AEROSPACE REPORT NO.
TOR-2016-01203

AE9/AP9 Internal Charging Kernel Utility—Beam-Slab Geometry Adapted to Hemispherical Shell for Aluminum Shields

March 3, 2016

T. P. O'Brien, James L. Roeder, and Mark D. Looper
Space Science Applications Laboratory
Physical Sciences Laboratories

Prepared for:

Space and Missile Systems Center
Air Force Space Command
483 N. Aviation Blvd.
El Segundo, CA 90245-2808

Contract No. FA8802-14-C-0001

Authorized by: Systems Planning, Engineering, and Quality

Abstract

The AE9/AP9 internal charging (IC) kernel utility is a Python application to generate kernel data files that AE9/AP9 can use to estimate electron current behind shielding. Each kernel contains all the information needed to convert from electron flux to transmitted current at various depths of shielding. The aim is to provide a rapid means of computing internal charging current analogous to the dose-depth calculations that are customarily provided by space environment radiation applications. The utility approximates the effects of shielding using a transmittance function that is scaled to the shielding as a fraction of the maximum range for each incident energy. The transmittance function is based on Monte Carlo test particle simulations of monoenergetic, normally incident beams incident on planar aluminum slabs. This simulation has been used for many years to design and evaluate laboratory testing. An approximation is made to scale this pre-existing simulation to the case of isotropic shielding. It is believe that this approximation is accurate to within about a factor of 2, and that accuracy is not a strong function of shielding depth. Thus, the utility allows for estimation of the benefit of adding shielding to mitigate internal charging. The kernel itself is an Extensible Markup Language (XML) file produced by the utility. This report describes algorithms used by the utility to generate the XML file, verification of the calculations, and limitations of the utility. Future versions of AE9/AP9 will be able to use such kernels to compute instantaneous and thereby worst-case, internal charging currents for use in orbit trade studies and preliminary design. We demonstrate the use of the kernel in an assessment of how much additional shielding is needed to protect a geostationary vehicle through a long spiral transfer orbit.

Acknowledgments

The authors acknowledge useful discussions with J. E. Mazur, as well as other colleagues at The Aerospace Corporation and on the AE9/AP9 team. IDL, the Interactive Data Language, is a registered trademark of Exelis, Inc.

Contents

Figures

1. Introduction

The AE9/AP9 kernel concept has been introduced in previous technical reports [*O'Brien and Kwan*, 2013; *O'Brien and Whelan*, 2015; *O'Brien*, 2015]. Briefly, the kernel provides a means of describing a linear radiation effect in a standardized data file so that the AE9/AP9 application can read it in at runtime and use it to transform model particle flux into radiation effects. Many radiation effects are linear in the incident flux, and so this allows AE9/AP9 to include many kinds of generic or custom effects calculations without modification of the application source. The kernel itself is an extensible markup language (XML) text file that contains a flux-to-effect transform matrix and the metadata needed to use that matrix. For kernels that account for shielding, AE9/AP9 will allow the kernel to express shielding in areal density (g/cm^2), and another XML file (a shielding library) will enable the application to convert to equivalent shielding thickness for a variety of materials. For some effects, this shielding equivalence via equal areal density is a very good approximation. For internal charging, it may not be—that remains a research question.

The internal charging current simulation represents only one specific laboratory test geometry, which we have adapted to approximate on orbit conditions. The lab and kernel configurations are illustrated in Figure 1. In the lab simulation, all electrons are normally incident, and all electrons exiting the planar shield are counted, regardless of their angle of exit. In the kernel application of that data, electrons are normally incident on a hemispherical shell shield, and they are assumed to strike an infinitesimal target at the center of the shell regardless of their angle or location of exit from the shell. The error associated with this approximation is thought to be no more than a factor of 2. The two main sources

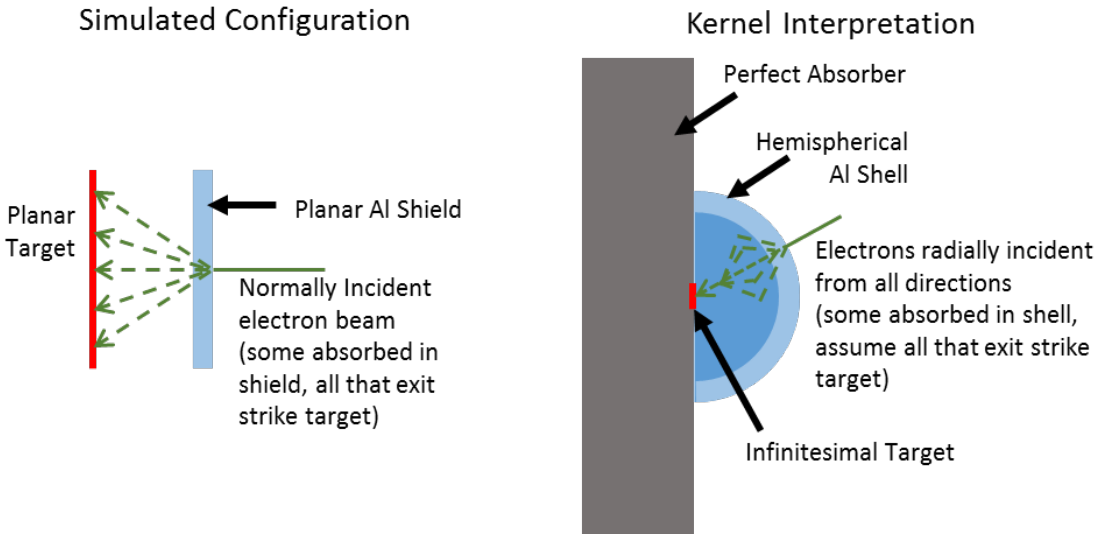

Figure 1. Simulated and interpreted geometries. In the simulated geometry (left) electrons are only normally incident, and all electrons exiting the shield strike the target. In the kernel interpretation (right), meant to represent on orbit geometry, only normally incident electrons are considered, but all particles exiting the shield are assumed to reach the target, regardless of their angle or location of exit.

of error are: for every real particle exiting the shield and not striking the target, there is approximately one replacement particle that is not radially incident but does strike the shield, and particles striking the shield at low angles of incidence have a longer path through the shield. An ongoing project will upgrade this kernel to account for more realistic spherical shielding geometry with isotropic flux and different shielding materials besides Al. The kernel is adapted from a shielded current calculation that has been implemented in IDL (Interactive Data Language) and used for various analyses [see, e.g., *Fennell et al.*, 2002].

In the following sections, we will describe the equations used for the shielded current calculation. We will then compare the results to example values calculated from the original IDL routine. Finally, we will demonstrate the application of the kernel to determine how much additional shielding is needed to protect a geostationary satellite during a long spiral transfer orbit.

All of the relevant calculations and the ability to write the AE9/AP9 standard XML file format are provided in a Python utility with a command line interface (see the Appendix) and a programmatic interface (documented in the source code itself following Python conventions).

2. The Transmitted Current

The following assumptions are made to calculate the effects of shielding:

- The shield is a hemispherical shell of aluminum
- The target is an infinitesimal planar area at the center of the shield
- No electrons arrive from the backside either directly or through scattering
- Electrons are only normally incident on the shield
- All electrons that emerge from the shield strike the target as if originating along their initial ray path

The latter two assumptions are necessary to interpret the beam-slab simulation in a space-like geometry. The original simulation was performed using EGSnrc [Kawrakow, 2000]. In the lab geometry, the transmitted fraction was computed for several incident energies through several Al slab shields, and the transmitted fraction was fit with analytical expressions.

The AE9 model will provide the omnidirectional electron flux *j(E)* at incident energy *E*. Assuming that flux is isotropic, with directional intensity $\hat{\jmath}(E) = j(E)/4\pi$, the current transmitted along any radial path through the spherical shield is:

$$\hat{J}(T) = e \int \tau(E;T)\hat{\jmath}(E)dE, \tag{1}$$

where *e* is the electron charge, and $\tau(E;T)$ is the transmitted fraction. The electron current is incident on the target area over only a hemisphere, and there is a cosine areal projection effect for impinging on a planar target. Thus, the total current integrating over a hemisphere of polar angles (θ) and all azimuths (ϕ) is:

$$J(T) = \int_0^{2\pi} \int_0^{\pi/2} \hat{J}(T) \cos\theta \, d\theta \, d\phi = \pi \hat{J}(T). \tag{2}$$

Combining Eqs. (1) and (2) and substituting in $j(E)/4\pi$ for $\hat{\jmath}(E)$, we have:

$$J(T) = \frac{e}{4} \int \tau(E;T) j(E) dE. \tag{3}$$

Thus, the transmitted current in our approximation is a linear operation (a convolution) of the incident flux, where the coefficients of the convolution depend on incident energy and shielding depth.

The transmitted fraction itself can be approximated by:

$$\tau(E;T) = \tau\big(N(E;T)\big) = \frac{1}{\cosh(0.362665 + 3.29776N - 3.51115N^2 + 4.95554N^3)}. \tag{4}$$

Here, *N(E;T)* is the shielding *T* normalized to be the maximum range *R(E)* for an electron with energy *E* (in Al):

$$N(E;T) = T/R(E). \tag{5}$$

The maximum range is given by [*Katz and Penfold,* 1952] as:

$$R(E) = \begin{cases} 0.412E^{1.265-0.0954\ln(E)} & 0.01 \le E \le 2.5 \text{ MeV} \\ 0.530E - 0.106 & 2.5 < E \le 20 \text{ MeV} \end{cases} \tag{6}$$

Taken together, Eqs. (3)–(6) define the kernel transform itself. In the next section, we describe how the integral is represented numerically as a matrix-vector operation.

3. Calculating the Kernel

The IC kernel itself evaluates Eq. (3) numerically for a variety of shielding depths. The user may specify both the incident energy grid (with N points) and the output shielding grid (with M points). The user may specify linearly spaced grids, logarithmically spaced grids, a logarithmic grid that also includes zero, or even a simple list of values to use as a grid. The Al shielding grid may also be provided in length units of cm, mm, or mils, rather than areal density, and the utility will convert to areal density using an Al mass density of 2.7 g/cm^3.

At a high level, the kernel can be thought of as the matrix $\underline{\underline{A}}$ in a matrix-vector equation:

$$\vec{J} = \underline{\underline{A}}\vec{j}. \tag{7}$$

In this equation, the transmitted current is specified on a grid of depths T_i:

$$J_i = J(T_i). \tag{8}$$

The incident flux is given on a grid of energies E_k:

$$j_k = j(E_k). \tag{9}$$

The kernel matrix $\underline{\underline{A}}$ is then built to achieve:

$$J(T_i) = \frac{e}{4}\int \tau(E;T_i)j(E)dE \approx \sum_k A_{ik}j_k. \tag{10}$$

The bounds of the integration are determined by the incident flux supplied by the user, and are also constrained by Eq. (6) (the utility generates an error if the incident energy grid extends above or below the limits of the range formula).

Equation (10) can be represented numerically as:

$$\frac{e}{4}\int \tau(E;T_i)j(E)dE \approx \frac{e}{4}\sum_k \tau(E_k;T_i)j_k\Delta E_k, \tag{11}$$

where ΔE_k is a trapezoidal integration weight:

$$\Delta E_k = \begin{cases} \frac{E_2 - E_1}{2} & k = 1 \\ \frac{E_N - E_{N-1}}{2} & k = N \\ \frac{E_{k+1} - E_{k-1}}{2} & \text{otherwise} \end{cases} \tag{12}$$

Therefore,

$$A_{ik} = \frac{e}{4}\tau(E_k; T_i)\Delta E_k. \tag{13}$$

This completes the definition of the kernel matrix $\underline{\underline{A}}$, which can be precomputed for user-specified specified depth and energy grids, and then used for any incident flux supplied by AE9. The AE9/AP9 application addresses interpolation between the model energy grid and the kernel energy grid, conversion (via areal density) between shielding materials, and interpolation between the kernel shielding grid and shielding parameters specified by the user when running the AE9/AP9 application (i.e., after the kernel XML file is generated by the Python utility).

4. Verification

Because of the mismatch between the simulated geometry and kernel geometry, we cannot exactly validate the kernel calculation. However, we can *verify* that it matches the original IDL calculation. We do this in two steps. First, we verify that the transmittance calculation in the kernel matches the transmittance computed from the original simulation of the lab geometry. Second, we perform some point comparisons between the IDL and Python codes.

First, we verify the transmittance. Figure 2 shows a side-by-side comparison of the tabulated transmittance from the EGSnrc simulation and the computed values from the Python utility. Noting that the ripples in the simulation curves are statistical fluctuations in the simulation statistics, the two figures are in very good agreement.

Next, we consider transmitted current versus depth for a worst-case electron spectrum at geostationary orbit [*Fennell et al.*, 2000]:

$$j(E) = 3.67\times10^{7}\exp[-1.57E] \quad \#/\mathrm{cm}^2/\mathrm{s/MeV} \tag{14}$$

for E in MeV. Figure 3 shows the output of the two tools along with "safe levels" from three different sources: NASA-HDBK-4002A, *Bodeau et al.* [2005], and *Fennell et al.* [2000]. The agreement in transmitted current is very good, and we can consider the Python kernel utility to be an accurate implementation of the original IDL calculation.

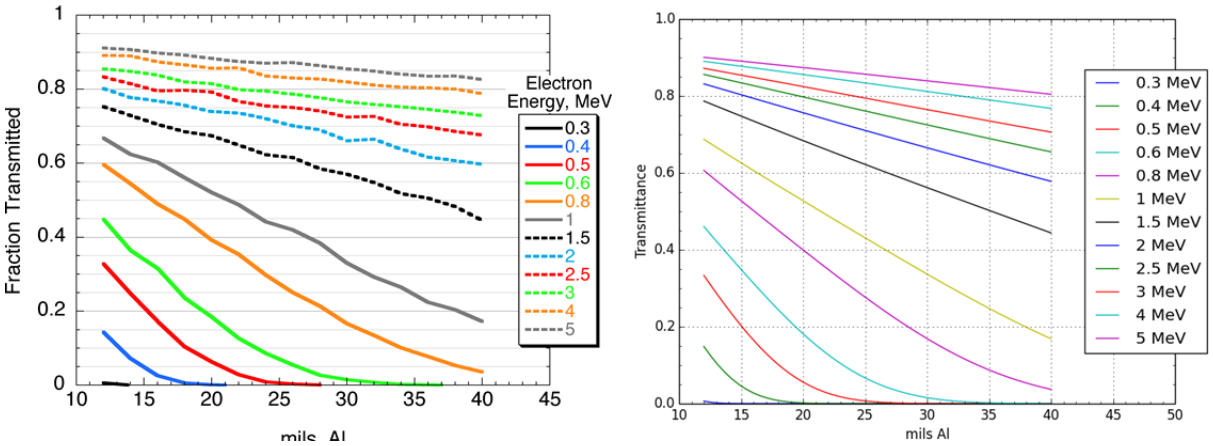

Figure 2. Transmittance from the EGSnrc simulation (left) and from the Python utility (right).

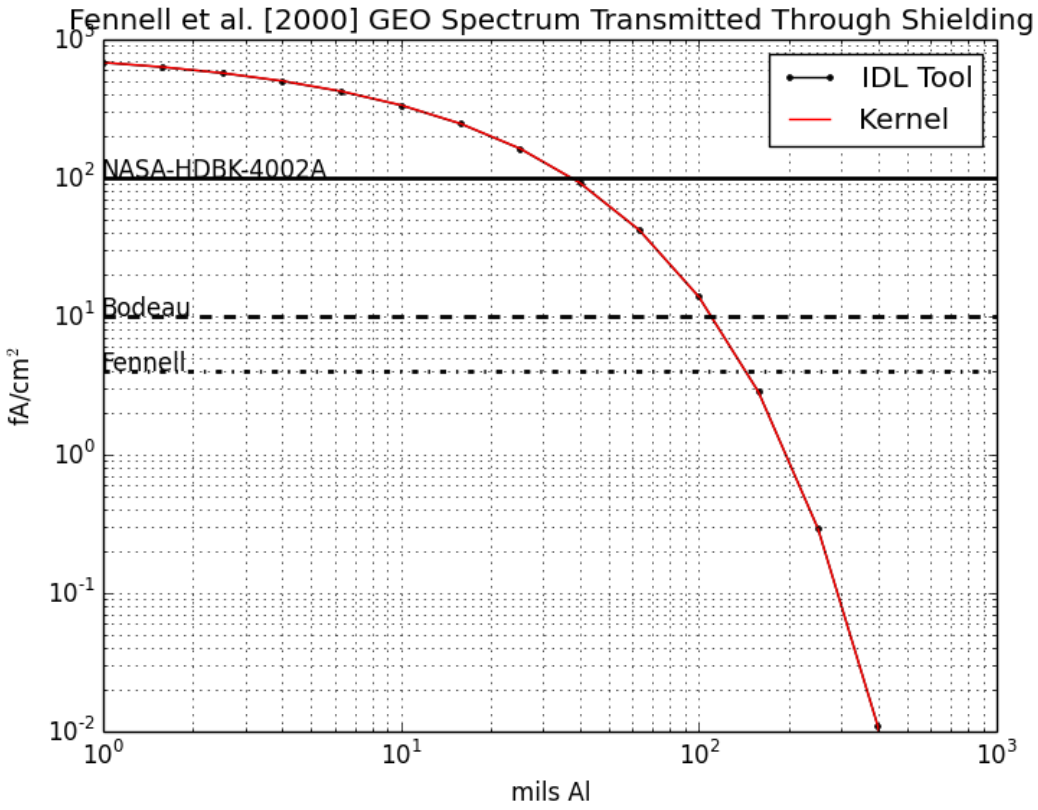

Figure 3. Comparison of IDL and Kernel calculation of internal charging current for a worst-case spectrum at geostationary orbit. Multiple "safe level" reference values are provided.

5. Demonstration

To demonstrate the use of the kernel, we consider a spiral transfer to geostationary orbit compared to direct insertion to geostationary. The spiral transfer is orbit GEO3 from *Kwan and O'Brien* [2015], which takes 267 days to reach geostationary orbit. The direct insertion spends only 1.5 orbits in an elliptical transfer orbit. In each case, the vehicle is expected to operate on station for 15 years.

AE9 [*Ginet et al.*, 2013] provides Monte Carlo scenarios that include the radiation belt dynamics that cause internal charging. For each trajectory, we simulate 40 scenarios with AE9 and compute the charging current versus depth at each time step for each scenario using the kernel. For each scenario, we track the highest 24-hour running average charging current at each depth. We then compute the 95th percentile worst-case charging current over the scenarios. Thus, we compute a charging current that has a 95% chance of never being exceeded at any time over the entire mission.

Figure 4 shows the transmitted current versus shielding for the spiral transfer (GEO3) case and the direct transfer case. The plot also shows (green curve) the spiral transfer case shifted left by 60 mils Al. We see that over the range of "safe levels," the added 60 mils if Al brings the current in the spiral case down to the safe levels, regardless of what definition of safe was used. For a one-meter-cube satellite, that extra 60 mils Al on all faces amounts to about 25 kg of extra mass.

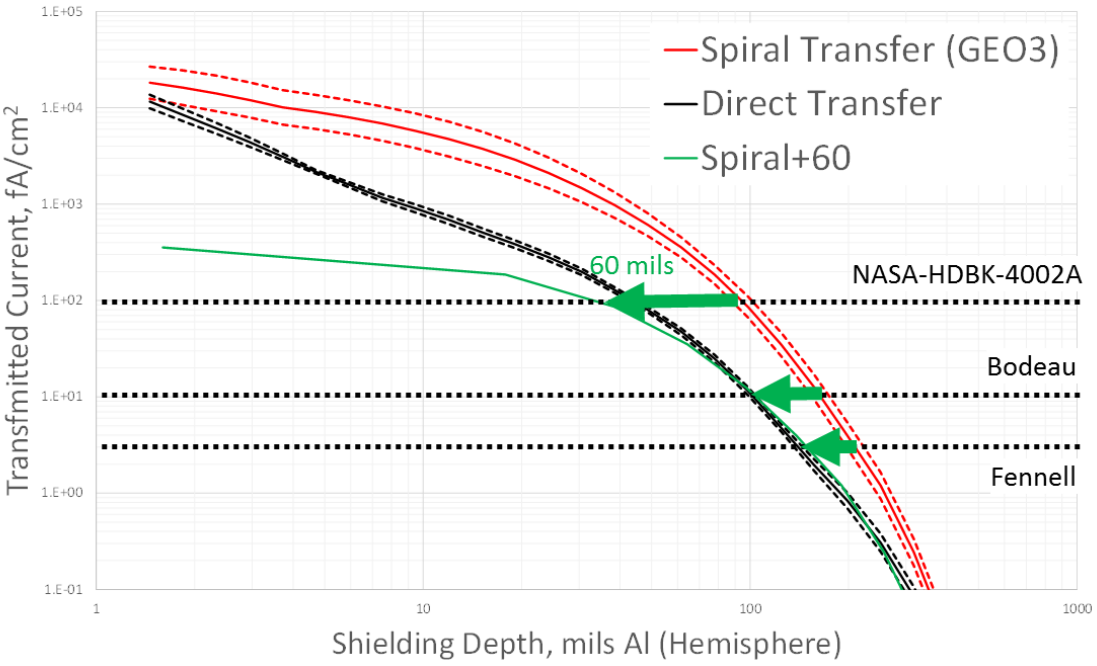

Figure 4. Internal charging current versus depth of shielding for geostationary orbit. The "Spiral+60" curve shifts the spiral transfer curve left 60 mils Al, showing that it takes approximately 60 mils Al extra shielding to achieve the same transmitted current over the typical range of "safe" current levels.

6. Summary

We have described, verified, and demonstrated a utility for generating internal charging kernels for use with AE9/AP9. The kernel computes transmitted electron current behind shielding. It makes use of a pre-existing simulation designed for beam tests to approximate hemispherical aluminum shielding representative of on-orbit satellite configurations. This approximation may carry an error as large as a factor of two. We have verified that the kernel matches an existing IDL tool that performs the same calculation.

The kernels will be used by the AE9/AP9 application as developer- or user-provided XML files. Using an AE9/AP9 prototype, we have demonstrated how the kernel can be used to compute the 95% worst-case charging current as a function of shielding depth. We applied this technique to estimate the additional shielding needed to protect a geostationary satellite with a long spiral transfer orbit. A future production version of AE9/AP9 will incorporate the ability to use these kernels.

We have begun a study to perform new simulations of space-like shielding configurations to eliminate the geometric uncertainty in the kernel presented here. That study will also address the appropriateness of using areal mass density to compute the equivalent amount of aluminum shielding for other shielding materials.

7. References

Bodeau, M. (2005), "Going Beyond Anomalies to Engineering Corrective Action, New IESD Guidelines Derived From A Root-Cause Investigation," Presented at the 2005 Space Environmental Effects Working Group, The Aerospace Corp., El Segundo, CA.

Fennell, J. F., H. C. Koons, M. W. Chen, and J. B. Blake, "Internal charging: A preliminary environmental specification for satellites," *IEEE Trans. Plasma Sci.* **28**(6), 2029-2036, 2000.

Fennell, J. F., J. L. Roeder, H. C. Koons, "Substorms and magnetic storms from the satellite charging perspective," *Proceedings of the COSPAR Colloquium*, COSPAR Colloquia Series *12*, 163-173, 2002.

Ginet., G. P., T. P. O'Brien, S. L. Huston, W. R. Johnston, T. B. Guild, R. Friedel, C. D. Lindstrom, C. J. Roth, P. Whelan, R. A. Quinn, D. Madden, S. Morley, and Y. J. Su, "AE9, AP9 and SPM: New models for specifying the trapped energetic particle and space plasma environment," *Space Sci. Rev.,* 2013, doi:10.1007/s11214-013-9964-y.

Katz, L., and A. S. Penfold, "Range-energy relations for electrons and the determination of beta-ray end-point energies by absorption," *Rev. Modern Phys.* **24**(1), 28-44, 1952.

Kawrakow, I., "Accurate condensed history Monte Carlo simulation of electron transport. I. EGSnrc, the new EGS4 version," *Med Phys.* **27**: 485-498, doi: 10.1118/1.598917 (2000)

Kwan, B. P., and T. P. O'Brien, "Using static percentiles of AE9/AP9 to approximate dynamic Monte Carlo runs for radiation analysis of spiral transfer orbits," *EEE Trans. Nucl. Sci*. **62**(3), 1357-61, 2015. doi:10.1109/TNS.2015.2423235

NASA-HDBK-4002A, *Mitigating in-space charging effects – a guideline*, National Aeronautics and Space Administration, Washington, DC, 2011.

O'Brien, T. P., *AE9/AP9 Proton Single-Event Effect Kernel Utility*, TOR-2015-02707, The Aerospace Corporation, El Segundo, CA, 2015.

O'Brien, T. P., and B. P. Kwan, *Using pre-computed kernels to accelerate effects calculations for AE9/AP9: A displacement damage example*, TOR-2013-00529, The Aerospace Corporation, El Segundo, CA, 2013.

O'Brien, T. P. and Whelan, P., *Specification for radiation effects kernels for use with AE9/AP9*, ATR-2015-02436, The Aerospace Corporation, El Segundo, CA, 2015.

Appendix—Internal Charging Utility Command-Line Interface

The AE9/AP9 internal charging utility creates an XML kernel file for use with AE9/AP9.

Options are available to specify the shielding grid and the electron energy grid. The utility can be invoked on the command line as:

python ae9ap9_ic_kernel.py *arguments*,

where *`arguments`* represents one or more user-supplied options.

Special arguments:

`--help` print this help message

`--test` run a predefined test

Normally required command line arguments (user supplied-values in *italics*):

Specifying the kernel Tag:

`--Tag=`*`tag`*

The kernel tag cannot have spaces or special characters.

Exactly one tag option is required. The tag is used by AE9/AP9 to generate filenames when saving kernel outputs.

Optional command line arguments:

Specifying the output file name (optional):

`--xmlfile=`*`filename`*

The filename can be a local file or it can include a full path as well.

If the filename is not specified, then one is created by appending ".xml" to the *tag* .

Specifying the shielding grid (optional):

`--Tgrid=`*`first:last/N`* *first* to *last* in *N* evenly spaced steps

`--Tgrid=`*`first:last/N`*`/log` *first* to *last* in *N* logarithmically spaced steps

`--Tgrid=`*`first:last/N`*`/0log` zero followed by *first* to *last* in *N*-1 logarithmic steps

`--Tgrid=`*`first:step:last`* *first* to *last* with specified *step* size (*last* may not be included)

`--Tgrid=`*`first,second,third,...,last`* complete list of grid points

At most one `--Tgrid` option is allowed

The default is `--Tgrid=0.01:7/30/0log`

Specifying the shielding unit (optional):

`--Tunit=milsAl` mils Al

`--Tunit=cmAl` cm Al

`--Tunit=mmAl` mm Al

`--Tunit=g/cm2` g/cm^2 areal density (default)

At most one `--Tunit` option is allowed

The default is `--Tunit=g/cm2`

Specifying the energy grid in MeV (optional):

`--Egrid=`*`first:last/N`* first to last in N evenly spaced steps

`--Egrid=`*`first:last/N`*`/log` first to last in N logarithmically spaced steps

`--Egrid=`*`first:step:last`* first to last with specified step size (last may not be included)

`--Egrid=`*`first,second,third,...,last`* complete list of grid points

Arguments follow same format as `--Tgrid`, except there is no `0log` option

At most one `--Egrid` option is allowed

The default is `--Egrid=0.01:7/30/log`