AEROSPACE REPORT NO.
ATR-2016-01756

Developing AE9/AP9 Kernels from Third-Party Effects Codes

May 13, 2016

T. P. O'Brien
Space Science Applications Laboratory
Physical Sciences Laboratories

Prepared for:

Sr. Vice President
Engineering and Technology Group

Authorized by: Engineering and Technology Group

Abstract

The AE9/AP9 project is developing the capability to capture linear radiation effects as "kernels" that can be used to transform particle flux into effects within the AE9/AP9 statistical and dynamics simulations. These kernels are represented as data files that the AE9/AP9 application can read and apply during the mission simulation. One of the challenges in developing kernels for established third-party effects calculations is that those calculations do not expose all the raw numerical data needed to generate the kernel data file. In this report, we describe a method to obtain the kernel data from an existing third-party application: The method consists of running the third-party application for many test particle flux spectra, and then deriving the implied kernel transform matrix from the results of those runs. We demonstrate this method using the ShielDose2 program to generate equivalent AE9/AP9 kernels for proton and electron dose in slab, semi-infinite medium, and spherical shielding geometries for silicon targets behind aluminum shielding.

Acknowledgments

The author acknowledges useful discussions with colleagues at The Aerospace Corporation and on the AE9/AP9 team. The version of ShielDose2 used in this report is provided by the International Radiation Belt Environment Model Library (IRBEM-LIB, irbem.sf.net). All trademarks and service marks are the property of their respective owners.

Contents

Figures

1. Introduction

An AE9/AP9 kernel transforms particle flux or fluence *j(E)* as a function of energy *E* into some desired radiation effect such as dose *d(T)* vs depth *T* [*O'Brien and Kwan*, 2013; *O'Brien and Whelan*, 2015]. Because the statistical operations one wishes to perform on AE9/AP9 results are often nonlinear (e.g., percentiles) it is necessary to convert from particle flux to radiation effect before computing such statistics. Therefore, it is desirable for AE9/AP9 to be able to compute the effects for arbitrary effects so that its statistical machinery can be used on effects as well as fluxes. Kernels provide just such a capability for effects that are linear functions of the incident flux. Although not all kernels have a dependent variable like depth, the following treatment addresses the more general case when there is such a dependent variable. The kernel approximates the convolution integral:

$$d(T) = \int G(E;T)j(E)dE. \tag{1}$$

In practice, this integral is represented by the kernel as a matrix-vector transform:

$$\vec{d} = \underline{\underline{A}}\vec{j} \tag{2}$$

where

$$d_i = d(T_i) \tag{3}$$

$$A_{ik} = G(E_k;T_i)\Delta E_k \tag{4}$$

$$j_k = j(E_k) \tag{5}$$

Developing a kernel usually amounts to determining the transform matrix $\underline{\underline{A}}$ from some kind of physics calculation. However, in some cases, we do not have access to the underlying calculation; we only have access to a third-party application that performs it. In such a case, we can infer the transform by running a set of test cases through the application. There are two general approaches: differential and integral.

The differential approach uses numerical derivatives of the results of the third-party application,

$$A_{ik} = \frac{\partial d_i}{\partial j_k} \approx \frac{\Delta d_i}{\Delta j_k}. \tag{6}$$

In this approach, the third-party application is called twice for each energy channel, slightly perturbing the flux in that channel either positively or negatively to obtain the numerical derivative. Since

essentially all radiation effects are positive definite, all values A_{ik} must be non-negative; any negative values are set to zero.

In the integral approach, we run the third-party application many times and solve a linear system of equations to obtain $\underline{\underline{A}}$. If we denote each test case with the superscript $^{(m)}$, then we have:

$$\vec{d}^{(m)} = \underline{\underline{A}}\vec{j}^{(m)} \tag{7}$$

Performing many tests with the third-party application, we can assemble the matrices $\underline{\underline{D}}$ and $\underline{\underline{J}}$:

$$\underline{\underline{D}} = [\vec{d}^{(1)} \quad \vec{d}^{(2)} \quad \ldots] \tag{8}$$

$$\underline{\underline{J}} = [\vec{j}^{(1)} \quad \vec{j}^{(2)} \quad \cdots] \tag{9}$$

The test spectra $\vec{j}^{(m)}$ can either be generated like a large bowtie analysis (i.e., ad hoc spectra of convenient analytical forms like power laws or exponentials), or the test spectra can form a basis set, e.g., many sigmoids or Gaussians. Either approach can be problematic because in order for $\underline{\underline{J}}\underline{\underline{J}}^T$ to not be nearly singular, number of trials must be very large, or some of the test spectra must have sharp gradients. If we run M trials, such that M is much larger than the number of entries in the energy grid, then we can construct an over-determined linear equation for $\underline{\underline{A}}$:

$$\underline{\underline{D}} = \underline{\underline{A}}\underline{\underline{J}} \tag{10}$$

We can solve this equation in a least-squares sense:

$$\underline{\underline{A}} = \underline{\underline{D}}\underline{\underline{J}}^T \left(\underline{\underline{J}}\underline{\underline{J}}^T\right)^{-1}. \tag{11}$$

The inversion may lead to some (hopefully small) negative entries in $\underline{\underline{A}}$. A simple solution is to set those negative entries to zero. Alternatively, Eq. (11) can be solved via quadratic programming to force all the entries in $\underline{\underline{A}}$ to be positive. In this case, every row of $\underline{\underline{A}}$ can be obtained in a separate quadratic programming optimization that minimizes the error in a least-squares sense while only allowing positive solutions. Quadratic programming routines are available in Matlab® (quadprog), IDL® (IMSL_QUADPROG), and Python (CVXOPT). A quadratic programming problem is usually stated as minimizing

$$\frac{1}{2}\vec{x}^T\underline{\underline{H}}\vec{x} + \vec{f}^T\vec{x} \tag{12}$$

subject to the constraints:

$$\underline{\underline{B}}\vec{x} \leq \vec{b} \tag{13}$$

For our problem, to solve for the i^{th} row of $\underline{\underline{A}}$, we have:

$$x_k = A_{ik} \tag{14}$$

$$\underline{\underline{H}} = \underline{\underline{J}}\underline{\underline{J}}^T \tag{15}$$

$$f_k = -\sum_m j_k^m d_i^m \tag{16}$$

$$\underline{\underline{B}} = -\underline{\underline{I}} \tag{17}$$

$$\vec{b} = 0 \tag{18}$$

However one obtains $\underline{\underline{A}}$, one can use $\underline{\underline{A}}$ and $\underline{\underline{J}}$ to estimate $\underline{\underline{D}}$, thereby determining how well the kernel matrix represents the third-party calculation. Experimentation with the integral approach has shown it to be much more computationally intensive and noisy than the differential approach. Therefore, the differential approach is highly recommended.

The $\underline{\underline{A}}$ computed from the differential or integral method will not necessarily perfectly reproduce every run of the third-party program because even for linear effects, such programs may also introduce nonlinearities, such as interpolation involving log flux or log T. The effects of these nonlinearities can be minimized with sufficiently dense energy and T grids.

2. Demonstration: ShielDose2

To demonstrate the extraction of kernels from a third-party application, we build dose versus depth kernels from the ShielDose2 application [*Seltzer*, 1994]. ShielDose2 computes dose in silicon versus depth for input proton and electron spectra, separating solar and trapped protons in the input and output, and separating electron and bremsstrahlung dose in the output. For our purposes, we need not distinguish between solar and trapped protons, and we sum electron and bremsstrahlung dose. ShielDose2 also provides three geometries, to which we add a fourth (full sphere):

- Dose in semi-infinite aluminum medium
- Dose at transmission surface of finite aluminum slab shields
- Half dose at center of aluminum spheres
- Full dose at center of aluminum spheres

The full sphere geometry is simply twice the half sphere geometry. For protons, the first two geometries provide the same dose, but we create kernels for both in order to allow matching kernels for both protons and electrons. Each kernel treats one shielding geometry and one species, providing a total of eight kernels.

Our kernels convert AE9 or AP9 fluence output into dose versus depth on a grid that spans the same range of depths as the ShielDose2 implementation provided with the AE9/AP9 application (0.03 to 30 g/cm^2). However, we work in mils Al, the more common unit used in the USA, so the depth range spans 4.375 to 4375 mils. We provide 30 depth points, logarithmically spaced between the limits.

To obtain the kernels, we use the differential method. While it is possible to call ShielDose2 once for each input spectrum and have it output the proton or electron dose at each depth, instead, we call ShielDose2 once for each target depth, supplying only 5 depths, 0.8, 0.9, 1.0, 1.1, and 1.2 times the target depth of choice. We determined that this approach gives output doses that are less dependent on the choice of depths, presumably because of interpolations being used inside ShielDose2.

Both our electron and proton spectra are flat with a fluence of 1 #/cm^2/MeV. For electrons, we use 30 channels logarithmically spaced from 40 keV to 10 MeV. For protons, we use 299 channels logarithmically spaced from 100 keV to 2 GeV (because of the Bragg peak in the energy deposit at end of range for protons, a larger number of channels is required, and 299 appears to be the maximum allowed by ShielDose2). To perturb the spectrum for each channel, we supply a fluence of 0.9 in the channel, while leaving all other channels 1.0, and obtain the dose output, and repeat for a fluence of 1.1 in the channel. We then compute A_{ik} according to Eq. (6).

Figure 1 shows the ShielDose2 and kernel dose for the three ShielDose2 geometries for our flat electron spectrum and our set of depths. The error metrics are based on the natural log dose, except for the max error (max|Err|), which is simply in relative terms. For all three error metrics, the max error is about 5%, which is comparable to the error from supplying different depth lists to ShielDose2 for the same input spectrum. The typical (RMSE) and average (<Err>) errors are less than 1%.

Figure 2 shows the ShielDose2 and kernel dose for the three ShielDose2 geometries for our flat proton spectrum and our set of depths. For all three error metrics, the max is about 0.1%, which is smaller than the error from supplying different depth lists to ShielDose2.

Figure 3 compares the kernel output to a typical ShielDose2 call for an electron spectrum that is exponential with a 100-keV e-folding energy, $j(E) = \exp(-E/0.1)$, and for a proton spectrum that is exponential with a 20-MeV e-folding energy, $j(E) = 10^{-9}\exp(-E/20)$. The relative intensities are scaled so that the proton dose traces are well below the electron dose. The energy unit is MeV and the fluence unit is #/cm^2/MeV. Compared to the flat spectra used to develop the kernels, this more realistic case has somewhat larger errors. Still, the errors are never more than 8%, which is the same magnitude as the discrepancy one sees when running ShielDose2 with different lists of depths.

The eight kernels are captured in extensible markup files (XML) in accordance with the AE9/AP9 Kernel XML specification [*O'Brien and Whelan*, 2015]. The XML files can, in turn, be used by AE9/AP9.

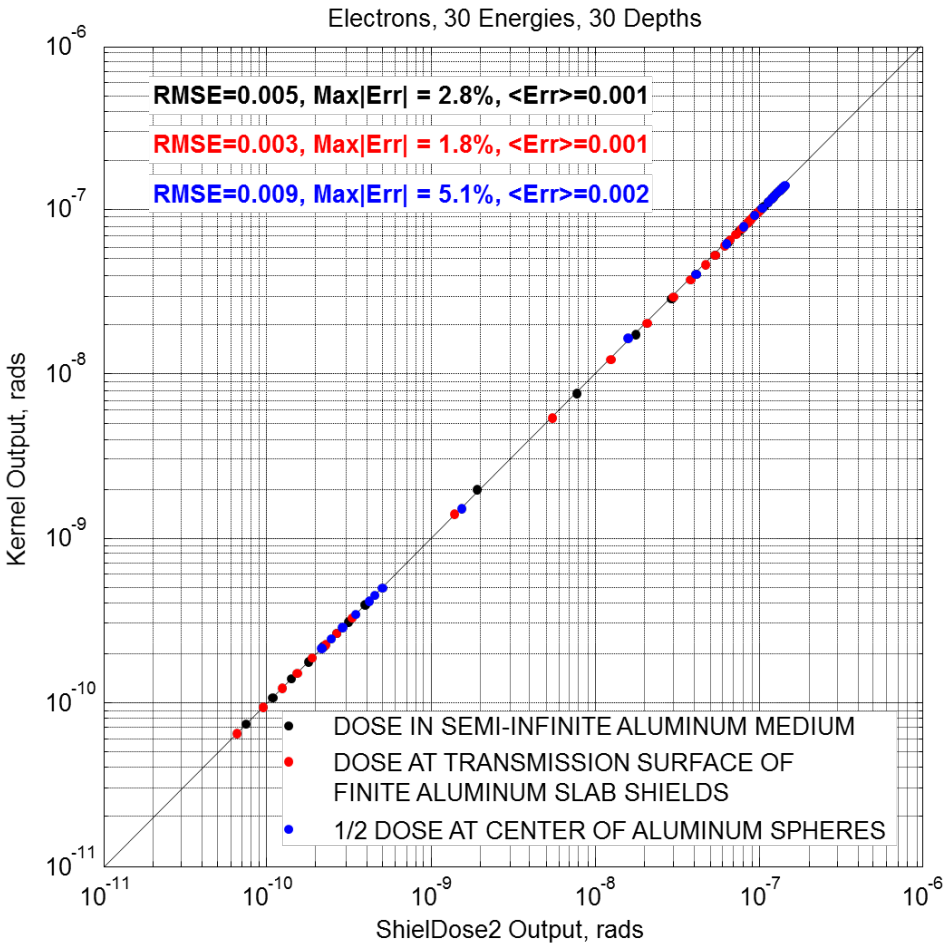

Figure 1. Kernel output versus ShielDose2 for flat electron spectrum for three shielding geometries.

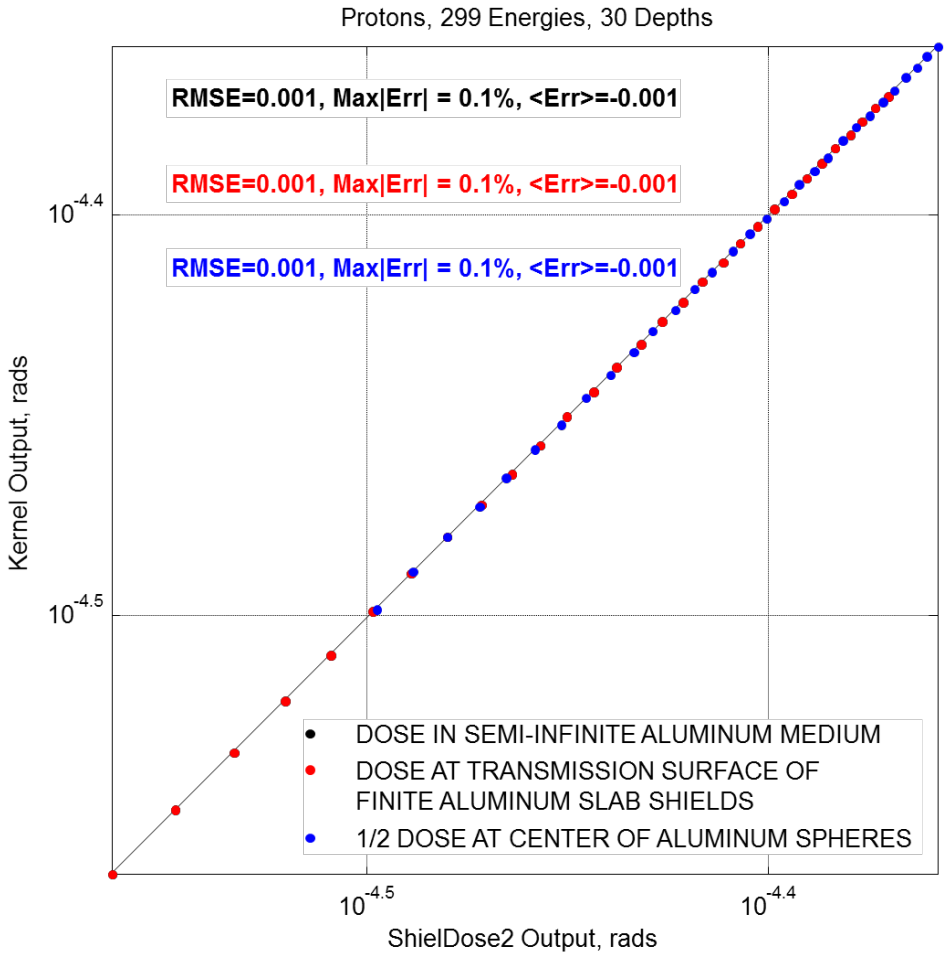

Figure 2. Kernel output versus ShielDose2 for flat proton spectrum for three shielding geometries.

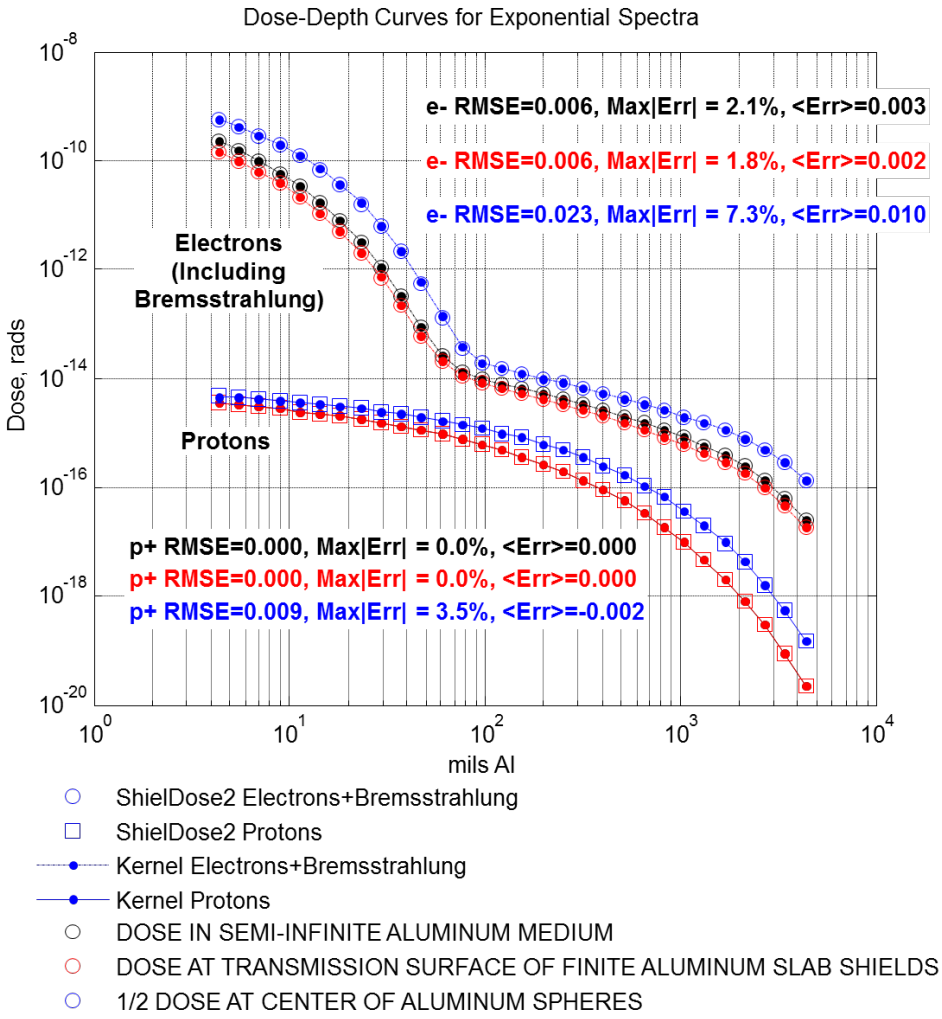

Figure 3. ShielDose2 and Kernel dose-depth curves for exponential spectra.

3. Summary

We have described two methods for obtaining the AE9/AP9 kernel transform matrix from a third-party application. The differential method involves computing numerical derivatives of the output of the application, while the integral method involves running many test spectra through the application and performing matrix operations to solve for the transform matrix. We have demonstrated the preferred differential method on the ShileDose2 application, which computes dose versus shielding depth for input proton and electron spectra.

We suggest that the differential method is generally easier to execute than the integral method because it is difficult to generate a robust set of test spectra. We have also learned that it may be necessary to run the effects code in a special way (e.g., running ShielDose2 with 5 nearby depths for each desired output depth) to obtain the most accurate outputs. We explained how to set up the integral method as a quadratic programming problem to ensure a positive definite transform with least error.

Overall, the ShielDose2 kernels match the output of the ShielDose2 application to within a few percent, which is comparable to the difference one sees running ShielDose2 with different output depths but for the same input spectrum.

4. References

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 P. Whelan, Specification for radiation effects kernels for use with AE9/AP9, ATR-2015-02436, The Aerospace Corporation, El Segundo, CA, 2015.

Seltzer, S. M., Updated Calculations for Routine Space-Shielding Radiation Dose Estimates: SHIELDOSE-2, Technical Report PB95-171039, National Institute of Standards and Technology, Gaithersburg, MD, December 1994.

AEROSPACE REPORT NO.
ATR-2016-01756

Developing AE9/AP9 Kernels from Third-Party Effects Codes

Approved Electronically by:

Margaret W. Chen, ASSOC DIRECTOR
SPACE SCIENCES DEPARTMENT
SPACE SCIENCE APPLICATIONS LABORATORY
ENGINEERING & TECHNOLOGY GROUP

Charles L. Gustafson, SR VP ENG & TECH
ENGINEERING & TECHNOLOGY GROUP

Technical Peer Review Performed by:

Mark D. Looper, RES SCIENTIST
MAGNETOSPHERIC & HELIOSPHERIC SCIENCES
SPACE SCIENCES DEPARTMENT
ENGINEERING & TECHNOLOGY GROUP

Office of General Counsel Approval Granted Electronically by:

Danielle E. Sherrod, ATTORNEY SENIOR
OFFICE OF THE GENERAL COUNSEL
OFFICE OF GENERAL COUNSEL & SECRETARY

SC2623