

AE9/AP9 Guidance for Third-Party Developers

March 25, 2014

T. Paul O'Brien
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: Engineering and Technology Group

Distribution Statement A: Approved for public release; distribution unlimited.

PHYSICAL SCIENCES LABORATORIES

The Aerospace Corporation functions as an “architect-engineer” for national security programs, specializing in advanced military space systems. The Corporation's Physical Sciences Laboratories support the effective and timely development and operation of national security systems through scientific research and the application of advanced technology. Vital to the success of the Corporation is the technical staff's wide-ranging expertise and its ability to stay abreast of new technological developments and program support issues associated with rapidly evolving space systems. Contributing capabilities are provided by these individual organizations:

Electronics and Photonics Laboratory: Microelectronics, VLSI reliability, failure analysis, solid-state device physics, compound semiconductors, radiation effects, infrared and CCD detector devices, data storage and display technologies; lasers and electro-optics, solid-state laser design, micro-optics, optical communications, and fiber-optic sensors; atomic frequency standards, applied laser spectroscopy, laser chemistry, atmospheric propagation and beam control, LIDAR/LADAR remote sensing; solar cell and array testing and evaluation, battery electrochemistry, battery testing and evaluation.

Space Materials Laboratory: Evaluation and characterizations of new materials and processing techniques: metals, alloys, ceramics, polymers, thin films, and composites; development of advanced deposition processes; nondestructive evaluation, component failure analysis and reliability; structural mechanics, fracture mechanics, and stress corrosion; analysis and evaluation of materials at cryogenic and elevated temperatures; launch vehicle fluid mechanics, heat transfer and flight dynamics; aerothermodynamics; chemical and electric propulsion; environmental chemistry; combustion processes; space environment effects on materials, hardening and vulnerability assessment; contamination, thermal and structural control; lubrication and surface phenomena. Microelectromechanical systems (MEMS) for space applications; laser micromachining; laser-surface physical and chemical interactions; micropropulsion; micro- and nanosatellite mission analysis; intelligent microinstruments for monitoring space and launch system environments.

Space Science Applications Laboratory: Magnetospheric, auroral and cosmic-ray physics, wave-particle interactions, magnetospheric plasma waves; atmospheric and ionospheric physics, density and composition of the upper atmosphere, remote sensing using atmospheric radiation; solar physics, infrared astronomy, infrared signature analysis; infrared surveillance, imaging and remote sensing; multispectral and hyperspectral sensor development; data analysis and algorithm development; applications of multispectral and hyperspectral imagery to defense, civil space, commercial, and environmental missions; effects of solar activity, magnetic storms and nuclear explosions on the Earth's atmosphere, ionosphere and magnetosphere; effects of electromagnetic and particulate radiations on space systems; space instrumentation, design, fabrication and test; environmental chemistry, trace detection; atmospheric chemical reactions, atmospheric optics, light scattering, state-specific chemical reactions, and radiative signatures of missile plumes.

Abstract

The AE9/AP9 model was envisioned from the beginning as a module to be integrated into existing and future satellite mission analysis packages. The basic interface to AE9/AP9 is described in an Application Programming Interface document. However, to exploit the statistical features of AE9/AP9, it is necessary to understand how to use the AE9/AP9 API to generate appropriate statistical quantities. This document provides guidance on what calculations to perform with AE9/AP9 itself, and what statistical manipulations to perform in post-processing to obtain valid statistical results. This document includes worked examples of computing percentiles of a dose-depth curve and percentiles of a worst-case internal charging proxy.

Acknowledgments

The author acknowledges useful discussions with the AE9/AP9 team, the AF-GEOSPACE team, the SPENVIS team, and the OMERE team.

Contents

1.	Introduction	1
2.	Simulating the Orbit with Adequate Time Sampling	2
3.	How to Generate Percentiles of a Dose-Depth Curve	3
3.1	Algorithm.....	3
3.2	Worked Example	5
4.	How to Generate Percentiles of Worst-Case Transient Effects	8
4.1	Algorithm.....	8
4.2	Worked Example	9
5.	A Note on Effects Kernels.....	13
	References	14

Figure

1.	Sample time series for AE9 Monte Carlo scenario #32 simulation with internal charging proxy. Only the first year is shown.	12
----	---	----

Tables

1.	Example AE9 Output for Perturbed Mean Scenario #32 ($\#/cm^2/s/MeV$)	5
2.	Example Fluence and Dose for Perturbed Mean Scenario #32 ($\#/cm^2/MeV$).....	6
3.	Dose vs Depth for Each Scenario (rads/10 yr)	6
4.	Sorted Dose-Depth Data (rads/10 yr)	7
5.	Example AE9 Flux Output for Monte Carlo Scenario #32 ($\#/cm^2/s$).....	10

6.	Example Internal Charging Proxy for Monte Carlo Scenario #32 (#/cm ² /s).....	10
7.	Running Maximum Internal Charging Proxy for Scenario #32 (#/cm ² /s)	11
8.	Worst-Case Internal Charging Proxy for Each Scenario (#/cm ² /s).....	11
9.	Sorted Worst-Case Internal Charging Proxy for Each Scenario (#/cm ² /s)	11

1. Introduction

AE9/AP9 adds a lot of functionality not present in AE8/AP8. In particular, AE9/AP9 adds the ability to compute error bars or confidence intervals on radiation effects. Past experience has shown that the statistical manipulations required with AE9/AP9 are often done incorrectly. The three most common problems are: (1) confusion between percentiles of effect and percentiles in the flux model, (2) confusion about where in the processing chain to run the effects code, (3) confusion over the difference between percentiles of accumulated quantities like dose versus worst-case transient quantities like single-event upset rate. This document aims to resolve that confusion through step-by-step instructions and worked examples.

When computing percentiles of radiation effects, it is tempting to fly the mission through static *percentile* environments, but results from such a calculation would be incorrect. In the case of an accumulated quantity like dose, flying part of the mission through a static 95th percentile flux map and then scaling up to the total mission duration would assume that the environment remained at its 95th percentile *for the entire mission*. That is exceedingly unlikely, and has no correspondence to anything that will occur with 95% probability. Conversely, attempting to compute a worst-case transient effect by flying a short interval in a 95th percentile static environment assumes that the environment *never* exceeds the 95th percentile over the course of the mission and that all particle populations in the relevant region of the belts are static and perfectly correlated with each other. Again, such a scenario is highly unrealistic. Unfortunately, because essentially all radiation effects mix together particles with different energies and locations in the radiation belts, and those different particle populations vary out-of-synch with each other and have widely varying dynamic timescales, it is not possible to estimate a relationship between a static percentile and the occurrence frequency of a transient effect.

To illustrate the correct procedure, we will walk through computation of a 95% confidence dose-versus-depth curve, which is a representative example of mission-accumulated quantities. We will also walk through computation of a 95% worst-case internal charging spectrum (or, more specifically, a spectrum of 95th percentile worst cases) as representative of worst-case transient effects. Finally, we will discuss the generalization of linear radiation effects as “kernels,” which will be a feature of a future version of AE9/AP9. Kernels will allow users and 3rd-party effects-code developers to exercise the AE9/AP9 machinery to manage runs and compute statistics on custom effects calculations.

2. Simulating the Orbit with Adequate Time Sampling

AE9/AP9 sometimes requires simulations of the entire mission duration for many scenarios. In such cases, choosing too short a time step can have severe penalties for computation time. Therefore, we provide the following guidelines to avoid oversampling.

For most orbits, we recommend a radius-dependent time step. We express this dependence in terms of the geocentric radius R expressed in Earth radii R_E (~ 6378 km):

- For $R \leq 1.3 R_E$, use 10 seconds
- For $R \leq 2 R_E$, use 1 minute
- For $2 < R \leq 4 R_E$, use 5-minute sampling
- For $R > 4 R_E$, use 15-minute sampling

For low Earth orbit (LEO) below ~ 2000 km or $R < 1.3 R_E$, we recommend a time step of 10 seconds. That is because the “horns” of the outer zone (outer Van Allen Belt) are fairly narrow in latitude, and so a small time step is necessary to ensure the time sampling does not inadvertently skip over part of the outer zone. As we move to greater distances, the spacecraft moves more slowly and the spatial features become more spread out.

For the special case of geostationary orbit (GEO), we recommend a time step of 1 hour because this unique orbit is confined to a very narrow part of the radiation belt and therefore moves slowly in relation to the belt.

As of version 1.05, AE9/AP9’s built-in *aggregators* do not account for variable time steps. Therefore, if the built-in aggregators are to be used, a fixed time step is required, and it should be the smallest time step anywhere in the orbit (it can be computed from the perigee radius).

3. How to Generate Percentiles of a Dose-Depth Curve

The first major category of radiation effect one can compute from AE9/AP9 is mission-accumulated effects. Such effects accumulate from the start of the mission and add up over the course of the mission. The most common example of a mission-accumulated effect is total dose. In this section, we walk through the calculation of total dose in algorithm form and then in the form of a worked example.

3.1 Algorithm

The first step is to obtain a flux time series along the spacecraft orbit. Because total dose depends on both protons and electrons, we must obtain flux time series for both AP9 and AE9. For mission-accumulated effects, there are two ways to run AE9/AP9 that can generate percentiles.

For all missions, it is appropriate to run many (e.g., $N_s = 40$ or 200) Monte Carlo scenarios for the entire mission duration for each species. This generates N_s time series of proton and electron flux versus energy along the spacecraft orbit for the entire mission. Each time series, or *Monte Carlo scenario*, represents a realistic variation of the radiation belts over the course of the mission. Each scenario also accounts for model uncertainty—each one has a slightly different mean environment. Running full-mission simulations via Monte Carlo scenarios can be a lengthy calculation and a large amount of data to handle, so we provide a faster alternative (*perturbed mean scenarios*) that we will explain below.

When using the Monte Carlo scenario method, the model should be set to run differential fluxes because that is the input required by the dose-depth calculator. We denote the flux time series for either species as $j_s(E_i, t_k)$, where s represents the scenario ID number, E_i represents the i^{th} energy channel, and t_k represents the k^{th} time step. Dose rate is a linear function of the input flux, so we can compute the dose accumulated over the entire mission from the time-integrated flux, denoted $J_s(E_i, t_k)$ and known as the *fluence*. Specifically, the fluence for scenario s in energy channel E_i is:

$$J_s(E_i, t_k) = \sum_{m=1}^k j_s(E_i, t_m) \Delta t_m \approx \int_0^{t'} j_s(E_i, t') dt'. \quad (1)$$

The time weight Δt_m can be defined in accordance with any weighting scheme appropriate for numeric integration. The critical fluence is the final fluence, i.e., at the last t_k , which we will denote as $J_s(E_i, \tau)$, where τ is the mission duration.

Under certain common circumstances, there is a faster way to estimate the fluence $J_s(E_i, \tau)$ for either species. For sufficiently long duration missions, the fluence “averages out” the dynamic variations. We, therefore, provide *perturbed mean scenarios* that leave out the dynamics, but do include uncertainty in the model itself (i.e., uncertainty in the mean), which tend to average out over long mission (e.g., $> 1\text{--}2$ years). When running the perturbed mean scenarios, one needs to run only a few repre-

sentative orbits. The perturbed mean scenarios share their ID numbers with the Monte Carlo scenarios because they share some of the same random number seeds. To compute the fluence, again, one must compute differential flux for a time series along the spacecraft orbit. However, when using the perturbed mean, the time series needs only to include a few representative orbits (usually a few days or weeks). We then scale the fluence for that short interval to the entire mission duration. We denote the short interval as τ' and the perturbed mean flux for one scenario as $j'_s(E_i, t_k)$. The fluence for the shorter perturbed mean run is:

$$J'_s(E_i, t_k) = \sum_{m=1}^k j'_s(E_i, t_m) \Delta t_m \approx \int_0^{t'} j'_s(E_i, t') dt'. \quad (2)$$

The fluence for the whole mission is then:

$$J_s(E_i, \tau) = \frac{\tau}{\tau'} J'_s(E_i, \tau'). \quad (3)$$

Note: the accumulation of flux into fluence is one kind of *aggregation*. The AE9/AP9 C++ code provides several pre-defined *aggregators*, including those for fluence and total dose. As of version 1.05, the pre-defined aggregators are not documented in the API [Whelan, 2014], but are available deeper in the source code. The command line utility and the GUI both provide some aggregator capability (fluence and total dose).

However we obtain the mission fluence $J_s(E_i, \tau)$, we must next compute the dose associated with that much fluence. In AE9/AP9, we pass the proton and electron fluence through ShieldDose2 [Seltzer, 1994]. We denote the resulting total dose at depth d_n for scenario s as $D_s(d_n)$, and n indexes the list of depths requested by the user.

Important: To get the statistics right, we must combine the proton and electron doses *before* we compute percentiles. That is because *percentiles do not add*; a percentile is a nonlinear statistic. For example, the sum of the 95th percentile electron dose and the 95th percentile proton dose is not the 95th percentile of the combined electron-plus-proton dose.

With a table $D_s(d_n)$ of doses versus depths, we need only compute the percentiles requested by the user. In this case, we are computing the 95th percentile. Typically, this is computed by sorting the scenarios at fixed depth to produce a new table: $D'_p(d_n)$ such that $D'_p(d_n) < D'_{p+1}(d_n)$, where p spans the integers from 1 to N_s . In this new table, the P^{th} percentile at a given depth d_n is given by the entry $D'_p(d_n)$ for d_n at q such that

$$q-1 \leq (N_s-1)P/100 < q. \quad (4)$$

There are other approaches to computing the specific percentile that involve interpolation, but to keep this exposition simple, we have used a simple look-up—the distinctions in the various methods become less significant as N_s grows large.

3.2 Worked Example

In our example for total dose, we will compute the 95th percentile total dose behind spherical shielding for a geostationary (GEO) vehicle. We will examine a 10-year mission, which is long enough for us to use the perturbed mean feature to only run 1 week and scale it up to 10 years. For best performance, developers are advised to use the direct API to access the AE9/AP9 “fly in” capability.

For ease of illustration, we will use the command line AE9/AP9 utility. Box 1 shows a partial AE9/AP9 command line utility input file. We have selected the default AE9 energy channels, and we have requested differential flux (*FluxType 1PtDiff*), which is what ShieldDose2 requires as input. After the setup lines, note that the 40 perturbed mean scenarios are requested, with ID numbers 1–40 via the *FluxOut* keyword with the *perturbed* parameter and the scenario ID (denoted *s* above). A similar file must be created to run AP9.

Box 1. Example AE9 input file for 1 week perturbed mean run

```

ModelType: AE9
Energies: 0.04,0.07,0.1,0.25,0.5,0.75,1,1.5,2,2.5,3,3.5,4,4.5,5,5.5,6,6.5,7,7.5,8
FluxType: 1PtDiff
CoordSys: RLL
OrbitFile: Run/GeoEphem1week.csv
...
FluxOut: perturbed,1
FluxOut: perturbed,2
...
FluxOut: perturbed,39
FluxOut: perturbed,40

```

We invoke the command line utility with a command like

```
CmdLineAe9Ap9.exe -i Run\DoseRun.AE9.CLinput.txt
```

This will create a set of output files in the Run directory with names like `DoseRun.AE9.CLoutput_pert_flux_032.txt`, where 032 is the scenario ID, and there will be scenario IDs 001 to 040. Table 1 shows the input time (MJD 53371.00 is midnight UTC on 1 Jan 2005) and ephemeris and the output results of this one scenario (with many rows and columns elided). The flux units are given in the header of the file and are $\#/cm^2/s/MeV$. The energy channels are the ones requested in the input file (Box 1).

Table 1. Example AE9 Output for Perturbed Mean Scenario #32 ($\#/cm^2/s/MeV$)

MJD	R (R _E)	lat	lon	$j_{32}(E_1, t_k)$	$j_{32}(E_2, t_k)$	$j_{32}(E_3, t_k)$...	$j_{32}(E_{20}, t_k)$	$j_{32}(E_{21}, t_k)$
53371.00	6.6	0	0	3.55E+09	2.29E+09	4.24E+08	...	9.59E+00	4.79E+00
53371.04	6.6	0	0	3.54E+09	2.28E+09	4.24E+08	...	9.12E+00	4.56E+00
53371.08	6.6	0	0	3.50E+09	2.25E+09	4.22E+08	...	8.48E+00	4.24E+00
53371.13	6.6	0	0	3.55E+09	2.28E+09	4.25E+08	...	9.06E+00	4.53E+00
...
53377.92	6.6	0	0	3.52E+09	2.26E+09	4.23E+08	...	9.19E+00	4.59E+00
53377.96	6.6	0	0	3.54E+09	2.27E+09	4.24E+08	...	9.45E+00	4.73E+00

Next, we compute the fluence $J'_s(E_i, t_k)$ for each scenario for the 1-week simulation ($\tau' = 1$ week) using Eq. (1). The time step is $\Delta t_k = 3600$ s for all time steps (AE9/AP9 v1.05 uses a slightly different time weighting scheme when it computes fluence). Table 2 shows the results of the fluence calculation for scenario #32. The fluence for a 1-week mission $J'_s(E_i, \tau')$ is given at MJD = 53377.96. We scale this up to 10 years by multiplying by $10 \times 365.25 / 7 = 521.79$. The final fluence for a 10-year mission is given in the last row of the table. The last few rows of Table 2 provide the 10-year fluence for both electrons and protons for scenario 32.

Next, for each scenario, we compute the total dose. Using ShieldDose2, we turn off nuclear interactions, select silicon as the target, and provide a list of 30 depths spaced logarithmically from 10 to 1000 mils Al. We feed the proton and electron fluence for each scenario to ShieldDose2, and compute the total dose for that scenario as the sum of the proton, electron, and bremsstrahlung (an electron effect) dose outputs for spherical geometry (i.e., 1/2 the dose at the centers of Al spheres). The resulting 40 dose depth data are given in Table 3.

Next, we sort the dose-depth data across scenarios. That is, we sort each column of Table 3 independently from least to greatest. Table 4 shows the results of sorting the columns of Table 3. Finally, we select the row for $p = q = 38$ (boldface), which satisfies Eq. (4) for $N_s = 40$ and $P = 95$. That is, the 38th smallest value in each column of Table 3 gives the 95th percentile because there are 40 scenarios. As it happens, at all depths d_n , scenario $s = 32$ ended up rank $p = 10$ after sorting, and all the rank $p = 38$ doses came from scenario $s = 30$; this is not merely a coincidence. Dose, like many effects, responds to a broad range of energies. So, it is quite common for sorting process to largely keep the values from individual scenarios together, even though the columns are sorted independently. We

Table 2. Example Fluence and Dose for Perturbed Mean Scenario #32 (#/cm²/MeV)

Electron Fluence, 1-week simulation						
MJD	$J'_{32}(E_1, t_k)$	$J'_{32}(E_2, t_k)$	$J'_{32}(E_3, t_k)$...	$J'_{32}(E_{20}, t_k)$	$J'_{32}(E_{21}, t_k)$
53371.00	1.28E+13	8.23E+12	1.53E+12	...	3.45E+04	1.73E+04
53371.04	2.55E+13	1.64E+13	3.06E+12	...	6.74E+04	3.37E+04
53371.08	3.81E+13	2.45E+13	4.57E+12	...	9.79E+04	4.89E+04
53371.13	5.09E+13	3.27E+13	6.10E+12	...	1.30E+05	6.52E+04
...
53377.92	2.17E+15	1.41E+15	2.52E+14	...	8.93E+06	4.81E+06
($\tau' = 7$ d) 53377.96	2.18E+15	1.42E+15	2.53E+14	...	8.97E+06	4.83E+06
1-week fluence rescaled to 10 years ($\tau = 10$ yr)						
	$J_{32}(E_1, \tau)$	$J_{32}(E_2, \tau)$	$J_{32}(E_3, \tau)$...	$J_{32}(E_{20}, \tau)$	$J_{32}(E_{21}, \tau)$
Electrons	1.14E+18	7.40E+17	1.32E+17	...	4.68E+09	2.52E+09
Protons ¹	5.20E+12	1.57E+12	3.78E+11	...	0	0

¹Note: Protons have 21 different energy channels

Table 3. Dose vs Depth for Each Scenario (rads/10 yr)

	$D_s(d_1)$	$D_s(d_2)$	$D_s(d_3)$...	$D_s(d_{29})$	$D_s(d_{30})$
Scenario, $s = 1$	5.66E+07	4.30E+07	3.26E+07	...	1.07E+03	9.04E+02
2	3.80E+07	2.94E+07	2.27E+07	...	7.26E+02	6.19E+02
...
32	4.10E+07	3.11E+07	2.36E+07	...	8.35E+02	7.06E+02
...
39	3.31E+07	2.61E+07	2.05E+07	...	7.13E+02	6.13E+02
40	3.39E+07	2.61E+07	2.01E+07	...	7.02E+02	5.97E+02

Table 4. Sorted Dose-Depth Data (rads/10 yr)

	$D'_p(d_1)$	$D'_p(d_2)$	$D'_p(d_3)$...	$D'_p(d_{29})$	$D'_p(d_{30})$
rank, $p=1$	2.74E+07	2.17E+07	1.72E+07	...	626.402	542.343
2	2.80E+07	2.27E+07	1.83E+07	...	691.726	602.249
...
10	4.10E+07	3.11E+07	2.36E+07	...	8.35E+02	7.06E+02
...
$p=q=38$	7.17E+07	5.33E+07	3.95E+07	...	1417.01	1190.23
39	8.39E+07	6.26E+07	4.65E+07	...	1587.96	1332.3
40	9.63E+07	6.96E+07	5.02E+07	...	1874.35	1542.19

cannot, however, know in advance which scenario will correspond to a specific percentile for a given N_s , and whether that will hold at all depths. Thus, we must run all N_s scenarios.

Picking the row with $p = 38$ from Table 4 completes our calculation of the 95th percentile dose-depth curve for 10 years at GEO. Other percentiles can be obtained by computing other values of q from the desired percentile P and $N_s = 40$ according to Eq. (4).

4. How to Generate Percentiles of Worst-Case Transient Effects

The second major category of radiation effect one can compute from AE9/AP9 is worst-case transient effects. Such effects are due to temporary conditions that, if sufficiently severe, could cause an anomaly or failure on the satellite. Typical examples include single-event upset rate or internal charging. The internal charging case is somewhat more difficult, so that is the one we will examine. In this section, we walk through the calculation of a worst-day internal charging environment in algorithm form and then in the form of a worked example.

Interpretation of percentiles for worst-case transient effects is sometimes confused. To clarify, a 95% confidence value for a worst-case transient effect is the threshold that will not be exceeded in 95% of missions at any time during those missions. That is different from and usually much higher than the threshold that will be exceeded approximately 5% of the time during every mission. One would not want a design to be out of spec 5% of the time (the latter); rather, one designs to not be out of spec at all, or with low probability (the former).

4.1 Algorithm

To compute percentiles for worst-case transient effects, it is necessary to simulate the entire mission duration for many *Monte Carlo scenarios*. For most transient effects, the effect itself depends on a range of energies. For example, single-event upset rates often have a fairly sharp proton energy threshold followed by a level response out to arbitrarily high energies. Similarly, internal charging tends to respond approximately uniformly to electrons above a threshold, albeit that threshold tends to be not very sharp. For these reasons, in the absence of a definition of the actual energy response of a specific effect, we recommend using integral fluxes (i.e., each flux is integrated over all energies above a threshold). The AE9/AP9 model can be set up to produce integral fluxes directly, with no need for the third-party developer to perform the integral. We denote the time series of these integral fluxes as $j_s(>E_i, t_k)$, where s represents the Monte Carlo scenario ID, E_i represents the i^{th} energy threshold, and t_k represents the k^{th} time step.

Internal charging is typically modeled as an RC circuit: the environment provides a current source, and the affected material “bleeds off” the accumulated charge in the same manner as a resistor-capacitor circuit. Namely, the charge bleeds off with an exponential time constant τ_{RC} . Because current is constantly being supplied by the environment, the charge at any given time is a weighted integral over all past fluxes. We use a weighted time integral of flux rather than charge as a proxy for this process (one can convert from flux to charge with certain additional properties of the hardware). The proxy at any time step is denoted $J_s(>E_i, t_k)$ and is given by:

$$\begin{aligned} J_s(>E_i, t_k) &= e^{-\Delta t_k / \tau_{RC}} J_s(>E_i, t_{k-1}) + (1 - e^{-\Delta t_k / \tau_{RC}}) j_s(>E_i, t_k) \\ &\approx \frac{1}{\tau_{RC}} \int_0^{t'} e^{-(t-t') / \tau_{RC}} j_s(>E_i, t') dt' \end{aligned} \quad (5)$$

We know that $J_s(>E_i, t_0) = 0$, and Δt_k can be computed using any one of several numerical integration weighting schemes.

The next step is to compute the maximum of $J_s(>E_i, t_k)$ at any point during the mission. It can be convenient to see how this maximum grows, so we can define the maximum as $M_s(>E_i, t_k)$:

$$M_s(>E_i, t_k) = \begin{cases} J_s(>E_i, t_k) & J_s(>E_i, t_k) > M_s(>E_i, t_{k-1}) \\ M_s(>E_i, t_{k-1}) & \text{otherwise} \end{cases} \quad (6)$$

At the end of the mission, we have the final value $M_s(>E_i, \tau)$, which represents the worst-case internal charging proxy for threshold E_i any time during scenario s . We create a new table M' , which we sort across scenarios such that $M'_{p'}(>E_i, \tau) \leq M'_{p'+1}(>E_i, \tau)$. From this sorted table, we can obtain percentile P by selecting the p such that $p-1 \leq (N_s-1)P/100 < p$. Again, there are more sophisticated ways to estimate the percentiles, but as N_s increases, all the different methods converge.

4.2 Worked Example

In our example for internal charging, we will compute the 95th percentile worst-case internal charging proxy for an RC circuit with a 24-hour time constant, τ_{RC} . We will again examine a 10-year mission. However, because we are dealing with a transient phenomenon, we cannot simulate a shorter interval in a static model and scale up to a 10-year mission. Instead, we must use dynamic Monte Carlo scenarios. For best performance, developers are advised to use the direct API to access the AE9/AP9 “fly in” capability. However, for this illustration, we will again use the command line utility. Box 2 shows the critical parts of the input file for the 10-year run of 40 scenarios. Note first that the *FluxOut* type is now *montecarlo*, with scenario IDs running from 1 to 40. Note also that we have selected the *Flux-Type Integral* to get integral energy channels rather than differential ones. We do not need to run AP9 for this internal charging example.

Box 2. Example AE9 input file for 10 year Monte Carlo scenario run

```
ModelType: AE9
Energies: 0.04,0.07,0.1,0.25,0.5,0.75,1,1.5,2,2.5,3,3.5,4,4.5,5,5.5,6,6.5,7,7.5,8
FluxType: Integral
CoordSys: RLL
OrbitFile: Run/GeoEphem10yr.csv
...
FluxOut: montecarlo,1
FluxOut: montecarlo,2
...
FluxOut: montecarlo,39
FluxOut: montecarlo,40
```

We invoke the command line utility with a command like

```
CmdLineAe9Ap9.exe -i Run\ICRun.AE9.CLinput.txt
```

This will create a set of output files in the Run directory with names like `DoseRun.AE9.CLoutput_mc_flux_032.txt`, where 032 is the scenario ID and there will be

scenario IDs 001 to 040. Table 5 shows the input time and ephemeris and the output results of this one scenario (with many rows and columns elided). The energy channels are the ones requested in the input file (Box 2). Compared to Table 1, we note several differences. The flux units are $\#/cm^2/s$, which reflects the choice of integral energy channels. The flux values are different, both because of the integral channel choice and because the Monte Carlo dynamic scenarios vary about their mean (but they have the same mean as the perturbed mean scenario with the same ID number). The last MJD is 57022.96, or 10 years after the first simulation. In this case, we ended the simulation at 23 Dec 2014 at 23:00 UTC, which is 10 calendar years after the start date of 1 Jan 2005 at 00:00 UTC.

Next, we compute the internal charging proxy $J_s(>E_i, t_k)$ according to Eq. (5). Table 6 provides a subset of the proxy for scenario 32.

Next, we compute $M_s(>E_i, t_k)$, the maximum of the internal charging proxy up to each time t_k , according to Eq. (6). Table 7 shows that running maximum for scenario 32. Note that in middle of the time series, $M_{32}(>E_i, t_k)$ is the same for many time steps in a row. At the end of the simulation, in the final row of Table 7, we have the worst case for scenario 32 (boldface), $M_{32}(>E_i, \tau)$ for $\tau = 10$ years.

Next, we assemble the final row from each scenario into a table of $M_s(>E_i, \tau)$. Table 8 shows the worst case (i.e., maximum over the entire simulation) of the internal charging proxy for each scenario.

Table 5. Example AE9 Flux Output for Monte Carlo Scenario #32 ($\#/cm^2/s$)

MJD	R (R_E)	lat	lon	$j_{32}(>E_1, t_k)$	$j_{32}(>E_2, t_k)$	$j_{32}(>E_3, t_k)$...	$j_{32}(>E_{20}, t_k)$	$j_{32}(>E_{21}, t_k)$
53371.00	6.6	0	0	8.52E+06	6.85E+06	5.56E+06	...	5.99E+00	1.50E+00
53371.04	6.6	0	0	9.41E+06	7.48E+06	6.02E+06	...	5.59E+00	1.40E+00
53371.08	6.6	0	0	1.03E+07	8.14E+06	6.50E+06	...	5.09E+00	1.27E+00
53371.13	6.6	0	0	1.14E+07	8.89E+06	7.02E+06	...	5.38E+00	1.35E+00
...
57022.92	6.6	0	0	2.61E+08	1.31E+08	6.87E+07	...	1.97E+00	4.92E-01
57022.96	6.6	0	0	2.58E+08	1.31E+08	6.91E+07	...	2.10E+00	5.26E-01

Table 6. Example Internal Charging Proxy for Monte Carlo Scenario #32 ($\#/cm^2/s$)

MJD	$J_{32}(>E_1, t_k)$	$J_{32}(>E_2, t_k)$	$J_{32}(>E_3, t_k)$...	$J_{32}(>E_{20}, t_k)$	$J_{32}(>E_{21}, t_k)$
53371.00	3.48E+05	2.79E+05	2.27E+05	...	2.44E-01	6.11E-02
53371.04	7.17E+05	5.73E+05	4.63E+05	...	4.63E-01	1.16E-01
53371.08	1.11E+06	8.82E+05	7.09E+05	...	6.52E-01	1.63E-01
53371.13	1.53E+06	1.21E+06	9.67E+05	...	8.45E-01	2.11E-01
53371.17	1.97E+06	1.55E+06	1.23E+06	...	1.10E+00	2.75E-01
53371.21	2.44E+06	1.90E+06	1.49E+06	...	1.40E+00	3.51E-01
...
56312.54	7.95E+08	2.85E+08	1.02E+08	...	1.04E-01	3.91E-02
56312.58	7.95E+08	2.86E+08	1.03E+08	...	1.09E-01	4.21E-02
56312.63	7.95E+08	2.87E+08	1.03E+08	...	1.13E-01	4.38E-02
56312.67	7.94E+08	2.87E+08	1.04E+08	...	1.16E-01	4.38E-02
...
57022.92	3.39E+08	1.48E+08	6.78E+07	...	2.57E+00	8.75E-01
57022.96	3.36E+08	1.47E+08	6.78E+07	...	2.55E+00	8.61E-01

Table 7. Running Maximum Internal Charging Proxy for Scenario #32 ($\#/cm^2/s$)

MJD	$M_{32}(>E_1, t_k)$	$M_{32}(>E_2, t_k)$	$M_{32}(>E_3, t_k)$...	$M_{32}(>E_{20}, t_k)$	$M_{32}(>E_{21}, t_k)$
53371.00	3.48E+05	2.79E+05	2.27E+05		0.244258	0.061064
53371.04	7.17E+05	5.73E+05	4.63E+05		0.462526	0.115631
53371.08	1.11E+06	8.82E+05	7.09E+05		0.651502	0.162876
53371.13	1.53E+06	1.21E+06	9.67E+05		0.844621	0.211155
53371.17	1.97E+06	1.55E+06	1.23E+06		1.101728	0.275432
53371.21	2.44E+06	1.90E+06	1.49E+06		1.40361	0.350903
56312.54	7.95E+08	2.90E+08	1.27E+08		43.53709	15.94653
56312.58	7.95E+08	2.90E+08	1.27E+08		43.53709	15.94653
56312.63	7.95E+08	2.90E+08	1.27E+08		43.53709	15.94653
56312.67	7.95E+08	2.90E+08	1.27E+08		43.53709	15.94653
57022.92	7.95E+08	2.90E+08	1.27E+08		46.25081	17.30999
($\tau=10$ yr) 57022.96	7.95E+08	2.90E+08	1.27E+08		46.25081	17.30999

 Table 8. Worst-Case Internal Charging Proxy for Each Scenario ($\#/cm^2/s$)

	$M_s(E_1, \tau)$	$M_s(E_2, \tau)$	$M_s(E_3, \tau)$...	$M_s(E_{20}, \tau)$	$M_s(E_{21}, \tau)$
Scenario, $s = 1$	1.38E+09	7.38E+08	4.19E+08	...	3.54E+02	9.86E+01
2	6.72E+08	3.63E+08	2.07E+08	...	2.45E+02	8.40E+01
...
32	7.95E+08	2.90E+08	1.27E+08	...	4.63E+01	1.73E+01
...
39	5.29E+08	2.65E+08	1.40E+08	...	2.48E+02	8.74E+01
40	6.43E+08	2.87E+08	1.45E+08	...	1.10E+02	3.72E+01

Finally, we sort the columns of Table 8 to obtain $M'_p(E_i, \tau)$, and pick off $q=38$, which satisfies Eq. (4) for $N_s=40$ scenarios and the 95th percentile ($P=95$). Table 9 shows the results of sorting each column of Table 8 independently. In this simulation, worst cases for each energy channel from scenario $s = 32$ became rank $p = 17$, and rank $p = 38$ came from scenario $s = 5$. As with total dose, because we have used integral energy channels (i.e., an effect proxy with a broad energy response), the results for individual scenarios tend to stay together when we sort the estimated effects, even though the columns are sorted separately. This will not necessarily happen for effects that respond to a narrow range of energies.

 Table 9. Sorted Worst-Case Internal Charging Proxy for Each Scenario ($\#/cm^2/s$)

	$M'_p(E_1, \tau)$	$M'_p(E_2, \tau)$	$M'_p(E_3, \tau)$...	$M'_p(E_{20}, \tau)$	$M'_p(E_{21}, \tau)$
Rank, $p = 1$	2.41E+08	1.40E+08	9.01E+07	...	1.03E+02	3.44E+01
2	2.59E+08	1.38E+08	8.09E+07	...	1.58E+02	4.22E+01
...
17	7.95E+08	2.90E+08	1.27E+08	...	4.63E+01	1.73E+01
...
$p=q=38$	2.08E+09	1.01E+09	5.02E+08	...	2.84E+02	1.10E+02
39	2.18E+09	8.78E+08	3.76E+08	...	2.25E+02	8.09E+01
40	6.37E+09	2.35E+09	7.95E+08	...	2.35E+02	8.64E+01

Picking the row with $p = 38$ from Table 9 completes our calculation of the 95th percentile worst-case internal charging for 10 years at GEO. As with our total dose example, other percentiles can be obtained by computing other values of q from the desired percentile P and $N_s = 40$ according to Eq. (4).

Figure 1 shows the various steps in processing a single scenario (#32): the flux time series, the internal charging proxy time series ($\tau_{RC}=24$ h), the running maximum proxy, the final maximum proxy ($\tau=10$ yr), and the 95th percentile value (which is taken from Table 9, and originates in scenario #5). For ease of resolving details, only the first year of ten is shown.

We have produced a spectrum of 95th percentile worst cases, rather than a worst-case spectrum at 95% confidence, which is not well-defined without specific assumptions about shielding design and part/system susceptibility. By using integral electron fluxes, however, we have maintained much of the correlation that would be captured by simulation of an exact design. We will explore how to improve upon the integral channel as proxy for a real system in the next section.

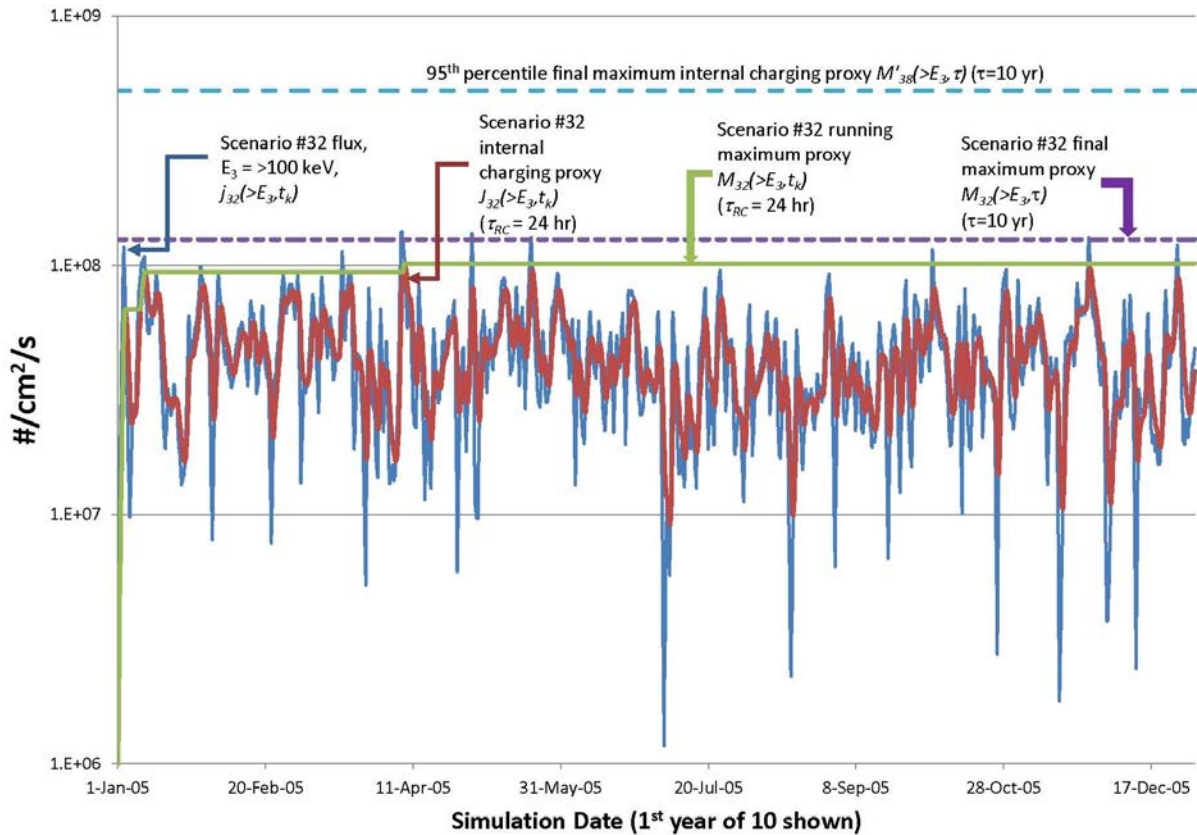


Figure 1. Sample time series for AE9 Monte Carlo scenario #32 simulation with internal charging proxy. Only the first year is shown.

5. A Note on Effects Kernels

We are developing the concept of “kernels,” which will be defined in XML files externally to AE9/AP9. These XML files can be read in by the AE9/AP9 library and used to compute user-defined effects. Kernels can be used to define any effect that is a linear function of the flux or fluence. Because they are defined outside the AE9/AP9 code base, new kernels can be utilized by AE9/AP9 without modifying the AE9/AP9 source code. Examples of such effects include: ionizing dose or dose rate for a custom shielding geometry, displacement damage or damage rate, single-event effects rate (for a part that does not get progressively noisier), internal charging current (neglecting electrostatic repulsion from built-up potential), and internal charging potential (assuming no discharges or electrostatic deflection by large built-up fields).

Because many effects are linear in the flux, they can be represented as convolution integrals—they convolve the flux or fluence with a “Green’s function” or impulse response function. In fact, many modern effects codes are exactly that: convolutions. The kernel is a way to capture that convolution as a pre-defined matrix (a matrix is a discrete convolution). The convolution kernel is often derived from a Monte Carlo simulation, such as GEANT4 [Agostinelli et al., 2003]. O’Brien and Kwan [2013] describe the kernel concept in more detail in the context of displacement damage in silicon beneath spherical Al shields.

Kernels are a partial alternative to implementing a derived C++ aggregator class. (aggregator classes can do nonlinear things, too, while kernels are limited to only linear effects). The kernel XML file specification is under development as [O’Brien and Whelan, 2014].

References

- Whelan, P., "AE9/AP9/SPM Model Application Programming Interface, Version 1.00.000," AFRL Report, Kirtland AFB, NM, 2012, AFRL-RV-PS-TR-2014-0018, 2014.
- Agostinelli, S., et al., "Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment," *Nucl. Inst. and Methods*, **506**(3), 250, 1 July 2003.
- 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," The Aerospace Corporation, El Segundo, CA, 2014. Document in preparation.
- Seltzer, S. M., "Updated calculations for routine space-shielding radiation dose estimates: SHIELD-DOSE-2." Gaithersburg, MD, NIST Publication NISTIR 5477, 1994.

AE9/AP9 Guidance for Third-Party Developers

Approved Electronically by:

James L. Roeder, DIRECTOR
SPACE SCIENCES DEPARTMENT
SPACE SCIENCE APPLICATIONS
LABORATORY
ENGINEERING & TECHNOLOGY GROUP

Rami R. Razouk, SR VP ENG & TECH
ENGINEERING & TECHNOLOGY GROUP