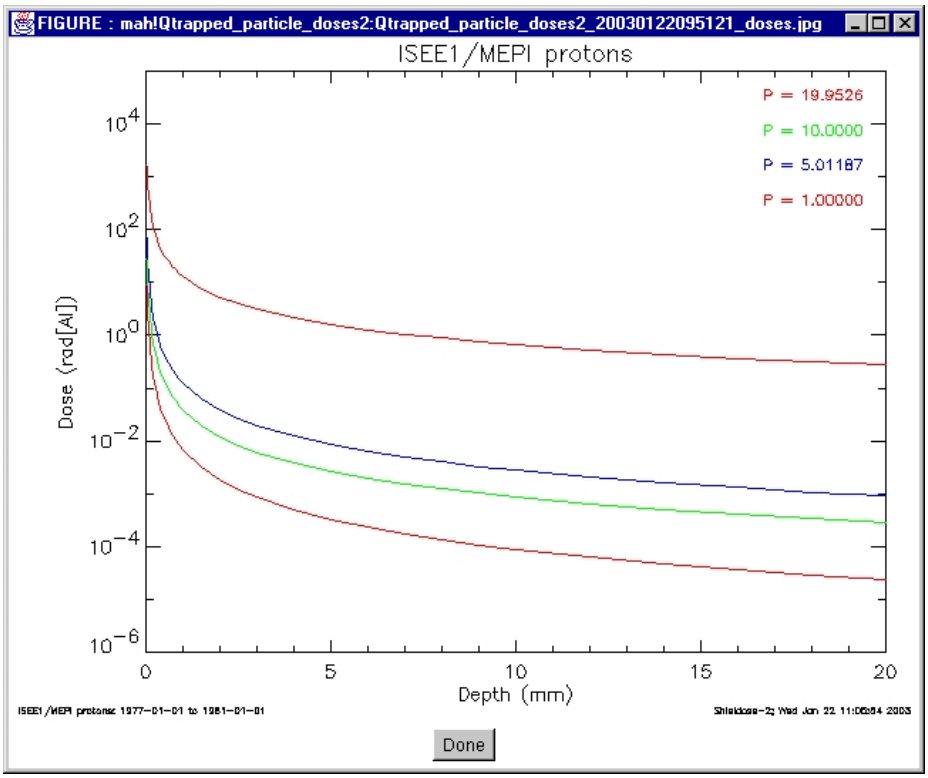

Figure 23. Dose-depth curves due to trapped protons for the Bepi-Colombo radiation belt crossings and using the ISEE/MEPI dataset.

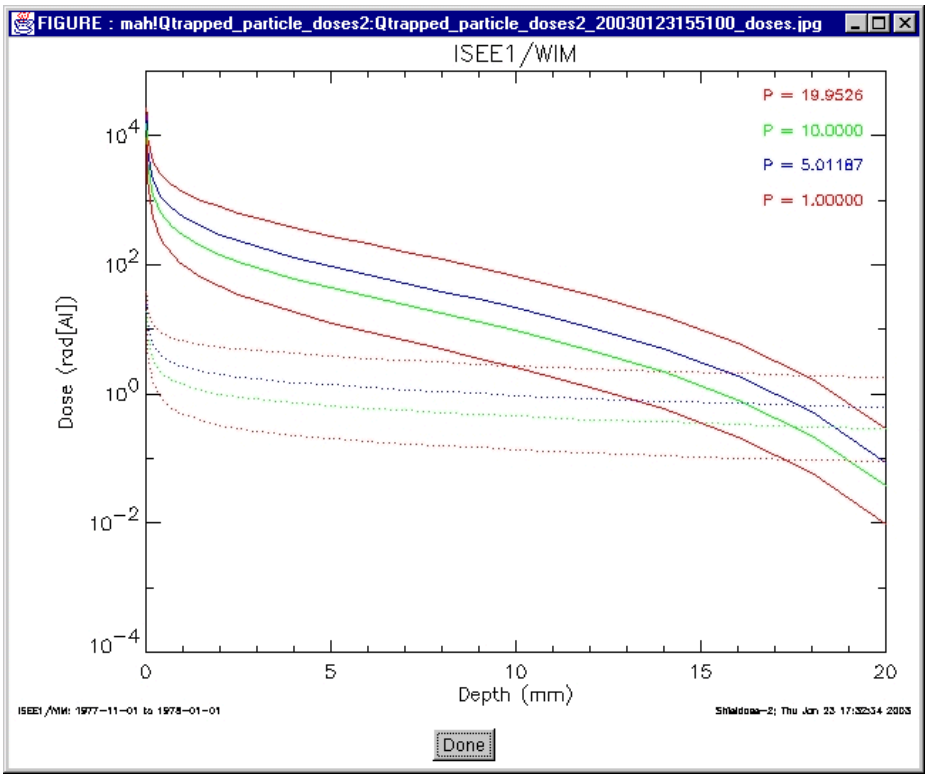

Figure 24. Dose-depth curves due to trapped electrons for the Bepi-Colombo radiation belt crossings and using the ISEE/WIM dataset.

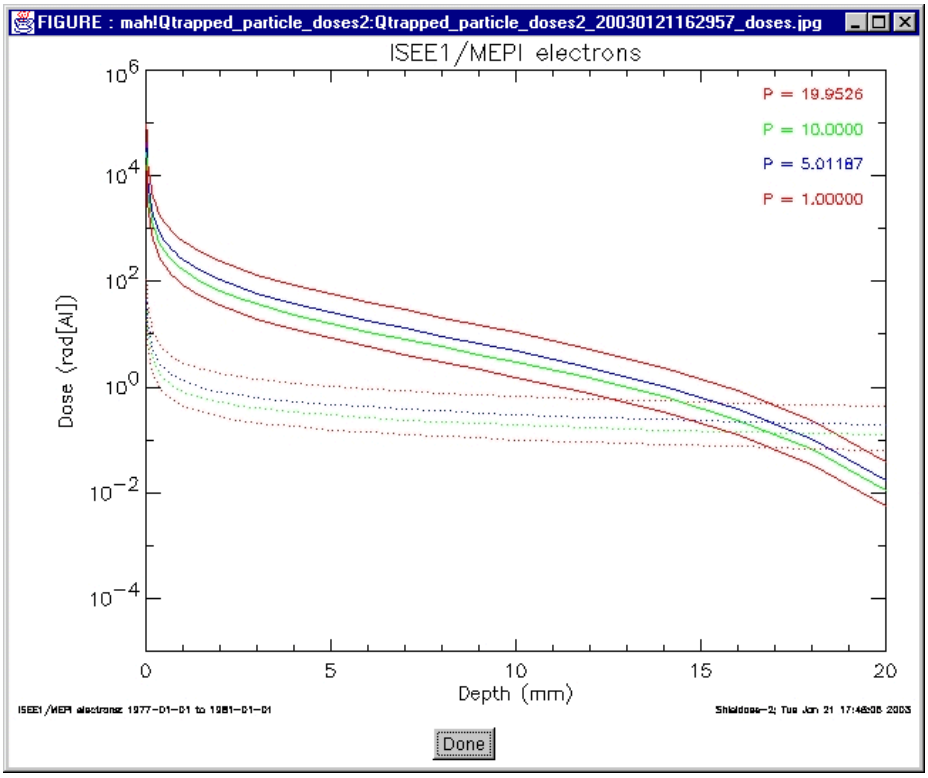

Figure 25. Dose-depth curves due to trapped electrons for the Bepi-Colombo radiation belt crossings and using the ISEE/MEPI dataset.

9.5 Merged results

The results for individual particle types can be merged using the mah!merge_doses tool described in section 8. The resulting plot for a confidence level of 95% is shown in Figure 26. You can see that solar proton doses dominate except at very small shield thicknesses as might be expected given Bepi Colombo's heliospheric trajectory.

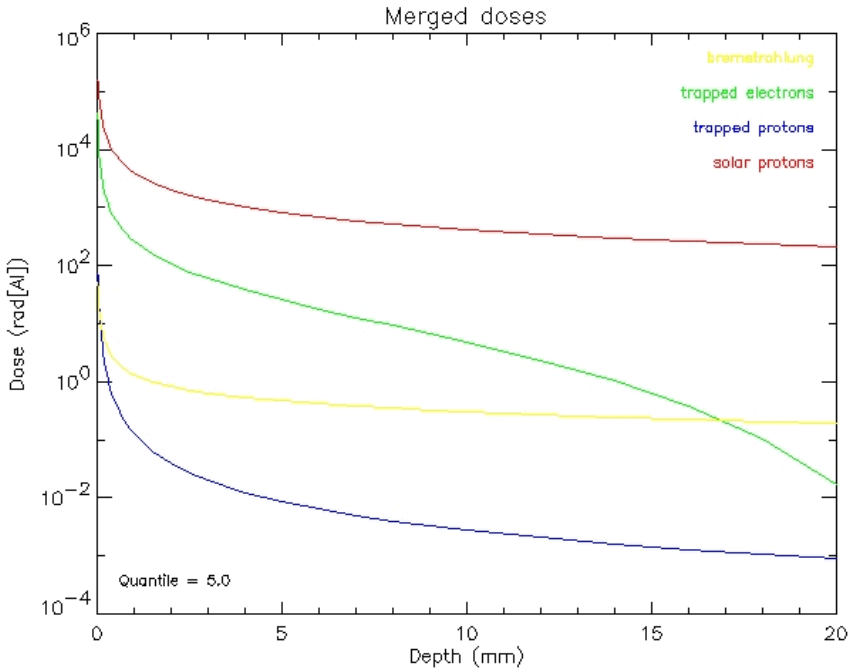

Figure 26. Merged dose-depth curves for Bepi-Colombo.

10 Conclusions

The Shieldose code has been adapted for use as a SEDAT tool. This involved the adaptation of the existing Fortran code to run as a sub-routine, rather than a standalone programme, and the invocation of that sub-routine from a SEDAT tool via IDL's CALL_EXTERNAL interface. The key steps in this adaptation were:

- The conversion of the code to run as Fortran subroutine. This involved the development of a suitable applications programming interface (API) and ensuring that API data, both input and output, were correctly transferred between the API and the relevant points in the code. The underlying philosophy was to make the minimum changes needed for success. The subroutine was set up so that the API could control whether (a) internal data structures should be populated by reading the proton and electron data files used by Shieldose, or (b) those data structures should be retained between calls to Shieldose (i.e. the Fortran SAVE command); this facility improves efficiency by reducing the need for repeated input from filestore.
- Testing of the subroutine. Care was taken to confirm that the subroutine functioned correctly when called from a Fortran test harness. This ensured correct functioning of the core code before integration into SEDAT. These tests included quantitative checks in which the code was run using reference inputs supplied with the source code and the output was compared to reference outputs also supplied with the source code. These showed good agreement with only small differences that could be attributed to platform-specific rounding effects. The test results provided new reference outputs that could be used in testing Shieldose integration into SEDAT; the new reference outputs had the advantage that they should be identical to results obtained during that integration and thus checks could be performed using tools such as diff and vdiff.
- Integration into SEDAT. This involved writing:
    - A Fortran wrapper routine to sit between Shieldose subroutine and the CALL_EXTERNAL access from IDL. This routine converts information passed by CALL_EXTERNAL (e.g. addresses of variables) into the form needed by the Fortran API. The wrapper was tested by a dedicated test harness, written in IDL, that calls Shieldose via the CALL_EXTERNAL interface and supplies the reference input data discussed above. The test harness output was identical to the reference output data thus confirming correct operation.
    - An IDL front-end procedure that calls Shieldose via the CALL_EXTERNAL interface and but supplies input data as provided by the arguments of the IDL procedure. The front-end was tested by using a separate procedure to put the reference input data in the form (data structures) required by the front-end. The front-end could then be executed. It was identical to the reference output data thus confirming correct operation.
- Various small changes were made to improve reporting of errors and to provide control of diagnostic messages (i.e. suppress them in normal operation, but allow easy switch-on if needed for trouble-shooting). Some recently discovered errors in the Shieldose code were also fixed.

Throughout this adaptation process much care has been taken to maintain a standard test environment and apply to confirm correct operation at each step of the adaptation. As a result the Shielddose code has been successfully converted to run as a code library that can be invoked from Fortran, IDL and SEDAT.

The testing of the adapted Shielddose code revealed several features that are not well documented. Most importantly, the code truncates the calculation of the dose at the nominal penetration depth of the type and energy of the particle concerned. Thus when using observed particle spectra the dose-

depth curve will exhibit a rapid decline at depths beyond the nominal penetration depth for the highest measured energy. To determine the dose at greater depths, it is essential to fit the measured flux spectra to a suitable mathematical form and use that to extrapolate the measured spectrum to greater energies. Other undocumented features include the ability of the code to process multiple sets of flux spectra. This was disabled in the adaptation so that such looping must be controlled by the user at higher level in the code.

A number of SEDAT tools have been developed to exploit Shieldose by calculating dose-depth curves due to solar protons and due to trapped particles (both protons and electrons). These tools include the functionality required to put observed particle data in the correct format (e.g. the integral fluences normally used for solar proton data must be converted to differential values) and extrapolate them to higher energy to overcome the cut-off issue discussed above. In all cases, the tools calculate the flux spectra for a range of probabilities and use them to derive dose-depth curves over the same range. The outputs of these tools are datasets containing dose-depth-probability values for different radiation types – solar protons, trapped protons, trapped electrons and bremsstrahlung. These output can be merged and displayed on a single plot using a merging took.

The final stage of this demonstration was to apply the tools to a reference mission selected by ESA and using orbit data supplied by ESA. This was Bepi-Colombo; the orbit data were supplied as its HAE coordinates (Epoch J2000.0) based on a launch in 2009. The orbit data were analysed using a couple of new tools: (a) to confirm that the plotted heliospheric orbit matched a plot supplied by ESA, (b) to calculate the mission duration and an enhancement ratio that could adjust the predicted 1AU solar proton fluences to those expected at the spacecraft, (c) to display the geocentric trajectory following launch and during fly-by, and (d) to construct a pseudo-trajectory that concisely represented all the radiation belt crossings by Bepi.

The dose-depth tools discussed above were then used to calculate the doses on Bepi due to solar protons, trapped protons, trapped electrons and bremsstrahlung. In the case of solar protons some adaptation was required to incorporate the effects of mission duration and the enhancement ratio. Finally the merging tool was used to display the doses expected on Bepi due to the different types of radiation. Unsurprisingly, this showed that solar protons were expected to dominate.

11 Annex A - Shieldose input data

The following tables give a detailed specification of the tag names used in the control and flux data structures that are the SEDAT input interface for running Shieldose-2.

Table 5. Control data fields for the SEDAT implementation of Shieldose

Field name	Description
IDET	Detector type (1 to 11) thus: 1, Al detector 2, Graphite detector 3, Si detector 4, Air detector 5, Bone detector 6, Calcium fluoride detector 7, Gallium arsenide detector 8, Lithium fluoride detector 9, Silicon dioxide detector 10, Tissue detector 11, Water detector
INUC	Type of nuclear attenuation (1 to 3) 1, No nuclear attenuation for protons in Al 2, Nuclear attenuation, local charged-secondary energy deposition 3, Nuclear attenuation, local charged-secondary energy deposition, and approx exponential distribution of neutron dose
IUNT	Shield depth units (1 to 3) 1 = mils 2 = g/cm2 3 = mm
ZM_ARRAY	Shield depth values – an array of real values
EMINS	TBD
EMAXS	TBD
EMINP	TBD
EMAXP	TBD
NPTSP	TBD
EMINE	TBD
EMAXE	TBD
NPTSE	TBD
N_CALLS	Number of calls made to Shieldose since last initialisation. Zero forces new initialisation.

Table 6. Flux data fields for the SEDAT implementation of Shieldose

TAG	72 character string short description of current simulation
EUNIT	Conversion factor from /energy to /MeV, e.g. EUNIT = 1000 if flux is /keV.
DURATN	Mission duration in multiples of unit time, e.g. if unit = 1 sec, DURATN = 3.15360E+07
JSMAX	Number of points in solar proton spectrum (0 if not supplied)
JPMAX	Number of points in trapped proton spectrum (0 if not supplied)
JEMAX	Number of points in electron spectrum (0 if not supplied)
SP_ENERGIES	Energy levels in solar proton spectrum
SP_FLUXES	Solar proton spectrum (over mission duration – TBC) as incident omnidirectional fluence in /energy/cm-2
TP_ENERGIES	Energy levels in trapped proton spectrum
TP_FLUXES	Trapped proton spectrum as incident omnidirectional flux in /energy/cm-2/unit time
EL_ENERGIES	Energy levels in electron spectrum
EL_FLUXES	Electron spectrum as incident omnidirectional flux in /energy/cm-2/unit time

It is possible to specify use of model spectra by setting the number of points to three (tbc), the energy levels to zero, and supplying the model parameters in the spectrum. It is not planned to make use of this functionality during the SEDAT demonstrations, but the SEDAT interface to Shieldose will allow the user to control the parameters described above – and thus it will be possible to invoke this functionality.

12 Annex B. Data sources

Table 7. Data sources

Data_source code	*Description of data source*
1	IMP proton fluxes
4	AZUR protons
5	GOES-5 integral proton fluxes
6	GOES-6 integral proton fluxes
7	GOES-7 integral proton fluxes
9	ISEE1/MEPI protons
10	ISEE1/WIM electrons
19	ISEE1/MEPI electrons

13 Annex C – summary of tools used in this demonstration

Figure 27 and Figure 28 below show the top-level tools used this workpackage (blue boxes) and the SEDAT datasets flowing between those tools (green boxes). The first figure shows generic tools used to handle vector values of solar proton fluxes and fluences, while the second shows tools related to radiation analysis along a specific spacecraft trajectory (such that the Bepi-Colombo example used here). The full set of top-level tools and queries is listed in Table 8 together with a summary of the functionality that they provide. The parameters for these queries are listed in Table 9.

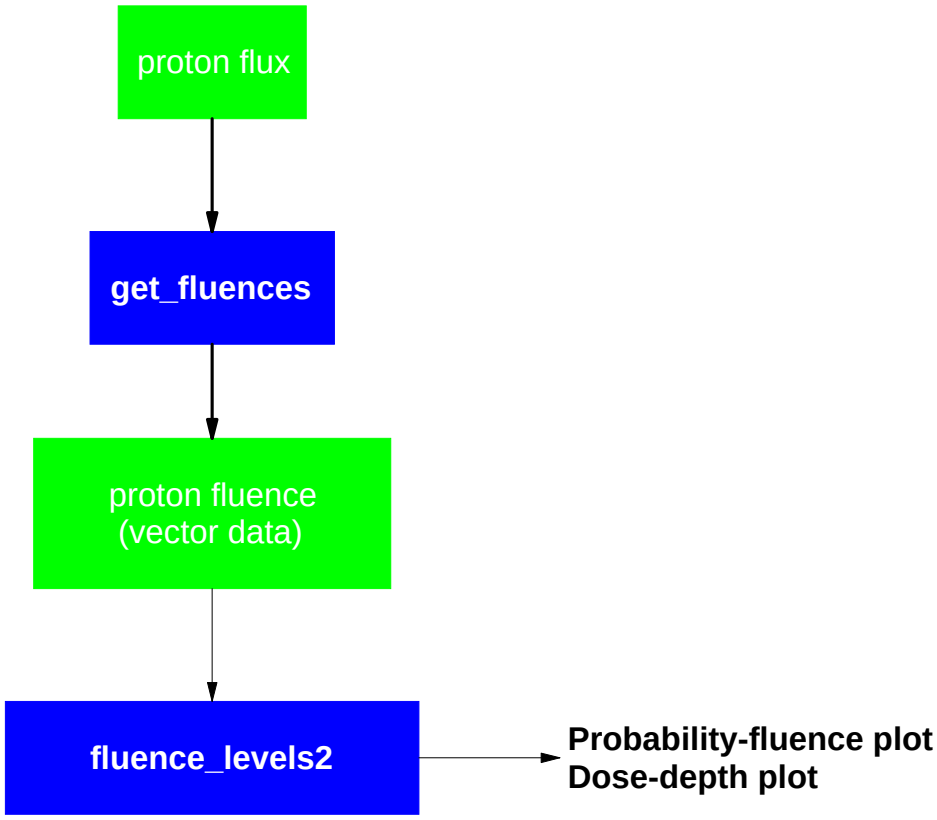

Figure 27. Tools for handling vector fluences

The data input to mah!get_fluences is a solar proton flux dataset. This is processed to derive a well-conditioned time series of the proton fluence accumulated from the start of the dataset and for each energy channel in the input data. This output can then be analysed statistically by mah!fluence_levels2 to derive two products:

- Probability-fluence curves for the range of energies in the data
- Dose-depth curves for a user-supplied range of exposure times

The conditioned time series of the proton fluence is also used as an input to spacecraft radiation analysis tools shown in Figure 28.

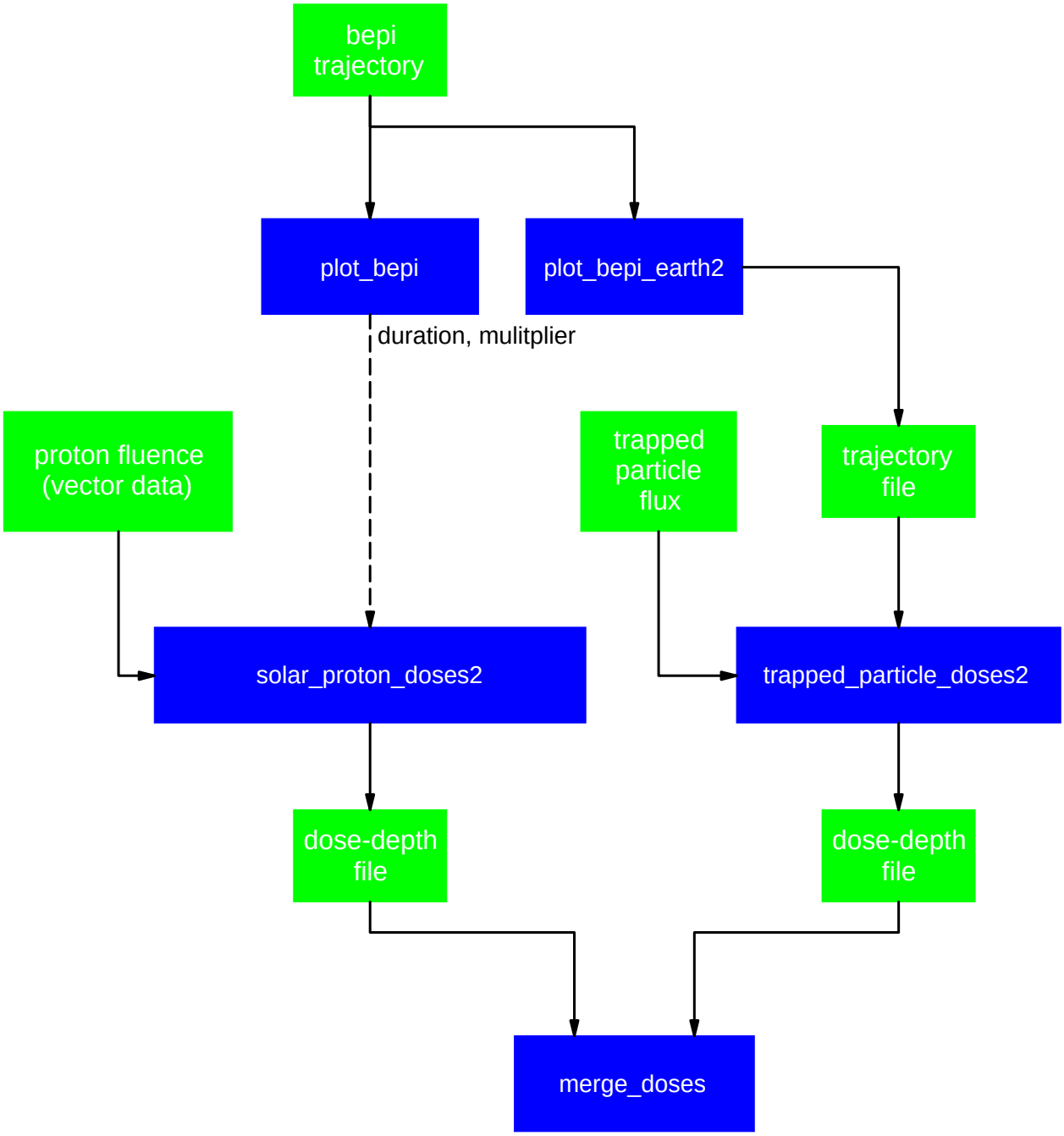

Figure 28. Data flow for spacecraft radiation analysis.

The primary data input for the tools shown in Figure 28 is the transfer orbit of Bepi-Colombo. This is plotted using mah!plot_bepi to allow comparison with plots supplied by ESA in conjunction with the orbit data. This tool also calculates two parameters needed as input to the solar proton dose tool – namely (a) the duration of the transfer orbit and (b) a multiplier or enhancement ratio C that adjusts for the changing heliospheric distance of Bepi ($C = \int 1/r(t)^2 \, dt$, where r(t) is the heliospheric distance of Bepi in AU).

The orbit data is also processed by the tool mah!plot_bepi_earth2. This calculates the position of Bepi with respect to the Earth – both following launch and during a fly-by about one year after

launch. This tool also constructs a pseudo trajectory that provides a concise representation of Bepi's flights through the Earth's radiation belts.

The radiation exposure for Bepi then calculated by two tools:

- The tool mah!solar_proton_doses2 calculates dose-depth curves for solar proton exposure at different probability levels. Its main input is the solar proton vector fluences as derived above using mah!get_fluences, but it also needs the mission duration and enhancement factors calculated by the mah!plot_bepi tool. It produces a dose-depth-probability dataset for solar proton exposure.
- The tool mah!trapped_particle_doses2 calculates dose-depth curves for trapped particle exposure at different probability levels. Its main inputs are the spacecraft paseudo-trajectory through the radiation belts and a dataset of trapped particle fluxes in the belts. The tool must be run twice. The first run takes trapped proton fluxes as input and produces a dose-depth-probability dataset for trapped proton exposure. The second run takes trapped electron fluxes as input and produces two dose-depth-probability datasets - for trapped electron exposure and for bremsstrahlung effects from those electrons.

The finaL tool is the merging tool (mah!merge_event_recs) which ingests a set of dose-depth-probability datasets and plots the dose-depth curve for each exposure type at one user-selected probability level (where the probability is that of the dose exceeding the plotted value). The tool checks the exposure types and plots one curve for each type.

Table 8. Top-level tools for WP302 plus their queries

Tool	Query	Function
mah!get_fluences	mah!Qget_fluences	Process proton data to remove spikes and derive time series of vector values of fluences (i.e. over a range of energies). Derived from mah!get_proton_data
mah!fluence_levels2	mah!Qfluence_levels2	Generate probability-fluence curves for a range of energies by direct statistical analysis of solar proton fluences over that range of energies.
mah!make_usoc_orbit	mah!Qmake_usoc_orbit	Calculate spacecraft trajectory given orbit elements in USOC format
mah!trapped_particle_doses2	mah!Qtrapped_particle_doses2	Calculate dose due to trapped protons or electrons
mah!solar_proton_doses2	mah!Qsolar_proton_doses2	Calculate dose due to solar protons
mah!merge_doses	mah!Qmerge_doses	To ingest, merge and plot dose data from four different streams - solar protons, trapped protons, electrons and bremstrahlung.
mah!plot_bepi	mah!Qplot_bepi	To plot Bepi's heliospheric orbit and calculate duration and adjustment factors for solar proton exposure
mah!plot_bepi_earth2	mah!Qplot_bepi_earth2a	To plot Bepi's geocentric trajectories following launch and during fly-by. And to construct pseudo-trajectory that concisely represents all Bepi crossings of the radiation belts.

Table 9. Parameters used by WP302 queries

Parameter	*Description*	*Recommended values*
mah!Qget_fluences		
proton_data	Dataset that contains the data to be searched for events. This should logically comprise a series of time-ordered records, each containing a time (epoch) field and the proton flux field(s).	Select "Dataset" from dropdown menu, then use second dropdown menu to select the name of the proton dataset to analyse
start_time	Start time of data search	Start time as CCSDS ASCII code A, i.e. yyyy-mm-ddThh:mm:ssZ
end_time	Stop time of data search	Stop time as CCSDS ASCII code A
data_source	String uniquely specifying the data format in use.	String describing the data format: a. GOES-I for GOES data in I format; b. IMP for IMP-J data
data_instance	Integer code identifying the spacecraft on which data were taken.	Spacecraft code as in Table 7
shift_factor	shift to be applied in median smoothing;	4
mah!Qfluence_levels2		
time_window	Time windows for which we calculate probability of exceeding fluence levels. Can enter multiple values separated by commas.	10, 100, 1000, 10000 (Four values of the window at steps of 10 in window size from 10 upwards)
window_si_rel	Units of time window as an SI_conversion string	3.6e+3>s for units of hours
scale_factor	Scale factor for sampling. We aim to sample the fluence time series with an average period = time window * scale factor	1.0
fluence_data	Name of dataset containing solar proton fluence data for range of energies (vector data)	Select "Dataset" from dropdown menu, then use second dropdown menu to select the name of the fluence dataset to analyse

Parameter	*Description*	*Recommended values*
mah!Qmake_usoc_orbit		
Orbit_data	Name of dataset holding elements in USOC style	Select "Dataset" from dropdown menu, then use second dropdown menu to select the name of the orbit data to analyse
Time_string	String that specifies start and end times, and step size, of calculation. Times are MJD and step size is in days. Values have Fortran format (18X,F7,7X,F7,7X, F5).	xxxxxxxxxxxxxxxxxx51566.5xxxxxxx51568.5xxxxxxx00.05
mah!Qtrapped_particle_doses2		
trajectory_data	Name of dataset holding the spacecraft trajectory in terms of time-tagged Cartesian position, L and B values.	Select "Dataset" from dropdown menu, then use second dropdown menu to select the name of the orbit data to analyse
flux_data	Name of dataset holding measurements of charged particle fluxes	Select "Dataset" from dropdown menu, then use second dropdown menu to select the name of the orbit data to analyse
start_time	Start time of data search	Start time as CCSDS ASCII code A, i.e. yyyy-mm-ddThh:mm:ssZ
end_time	Stop time of data search	Stop time as CCSDS ASCII code A
dataset_code	Integer code identifying the spacecraft on which data were taken.	Spacecraft code as in Table 7
mah!Qsolar_proton_doses2		
time_window	Duration for which dose is to be calculated	Real number as appropriate
window_si_ref	Units of time window as an SI_conversion string	3.15576e+7>s for units of years
scale_factor	Scale factor for fluences. We increase the sampled fluences by this factor to simulate the effect of changing heliospheric distance	Real number as appropriate
fluence_data	Name of dataset containing solar proton fluence data for range of energies (vector data)	Select "Dataset" from dropdown menu, then use second dropdown menu to select the name of the fluence dataset to analyse

Parameter	*Description*	*Recommended values*
mah!Qmerge_doses		
Dose1	Name of dataset containing solar proton dose-depth curves	Select "Dataset" from dropdown menu, then use second dropdown menu to select the name of the dose-depth dataset to analyse
Dose2	Name of dataset containing trapped protons dose-depth curves	
Dose3	Name of dataset containing trapped electron dose-depth curves	
Dose4	Name of dataset containing bremstrahlung dose-depth curves	
Prob_level	Probability level for which dose-depth curves should be plotted. This is the probability that the dose will exceed the plotted value, i.e. a 10% probability implies a 90% probability that the dose will be less than or equal to the plotted value.	percentage value, e.g. 10. Note that this is a free parameter. The tool will find, report and use the best match out of the probability levels in the input data.
mah!Qplot_bepi		
trajectory	Name of dataset holding Bepi's heliospheric trajectory in HAE coordinates for epoch 2000	mah!bepi_orbit_v2
mah!Qplot_bepi_earth2a		
trajectory	Name of dataset holding Bepi's heliospheric trajectory in HAE coordinates for epoch 2000	mah!bepi_orbit_v2
ds_indices	Indices of the points on the heliospheric trajectory that will form the pseudo-trajectory through the radiation belts.	NOT a formal parameter, but set internally by editing definition of array ds_indices within tool. Select indices by examination of geocentric coordinates listed in log of first run of tool, then re-run tool.

Table 10. Other important tools used in WP302

Name	Type	Description
mah!check_ds_code	Function	Checks for valid code and assigns a name to it
mah!get_ds_meta	Function	Get dataset-specific metadata
mah!get_ds_record	Function	Get items from a record in dataset-specific manner
mah!calc_xyz	Function	Calculate GEI Cartesian position of spacecraft given radial distance, true anomaly and orbit elements
mah!check_cycle_phase	Function	Determines if a given date lies in the solar maximum years.
mah!cluster_wrapper	Function	Maps the clustran_one call interface on to the system level convcoord interface
mah!dataset_lib	Function	Library of dataset specific codes for use in analysis of trapped particle datasets:
mah!dose_meta	Function	Build metadata for dose-depth curve data
mah!extract_si_con	Function	process an SI conversion string supplied in the format
mah!fluence_file_meta	Function	Build metadata for SEDAT fluence file format
mah!get_quantile	Function	Get quantiles of a distribution
mah!kepler	Function	Calculate radial distance and true anomaly given Keplerian elements and time
mah!make_orbit_pos	Function	Calculate GEI Cartesian position of spacecraft given orbit elements and time
mah!make_sd_control_structure	Function	Set up control data structure (sd2_control_data) used in the SEDAT interface to SHIELDOSE-2.
mah!make_sd_flux_structure	Function	Set up flux data structure (sd2_flux_data) used in the SEDAT interface to SHIELDOSE-2.
mah!make_sd_test_structures	Function	Populate control and flux data structures with reference input data for testing
mah!make_this_orbit	Function	Select the set of Keplerian elements applicable to given time
mah!print_test_dose_depth	Function	Print test results from SHIELDOSE-2.
mah!proton_data_instances	Function	Returns string with the name of a data instance given its code as in Table 7
mah!shieldose_sp_lib	Function	Library of routines to support use of Shieldose with solar proton data: a. setup_shieldose_sp - sets up Shieldose data structures for solar proton data b. get_diff_spec - converts integral fluences into differential fluences c. extend_diff_spec - extrapolates and interpolates differential fluences
mah!test_shielddose	Tool	Run Shieldose using the reference input data supplied with the Shieldose-2 source code and replicating the reference output data used in testing

		outside SEDAT
mah!time_lib	Functions	Library of tools for manipulating time values: • CDFepoch_CCSDS - function to convert CDF epoch to CCSDS A format • CCSDS_CDFepoch - function to convert CCSDS A format to CDF epoch • CDFepoch_MJD - function to convert CDF epoch to MJD • MJD_CDFepoch - function to convert MJD to CDF epoch
mah!valid_num	Function	Check if string is a valid IDL number – routine from SOHO library

14 Annex D - Compliance matrix

The table below shows the compliance between this report and the demonstration plan [R13]. The item numbers relate to items in the demonstration procedure section (4.2) of that plan and compliance section numbers relate to the sections of this document.

Item from Plan	Description	Compliance in section
Section 4.3.1 Plot heliospheric orbit		
1-4	Mission orbit	9.1
5	Solar proton exposure	9.3
Section 4.3.2 Plot geocentric orbit		
1-3	Position with respect to Earth – launch and fly-by	9.2
4-6	Pseudo-trajectory	9.4
Section 4.3.3 Doses from solar protons		
1-4	Get raw fluences	5.1 (also section 7 of [R1])
5-8	Calculate solar proton doses	5.2, 9.3
Section 4.3.4 Doses from trapped particles		
1-5	Trapped proton doses	6, 9.4
1-5	Trapped electron doses, including bremsstrahlung	7, 9.4
Section 4.3.5 Merging different dose types		
1-4	Merged doses	8, 9.5