AEROSPACE REPORT NO.
ATR-2016-00932

AE9/AP9 Monte Carlo Upgrades (From v1.2 to v1.3)

February 15, 2016

T. P. O'Brien and Timothy B. Guild
Space Science Applications Laboratory
Physical Sciences Laboratories

Prepared for:

Sr. Vice President
Technology and Laboratory Operations

Authorized by: Engineering and Technology Group

Abstract

This report describes upgrades to the AE9/AP9 Monte Carlo machinery that generates dynamic scenarios. These changes were developed to fix problems with AE9/AP9 v1.2 in preparation for release v1.3. The problems manifest in two ways: First, Monte Carlo scenarios of AP9 v1.2 would become unstable, producing unrealistically extreme flux values; second, AE9 Monte Carlo simulations did not produce the same long-term fluence as corresponding simulations in the perturbed mean environments. Two changes address numerical uncertainty in a calculation meant to ensure the Monte Carlo state does not become unstable after many time steps. The remaining changes ensure that the Monte Carlo time series reproduces the statistical distributions of the static flux maps much more accurately than in prior versions. Specifically, these changes ensure that the variance of the Monte Carlo state variables have the required value of 1 under realistic use cases. Because these shortcomings of the Monte Carlo machinery went unnoticed through multiple releases of AE9/AP9, we conclude with recommendations for a set of tests that will catch these kinds of discrepancies in the future. This document supersedes TOR-2012(1237)-3, and updates parts of TOR-2014-00294 and TOR-2014-00295.

Acknowledgments

The authors acknowledges useful discussions with the AE9/AP9 team as well as our colleagues at The Aerospace Corporation. The authors thank Betty P. Kwan for help testing AE9/AP9.

Contents

Figures

1. Introduction

In response to requirements from industry users, the AE9/AP9 model includes the ability to represent radiation belt dynamics [*Ginet et al.,* 2013]. The model represents dynamics through Monte Carlo scenarios, which evolve a set of principal component amplitudes (*q*'s) in time using an autoregressive (AR) equation. The autoregressive equation for AE9/AP9 is somewhat unusual because it only uses the past history at certain specific (geophysically significant) time lags [*O'Brien,* 2012]. The machinery for developing the AR coefficients was designed to produce time series that reflected the statistical properties of the static AE9/AP9 flux maps. This sparse time lag structure has proved more difficult to develop and implement correctly than was originally thought. Additionally, the typical user evaluates AE9/AP9 at times between the Monte Carlo time steps, necessitating time interpolation. The interpolation affects the statistical properties of the Monte Carlo scenarios and must be accounted for to produce correct particle flux statistics.

In the remainder of this document, we will review the Monte Carlo formulation, and then provide updated algorithms for deriving the AR coefficients, verifying them, and initializing the Monte Carlo state. Finally, we will recommend a set of routine verification and diagnostic tests that could be used to verify that the Monte Carlo machinery is working as intended before each future model release.

This report supersedes *O'Brien* [2012], which described the multiple time lags in AE9/AP9. This report also updates parts of *O'Brien* [2013a], which described the AE9/AP9 v1.0 runtime algorithms; and parts of [2013b], which described algorithms for generating the AE9/AP9 v1.0 data tables.

2. The Monte Carlo Formulation—the autoregressive equation

As described in *O'Brien* [2013a], the Monte Carlo state is represented by a small number (~10) of principal component amplitudes (q's). These amplitudes are represented as a state vector $\vec{q}_t$ that evolves in time according to an autoregressive (AR) equation:

$$\vec{q}_t = \sum_{i=1}^{N_G} \underline{\underline{G}}_i \vec{q}_{t-\tau_i} + \underline{\underline{C}} \vec{\eta}_t \tag{1}$$

In this equation, N_G is the number of different lags τ_i used. All the τ_i are integer multiples of a fiducial time spacing δt, so that Eq. (1) represents evolution in discrete time. A vector of zero-mean, unit-variance, mutually independent white noise $\vec{\eta}_t$ is added at each time step to produce random variations. The $\underline{\underline{G}}_i$ and $\underline{\underline{C}}$ matrices (the AR coefficients) are derived from observed spatiotemporal correlations. We will return to the derivation of these matrices in the next section. First, however, we must describe how the q's relate to the statistical distribution of flux at each grid point in the model.

Equation (1) provides $\vec{q}$ at a set of evenly spaced fiducial times, separated by δt. In a typical use case, however, AE9/AP9 must be evaluated at a user-specified list of time tags. We linearly interpolate from the surrounding model time steps t_j and t_{j+1} to the user's time tags, t'_k:

$$\vec{q}_{t'_k} = c_k \vec{q}_{t_j} + (1 - c_k) \vec{q}_{t_{j+1}}, \tag{2}$$

where

$$t_j \le t'_k < t_{j+1} \tag{3}$$

$$c_k = \frac{t_{j+1} - t'_k}{t_{j+1} - t_j} = \frac{t_{j+1} - t'_k}{\delta t}. \tag{4}$$

To convert from a state $\vec{q}_{t'}$ to a global flux map involves two transforms. The first transform maps the principal components onto the model grid in (E,K,Φ) and (E,K,h_{min}) coordinates defined in *Ginet et al.*, [2013]. The first transform is a matrix-vector operation:

$$\vec{z}_{t'} = \underline{\underline{Q}} \vec{q}_{t'}. \tag{5}$$

There is one z variable for each (E,K,Φ) and (E,K,h_{min}) grid point, and the principal component matrix $\underline{\underline{Q}}$ is derived from the spatial covariance of the fluxes [see *O'Brien*, 2013b]. Each z is converted to flux using either a Weibull distribution for AE9 or a log-normal distribution for AP9. This

conversion from z to flux assumes that each z is a Gaussian random variable with zero mean and unit variance. These assumptions become the following constraints:

$$\langle \vec{z}_{t'} \rangle = \underline{\underline{Q}} \langle \vec{q}_{t'} \rangle = 0 \tag{6}$$

$$\text{var}\, z_{t'}^{(i)} = \langle \left(z_{t'}^{(i)} \right)^2 \rangle - \left(\langle z_{t'}^{(i)} \rangle \right)^2 = \sum_j Q_{ij}^2 \left(\langle q_{t'}^{(i)} q_{t'}^{(j)} \rangle - \langle q_{t'}^{(i)} \rangle \langle q_{t'}^{(j)} \rangle \right) = 1 \tag{7}$$

The angle brackets $\langle \cdot \rangle$ indicate expected value or average. These conditions must be met whether the average is taken over time or over different scenarios. To meet these constraints, we have elected the following formulation:

$$\langle \vec{q}_{t'} \rangle = 0 \tag{8}$$

$$\langle \vec{q}_{t'} \vec{q}_{t'}^{\,T} \rangle = \underline{\underline{I}} \tag{9}$$

$$\sum_j Q_{ij}^2 = 1 \tag{10}$$

Thus, the q's have zero mean (8) and are independent of each other with unit variance (9), and the sum of squares of the rows of $\underline{\underline{Q}}$ is 1 for each row (10). The condition on $\underline{\underline{Q}}$ is accounted for by construction in O'Brien [2013b]. The condition on the means of the q's is satisfied because the white noise $\vec{\eta}_t$ in (1) has zero mean. Satisfying condition (9) results in a series of assumptions about c_k and constraints on the $\underline{\underline{G}}_j$ and $\underline{\underline{C}}$ matrices in (1), which are the subject of the next section.

3. Deriving the AR Coefficients

Deriving the AR coefficients involves several steps. First, we generate spatiotemporal covariance matrices for the principal component amplitudes (q's) by manipulating observed spatiotemporal covariances of fluxes. Then we solve a large system of equations for preliminary $\underline{\underline{\breve{G}_j}}$ and $\underline{\underline{\breve{C}}}$ matrices. We then iteratively refine these matrices to obtain the correct covariance of $\vec{q}_t$ (q's at fiducial time steps). We then correct for interpolation by making reasonable assumptions about c_k.

3.1 Generating Spatiotemporal Covariance Matrices for the Principal Component Amplitudes

We begin with a set of spatiotemporal covariances computed on a reduced grid. For AE9 and AP9, this reduced grid is essentially a random subset of points taken from the (E,K,Φ) grid. By reducing the grid, we are able to avoid having to compute spatiotemporal covariances for the thousands of grid points at every lag. When computing a spatiotemporal covariance, we start with a time series of daily-averaged (AE9) or weekly-averaged (AP9) fluxes for each sensor in each bin. Within each bin, for each sensor, we convert the fluxes into standard Gaussian variables (z's) by sorting and replacing. If flux j_i represents the i^{th} smallest flux in a bin for a sensor, then we replace that flux with its corresponding z:

$$j_i \to z_i = \Phi^{-1}\left(\frac{i}{N+1}\right). \tag{11}$$

Here N is the number of points for the sensor in the bin, and Φ^{-1} is the inverse of the cumulative distribution function for a Gaussian variable with zero mean and unit variance. (Note this is not the magnetic coordinate Φ used in the AE9/AP9 grid). We compute the covariance of the z's at each of the needed time lags by offsetting the time series for one sensor in one bin relative to the time series for the other sensor in the other bin. We do not have enough data to compute the lag covariance for every grid point against every other grid point (even on a reduced grid). Instead, at each lag, we compute a set of covariances by randomly selecting grid points (within the reduced grid) and sensor pairs. We then use a nearest-neighbors method to fill in the full covariance matrix at the given lag from these computed covariances [see *O'Brien*, 2013b]. We denote the resulting matrix

$$\breve{R}_M^{(i,j)} = \mathrm{cov}\left(z_t^{(i)}, z_{t-M\delta t}^{(j)}\right) = \langle z_t^{(i)} z_{t-M\delta t}^{(j)} \rangle, \tag{12}$$

where i,j each span the reduced grid. We then create a reduced $\underline{\breve{Q}}$, such that $\underline{\underline{\breve{Q}\breve{Q}^T}} = \underline{\underline{\breve{R}}}_0 = \underline{\underline{\breve{\Sigma}}}$ is the spatial covariance (at zero lag) on the reduced grid. We can convert $\underline{\underline{\breve{R}}}_M$ (lag covariance at M time steps for the z's) to $\underline{\underline{R}}_M$ (lag covariance at M time steps for the q's) using:

$$\underline{\underline{R}}_M = \langle \vec{q}_t \vec{q}_{t-M\delta t}{}^T \rangle = \underline{\underline{\breve{Q}}}^{-1} \underline{\underline{\breve{R}}}_M \left(\underline{\underline{\breve{Q}}}^{-1} \right)^T. \tag{13}$$

We recover the condition $\langle \vec{q}_t \vec{q}_t{}^T \rangle = \underline{\underline{I}}$:

$$\langle \vec{q}_t \vec{q}_t{}^T \rangle = \underline{\underline{R}}_0 = \underline{\underline{\breve{Q}}}^{-1} \underline{\underline{\breve{\Sigma}}} \left(\underline{\underline{\breve{Q}}}^{-1} \right)^T = \underline{\underline{I}}. \tag{14}$$

So, on the fiducial time steps the zero-lag covariance matrix of the q's is the identity matrix. That is why they are designated the principal components of spatial covariance. Next, we will solve for $\underline{\underline{\breve{G}}}_i$ and $\underline{\underline{\breve{C}}}$ matrices on the fiducial time steps using the spatiotemporal covariances $\underline{\underline{R}}_M$.

3.2 Solving for Preliminary AR Coefficients

In this section, we work through a set of equations that we can use to obtain preliminary $\underline{\underline{\breve{G}}}_i$ and $\underline{\underline{\breve{C}}}$ matrices. If we define the integer $T_i = \tau_i/\delta t$, then right multiply both sides of (1) by $\vec{q}_{t-T_j\delta t}{}^T$, and then take the expected value of both sides, we have:

$$\langle \vec{q}_t \vec{q}_{t-T_j\delta t}{}^T \rangle = \sum_{i=1}^{N_G} \underline{\underline{\breve{G}}}_i \langle \vec{q}_{t-T_i\delta t} \vec{q}_{t-T_j\delta t}{}^T \rangle + \underline{\underline{\breve{C}}} \langle \vec{\eta}_t \vec{q}_{t-T_j\delta t}{}^T \rangle. \tag{15}$$

We note that $\vec{\eta}_t$ has zero mean, and its covariance is the identity matrix, and it is uncorrelated with itself at all lags, or with any prior $\vec{q}_t$. We can, therefore, write the last term as:

$$\underline{\underline{\breve{C}}} \langle \vec{\eta}_t \vec{q}_{t-T_j\delta t}{}^T \rangle = \begin{cases} \underline{\underline{\breve{C}}} \langle \vec{\eta}_t \vec{\eta}_t{}^T \rangle \underline{\underline{\breve{C}}}^T = \underline{\underline{\breve{C}}} \underline{\underline{\breve{C}}}^T & T_j = 0 \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

Using (13) and (16), (15) becomes:

$$\underline{\underline{R}}_{T_j} = \sum_{i=1}^{N_G} \underline{\underline{\breve{G}}}_i \underline{\underline{R}}_{T_j - T_i} + \begin{cases} \underline{\underline{\breve{C}}} \underline{\underline{\breve{C}}}^T & T_j = 0 \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

We recognize that $\underline{\underline{R}}_{-M} = \underline{\underline{R}}_M{}^T$ and that $\underline{\underline{R}}_0 = \underline{\underline{I}}$. Using the $T_j > 0$ cases, we can set up a block matrix-matrix equation for $\underline{\underline{\breve{G}}}_i$ (working in the transposes so as to have the usual form $\underline{\underline{AX}} = \underline{\underline{B}}$):

$$\begin{bmatrix} \underline{\underline{I}} & \underline{\underline{R}}_{T_2-T_1} & \underline{\underline{R}}_{T_3-T_1} & \cdots & \underline{\underline{R}}_{T_{N_G}-T_1} \\ \underline{\underline{R}}^T_{T_2-T_1} & \underline{\underline{I}} & \underline{\underline{R}}_{T_3-T_2} & \cdots & \underline{\underline{R}}_{T_{N_G}-T_2} \\ \underline{\underline{R}}^T_{T_3-T_1} & \underline{\underline{R}}^T_{T_2-T_1} & \underline{\underline{I}} & \cdots & \underline{\underline{R}}_{T_{N_G}-T_3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \underline{\underline{R}}^T_{T_{N_G}-T_1} & \underline{\underline{R}}^T_{T_{N_G}-T_2} & \underline{\underline{R}}^T_{T_{N_G}-T_3} & \cdots & \underline{\underline{I}} \end{bmatrix} \begin{bmatrix} \underline{\underline{\check{G}}}^T_1 \\ \underline{\underline{\check{G}}}^T_2 \\ \underline{\underline{\check{G}}}^T_3 \\ \vdots \\ \underline{\underline{\check{G}}}^T_{N_G} \end{bmatrix} = \begin{bmatrix} \underline{\underline{R}}^T_{T_1} \\ \underline{\underline{R}}^T_{T_2} \\ \underline{\underline{R}}^T_{T_3} \\ \vdots \\ \underline{\underline{R}}^T_{T_{N_G}} \end{bmatrix} \tag{18}$$

This is a square matrix involving $\underline{\underline{R}}_{T_i-T_j}$ for all combinations of T_i and T_j. For example, if there are two lags of 1 and 7 steps, we need we need $\underline{\underline{R}}_M$ at M = 1, 7, *and* 7-1= 6. Thus, up to $N_G(N_G+1)/2$ $\underline{\underline{R}}_M$ are needed. For a typical case of ~5 lags and ~10 principal components, there are 50 unknowns. Thus (18) represents a large, but not intractable, equation for a modern computer.

To recover $\underline{\underline{\check{C}}}$, we use the T_j=0 case of Eq. (16) and the $\underline{\underline{\check{G}}}_i$ obtained by solving (18). Thus, we have:

$$\underline{\underline{\check{C}}}\underline{\underline{\check{C}}}^T = \underline{\underline{I}} - \sum_{i=1}^{N_G} \underline{\underline{\check{G}}}_i \underline{\underline{R}}_{T_i}{}^T. \tag{19}$$

We note that (18) can be used to show that $\underline{\underline{\check{C}}}\underline{\underline{\check{C}}}^T$ is symmetric. We can obtain $\underline{\underline{\check{C}}}$, the square root of $\underline{\underline{\check{C}}}\underline{\underline{\check{C}}}^T$, by Cholesky factorization or by eigenvalue decomposition. (It is good practice to symmetrize any floating-point error by performing $\underline{\underline{\check{C}}}\underline{\underline{\check{C}}}^T \rightarrow \left[\underline{\underline{\check{C}}}\underline{\underline{\check{C}}}^T + \left(\underline{\underline{\check{C}}}\underline{\underline{\check{C}}}^T\right)^T\right]/2$ before taking the matrix square root).

Solving (18) using approximate $\underline{\underline{R}}_M$ can only give us AR coefficients that approximately satisfy the $\langle \vec{q}_t \vec{q}_t{}^T \rangle = \underline{\underline{I}}$ constraint. In the next section, we will provide an iterative way to adjust $\underline{\underline{\check{G}}}_i$ and $\underline{\underline{\check{C}}}$ to exactly meet $\langle \vec{q}_t \vec{q}_t{}^T \rangle = \underline{\underline{I}}$, while approximating the other $\underline{\underline{R}}_M$.

3.3 Refining AR Coefficients for Ideal Properties on Fiducial Time Steps

In this section, we work through the equations needed to adjust the $\underline{\underline{\check{G}}}_i$ and $\underline{\underline{\check{C}}}$ matrices obtained in section 3.2 so that they preserve the constraint that $\langle \vec{q}_t \vec{q}_t{}^T \rangle = \underline{\underline{I}}$, i.e., that the principal components (without interpolation) are independent of each other and that each one has unit variance. To begin, we consider a generic, first-order AR process:

$$\vec{x}_t = \underline{\underline{A}}\vec{x}_{t-1} + \underline{\underline{B}}\vec{\eta}_t. \tag{20}$$

Again, $\vec{\eta}_t$ is zero mean, unit variance, mutually independent Gaussian white noise. Therefore, $\vec{x}_t$ has zero mean. The AR process is stable if the largest eigenvalue has magnitude less than 1 [see, e.g., *Neumaier and Schneider*, 2001]. We can obtain an expression for the covariance of $\vec{x}_t$ by right multiplying both sides by $\vec{x}_t{}^T$ and taking the expected value:

$$\langle \vec{x}_t \vec{x}_t{}^T \rangle = \underline{\underline{A}} \langle \vec{x}_{t-1} \vec{x}_t{}^T \rangle + \underline{\underline{BB}}^T \tag{21}$$

We have exploited the fact that $\langle \vec{\eta}_t \vec{\eta}_t{}^T \rangle = \underline{\underline{I}}$ and $\langle \vec{x}_{t-1} \vec{\eta}_t{}^T \rangle = 0$. If we right multiply both sides by $\vec{x}_{t-1}{}^T$ and take the expected value, we have:

$$\langle \vec{x}_t \vec{x}_{t-1}{}^T \rangle = \underline{\underline{A}} \langle \vec{x}_{t-1} \vec{x}_{t-1}{}^T \rangle. \tag{22}$$

Letting $\underline{\underline{\Sigma}} = \langle \vec{x}_t \vec{x}_t{}^T \rangle$ and $\underline{\underline{\mathrm{R}}} = \langle \vec{x}_t \vec{x}_{t-1}{}^T \rangle$, we can rewrite (21) and (22) as

$$\underline{\underline{\Sigma}} = \underline{\underline{A\mathrm{R}}}^T + \underline{\underline{BB}}^T \tag{23}$$

$$\underline{\underline{\mathrm{R}}} = \underline{\underline{A\Sigma}} \tag{24}$$

In the previous section, we solved the multi-lag counterpart of these equations to obtain the AR coefficients corresponding to $\underline{\underline{A}}$ and $\underline{\underline{B}}$. However, if we, instead, have $\underline{\underline{A}}$ and $\underline{\underline{B}}$ and we want to compute $\underline{\underline{\Sigma}}$ and $\underline{\underline{R}}$, then replace (23) with

$$\underline{\underline{\Sigma}} = \underline{\underline{A\Sigma\mathrm{A}}}^T + \underline{\underline{BB}}^T. \tag{25}$$

This equation is linear in $\underline{\underline{\Sigma}}$, but it cannot be solved in matrix-matrix form. Instead, we have to "unwrap" $\underline{\underline{\Sigma}}$ as a vector. First, we rewrite in subscript form:

$$\Sigma_{ij} = \sum_k \mathrm{A}_{ik} \sum_l \Sigma_{kl} \mathrm{A}_{jl} + \sum_m B_{im} B_{jm} \tag{26}$$

Then we rearrange with the help of the Kronecker delta:

$$\sum_k \sum_l \left(\mathrm{A}_{ik} \mathrm{A}_{jl} - \delta_{ik} \delta_{jl}\right) \Sigma_{kl} = -\sum_m B_{im} B_{jm} \tag{27}$$

We can exploit the symmetry of $\underline{\underline{\Sigma}}$ to further reduce this system:

$$\sum_k \sum_{l \le k} (\mathrm{A}_{ik} \mathrm{A}_{jl} - \delta_{ik} \delta_{jl} + \begin{cases} \mathrm{A}_{il} \mathrm{A}_{jk} & l < k \\ 0 & l = k \end{cases}) \Sigma_{kl} = -\sum_m B_{im} B_{jm} \quad (j \le i) \tag{28}$$

These two equations represent a linear system in Σ_{kl} for $l \le k$ and $j \le i$, with an exact solution that can be obtained by treating *ij* as a single index spanning the constraints and *kl* as a single index spanning the unknowns. This reduces the size of the matrix by almost a factor of 2 in each dimension, resulting in a faster calculation that uses less memory.

Now, we can recast (1) as a first-order AR process in a variable that includes not just $\vec{q}_t$ but also its history, we have:

$$\vec{x}_t = \begin{bmatrix} \vec{q}_t \\ \vec{q}_{t-\delta t} \\ \vec{q}_{t-2\delta t} \\ \vdots \\ \vec{q}_{t-\left(T_{N_G}-1\right)\delta t} \end{bmatrix} \tag{29}$$

$$\underline{\underline{B}} = \begin{bmatrix} \underline{\underline{\check{C}}} \\ \underline{\underline{0}} \\ \underline{\underline{0}} \\ \vdots \\ \underline{\underline{0}} \end{bmatrix} \tag{30}$$

$$\underline{\underline{A}} = \begin{bmatrix} \underline{\underline{\check{A}}}_1 & \underline{\underline{\check{A}}}_2 & \cdots & \underline{\underline{\check{A}}}_{N_G-1} & \underline{\underline{\check{A}}}_{N_G} \\ \underline{\underline{I}} & \underline{\underline{0}} & \cdots & \underline{\underline{0}} & \underline{\underline{0}} \\ \underline{\underline{0}} & \underline{\underline{I}} & \cdots & \underline{\underline{0}} & \underline{\underline{0}} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \underline{\underline{0}} & \underline{\underline{0}} & \cdots & \underline{\underline{I}} & \underline{\underline{0}} \end{bmatrix} \tag{31}$$

$$\underline{\underline{\check{A}}}_j = \begin{cases} \underline{\underline{\check{G}}}_j & j \in \{T_i\} \\ \underline{\underline{0}} & \text{otherwise} \end{cases} \tag{32}$$

This new $\underline{\underline{A}}$ matrix is very large. For example, for ~10 principal components and lags up to 365 time steps (i.e., 1 year lag in AE9), there are 3650 rows and columns in $\underline{\underline{A}}$. Thankfully, $\underline{\underline{A}}$ and $\underline{\underline{B}}$ are sparse, and so do not take up too much memory, nor do matrix operations necessarily take too long.

The covariance of $\vec{x}_t$ is given by solving the linear system (28), which is very large with on the order of a million rows and columns. The resulting covariance is:

$$\operatorname{cov} \vec{x}_t = \begin{bmatrix} \langle \vec{q}_t \vec{q}_t{}^T \rangle & \langle \vec{q}_t \vec{q}_{t-\delta t}{}^T \rangle & \cdots & \langle \vec{q}_t \vec{q}_{t-\left(T_{N_G}-1\right)\delta t}{}^T \rangle \\ \langle \vec{q}_{t-\delta t} \vec{q}_t{}^T \rangle & \langle \vec{q}_{t-\delta t} \vec{q}_{t-\delta t}{}^T \rangle & \cdots & \langle \vec{q}_{t-\delta t} \vec{q}_{t-\left(T_{N_G}-1\right)\delta t}{}^T \rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle \vec{q}_{t-\left(T_{N_G}-1\right)\delta t} \vec{q}_t{}^T \rangle & \langle \vec{q}_{t-\left(T_{N_G}-1\right)\delta t} \vec{q}_{t-\delta t}{}^T \rangle & \cdots & \langle \vec{q}_{t-\left(T_{N_G}-1\right)\delta t} \vec{q}_{t-\left(T_{N_G}-1\right)\delta t}{}^T \rangle \end{bmatrix} \tag{33}$$

We are only concerned with the upper left block (although, when we are done, all blocks on the diagonal will be the identity matrix).

Suppose using these manipulations, a given set of $\underline{\underline{\breve{G}_i}}$ and $\underline{\underline{\breve{C}}}$ yields $\langle \breve{q}_t \breve{q}_t{}^T \rangle \approx \underline{\underline{I}}$. Then, a transform exists $\vec{q}_t = \underline{\underline{\Gamma}} \breve{q}_t$ such that the transformed variable will have $\langle \vec{q}_t \vec{q}_t{}^T \rangle = \underline{\underline{I}}$. Specifically,

$$\langle \vec{q}_t \vec{q}_t{}^T \rangle = \underline{\underline{\Gamma}} \langle \breve{q}_t \breve{q}_t{}^T \rangle \underline{\underline{\Gamma}}^T = \underline{\underline{I}}, \tag{34}$$

which we solve as:

$$\underline{\underline{\Gamma}}^{-1} \underline{\underline{\Gamma}}^{-1^T} = \langle \breve{q}_t \breve{q}_t{}^T \rangle. \tag{35}$$

We can then compute $\underline{\underline{\Gamma}}^{-1}$ as the Cholseky factorization. Applying the transform to (1), we can write:

$$\underline{\underline{\Gamma}} \breve{q}_t = \sum_{i=1}^{N_G} \underline{\underline{\breve{G}_i}} \underline{\underline{\Gamma}} \breve{q}_{t-\tau_i} + \underline{\underline{\breve{C}}} \vec{\eta}_t. \tag{36}$$

Rewriting, we have:

$$\breve{q}_t = \sum_{i=1}^{N_G} \underline{\underline{\Gamma}}^{-1} \underline{\underline{\breve{G}_i}} \underline{\underline{\Gamma}} \breve{q}_{t-\tau_i} + \underline{\underline{\Gamma}}^{-1} \underline{\underline{\breve{C}}} \vec{\eta}_t. \tag{37}$$

By inspection, we see that the correction transforms that recover $\langle \vec{q}_t \vec{q}_t{}^T \rangle = \underline{\underline{I}}$ from $\langle \breve{q}_t \breve{q}_t{}^T \rangle \approx \underline{\underline{I}}$ are:

$$\underline{\underline{\breve{G}_i}} \rightarrow \underline{\underline{\Gamma}} \underline{\underline{\breve{G}_i}} \underline{\underline{\Gamma}}^{-1} \tag{38}$$

$$\underline{\underline{\breve{C}}} \rightarrow \underline{\underline{\Gamma}} \underline{\underline{\breve{C}}} \tag{39}$$

These transforms give Eq. (37) the same form as (1). The transform also changes the lag correlation structure, but in a small way that is necessary to meet the $\langle \vec{q}_t \vec{q}_t{}^T \rangle = \underline{\underline{I}}$ constraint. We apply these transforms iteratively, each time recomputing $\underline{\underline{A}}$, $\underline{\underline{B}}$, cov $\vec{x}_t$, and $\underline{\underline{\Gamma}}$ until the diagonal elements of $\langle \breve{q}_t \breve{q}_t{}^T \rangle$ are within 10^{-6} of unity.

At this point, we have a refined set of preliminary AR coefficients that give $\langle \vec{q}_t \vec{q}_t{}^T \rangle = \underline{\underline{I}}$ to the desired precision on the fiducial time tags (without time interpolation) and approximate the spatiotemporal covariance at multiple time lags. Next, we must address the effect of time interpolation.

3.4

3.4 Refining AR Coefficients for Realistic Time Interpolation

In this section, we concern ourselves with the variance reduction inherent in temporal interpolation. Specifically, taking the variance of (2), we have:

$$\langle \vec{q}_{t_k'} \vec{q}_{t_k'}{}^T \rangle = {c_k}^2 \langle \vec{q}_{t_j} \vec{q}_{t_j}{}^T \rangle + c_k(1-c_k) \langle \vec{q}_{t_{j+1}} \vec{q}_{t_j}{}^T \rangle$$

$$+(1-c_k)c_k \langle \vec{q}_{t_j} \vec{q}_{t_{j+1}}{}^T \rangle + (1-c_k)^2 \langle \vec{q}_{t_{j+1}} \vec{q}_{t_{j+1}}{}^T \rangle \tag{40}$$

This reduces to:

$$\langle \vec{q}_{t_k'} \vec{q}_{t_k'}{}^T \rangle = (1 - 2c_k + 2{c_k}^2)\underline{I} + (c_k - {c_k}^2)\left(\underline{\underline{R}}_1 + \underline{\underline{R}}_1{}^T\right). \tag{41}$$

We obtain $\underline{\underline{R}}_1$ from the first off-diagonal block in $\operatorname{cov} \vec{x}_t$ in Eq. (33). Ideally, over the life of a scenario, c_k varies almost uniformly between 0 and 1 since the user's time tags are unrelated to the Monte Carlo fiducial times. In that case, we can take the average over time (k) to obtain:

$$\langle \vec{q}_{t_k'} \vec{q}_{t_k'}{}^T \rangle = \tfrac{2}{3}\underline{I} + \tfrac{1}{6}\left[\underline{\underline{R}}_1 + \underline{\underline{R}}_1{}^T\right] \tag{42}$$

In this ideal case, we can again adjust the AR coefficients to obtain $\langle \vec{q}_{t_k'} \vec{q}_{t_k'}{}^T \rangle = \underline{I}$. The adjustment is:

$$\vec{q}'_t = \underline{\underline{H}} \vec{q}_t \tag{43}$$

$$\underline{\underline{G_j}} = \underline{\underline{H}}\,\underline{\underline{\check{G}_j}}\,\underline{\underline{H}}^{-1} \tag{44}$$

$$\underline{\underline{C}} = \underline{\underline{H}}\,\underline{\underline{\check{C}}}, \tag{45}$$

where

$$\underline{\underline{H}}\left[\tfrac{2}{3}\underline{I} + \tfrac{1}{6}\left(\underline{\underline{R}}_1 + \underline{\underline{R}}_1{}^T\right)\right]\underline{\underline{H}}^T = \underline{I}, \tag{46}$$

so that

$$\underline{\underline{H}}^{-1}\underline{\underline{H}}^{-1^T} = \tfrac{2}{3}\underline{I} + \tfrac{1}{6}\left(\underline{\underline{R}}_1 + \underline{\underline{R}}_1{}^T\right). \tag{47}$$

The time evolution equation for $\vec{q}'_t$ then resembles (1):

$$\vec{q}'_t = \sum_{i=1}^{N_G} \underline{\underline{G_i}} \vec{q}'_{t-\tau_i} + \underline{\underline{C}} \vec{\eta}_t. \quad (48)$$

We recognize that its covariance is

$$\langle \vec{q}'_t \vec{q}'^{\,T}_t \rangle = \underline{\underline{H}} \underline{\underline{H}}^T.$$

However, when we interpolate it onto time steps that fall at random offsets relative to the fiducial time steps, we recover $\langle \vec{q}'_{t'_k} \vec{q}'^{\,T}_{t'_k} \rangle = \underline{\underline{H}} \langle \vec{q}_{t'_k} \vec{q}^{\,T}_{t'_k} \rangle \underline{\underline{H}}^T = \underline{\underline{I}}$.

The critical assumption when correcting for interpolation is that c_k is uniformly distributed between 0 and 1. If, in fact, c_k has some other distribution, then we have:

$$\langle \vec{q}_{t'} \vec{q}^{\,T}_{t'} \rangle = (1 - 2\langle c \rangle + 2\langle c^2 \rangle) \underline{\underline{I}} + (\langle c \rangle - \langle c^2 \rangle) \left(\underline{\underline{R_1}} + \underline{\underline{R_1}}^T \right). \quad (49)$$

If $\langle c \rangle = 1/2$ and $\langle c^2 \rangle = 1/3$, then (49) reduces to (42), and the $\underline{\underline{H}}$ correction is still adequate. However, there are obvious cases where these conditions on *c* are not held. For example, for a scenario that starts at midnight and has a Monte Carlo time step of one day, sampling a geostationary orbit at 1-hour resolution will give c_k values of 0, 1/24, 2/24, … 23/24 every day. For such a case, the time average of c_k is, $\langle c \rangle \sim 0.48$ and $\langle c^2 \rangle \sim 0.31$, thus introducing a few percent defect in the variance of interpolated principal component amplitudes. If, instead, the user's samples are taken at the half hour, then c_k values of 1/48, 3/48, … 47/48, and the defect is only a few parts in 10^4.

An error of a few percent in the variance of the principal components can translate into a larger error for individual fluxes on the model grid. A typical use case involves combining many such fluxes, of which only a small fraction are likely to be dramatically affected. Still, we can employ one more strategy to minimize the impact of these pathological (but still likely) user time tags: we can start each Monte Carlo scenario with a slightly different initial time. If the initial time is offset from the epoch time by a random fraction of the fiducial time step, then we can guarantee that, at least across scenarios, the conditions on $\langle c \rangle$ and $\langle c^2 \rangle$ will be met. Further, because the Monte Carlo scenarios are used as ensembles, the net effect of pathological user time stepping, after mitigation by the random initial time, is to broaden the distribution of flux and fluence. That broadening is, in effect, conservative in the typical use of designing to a high percentile (e.g., the 95th percentile). That broadening can be reduced to an arbitrarily small value by shrinking the user's time step (e.g., reducing it from1 hour to a more typical 5 minutes or 10 seconds).

Another strategy to account for time interpolation is to examine the user's requested time tags at run time and compute $\underline{\underline{H}}$ on the fly. This approach is robust if the user is making an entire long run in a single call to the AE9/AP9 program. However, if the user is splicing together several runs, that may confound the runtime computation of $\underline{\underline{H}}$. Therefore, this alternate strategy is not used.

4. Initializing the Monte Carlo state

In this section, we describe the new initialization of the Monte Carlo state for v1.3. In v1.0 through v1.2, AE9/AP9 scenarios initialized the state fiducial time to the user-supplied epoch minus a conditioning time, usually many months or years. The conditioning time was derived from the largest eigenvalue of the $\underline{\underline{A}}$ matrix in Eq. (31). At that time, the stored history of $\vec{q}_t$, denoted above by $\vec{x}_t$, was initialized to a set of independent random Gaussian variables with zero mean and unit variance. By advancing $\vec{q}_t$ forward in time over an interval equal to the conditioning time, it was thought that the resulting history $\vec{x}_t$ would evolve to a reasonable state consistent with having evolved through $\underline{\underline{G_i}}$ and $\underline{\underline{C}}$ for an infinitely long time. In v1.3, we change both how we set the initial fiducial time and how we initialize the history $\vec{x}_t$.

As mentioned in section 3.4, starting with v1.3, we must set the Monte Carlo state's initial fiducial time (t_0) to a random offset prior to the user-specified epoch (t_e). We do this by scaling the Monte Carlo time step using a random variable u sampled uniformly from 0 to 1. We offset the initial time *before* the epoch because the Monte Carlo process can only move forward, and the user's time tags often start at the epoch time (it is an error for the user to request a time tag before the epoch). The Monte Carlo state's initial fiducial time is:

$$t_0 = t_e - u\delta t \tag{50}$$

This offset causes each interpolation weight c_k in (4) to be different for each scenario and to be uniformly distributed from 0 to 1 across scenarios, as required by (42).

Starting with v1.3, we also replace the conditioning time approach with an initialization based on the computed covariance of $\vec{x}_t$. This change allows us to initialize the entire history of $\vec{q}_t$ in one operation to a realistic $\vec{x}_t$. To create a set of Gaussian random variables with zero mean and a known covariance ($\mathrm{cov}\,\vec{x}_t$), one multiplies the square root $\underline{\underline{W}}$ of the covariance matrix by a vector $\vec{\delta}$ of independent Gaussian variables with zero mean and unit variance:

$$\underline{\underline{W}}\underline{\underline{W}}^T = \mathrm{cov}\,\vec{x}_t \tag{51}$$

$$\vec{x}_0 = \underline{\underline{W}}\vec{\delta} \tag{52}$$

We note that for the calculation of $\underline{\underline{W}}$, $\mathrm{cov}\,\vec{x}_t$ is computed from the final $\underline{\underline{G_i}}$ and $\underline{\underline{C}}$ values using (33) after all the corrections are applied. Starting with v1.3, the matrix $\underline{\underline{W}}$ is included in the AE9/AP9 runtime tables data files.

To ensure that scenarios are reproducible, the random values u and $\vec{\delta}$ used in the initializations are tied to each scenario via a scenario-specific seed to the random number generator.

5. Stability and Accuracy Tests

The equations for computing the AR coefficients are set up to produce a Monte Carlo process that is stable over time and which has the desired mean and variance (after interpolation). As part of the computation, we perform stability and accuracy tests. We perform these tests inside a loop, where each time the process fails one of the tests, the loop starts again with an adjustment to the observed correlation coefficients to taper long lag correlations. Specifically, the loop begins at Eq. (13), and tapering involves replacing every $\underline{\underline{R}}_M$ with $\underline{\underline{R}}_M \gamma^M$, where γ starts out as 1 on the first pass through the loop, and is scaled down by 0.999 each successive retry until the process passes all the tests. This correlation tapering is required because sometimes observed correlations do not decay quickly enough to give a consistent AR model with only the specified lags. This arises from statistical fluctuations in the observed correlations and from the fact that we have chosen not to include every consecutive lag in (1).

We employ two methods to test whether the Monte Carlo process is stable. At various stages in the development of the $\underline{\underline{G}}_i$ and $\underline{\underline{C}}$ matrices, we perform a test for the largest eigenvalue of $\underline{\underline{A}}$ from Eq. (31). As noted above, the process is stable if the largest eigenvalue of $\underline{\underline{A}}$ has magnitude less than 1. Because the radiation belt may have very long lags, we expect eigenvalues of $\underline{\underline{A}}$ to be near 1. Therefore, it helps to stabilize the eigenvalue calculation by multiplying $\underline{\underline{A}}$ by itself several times. If the leading eigenvalue has magnitude $(1 + \delta)$, where $|\delta| \ll 1$, then the leading eigenvalue of $\underline{\underline{A}}^n$ is $(1 + \delta)^n \approx 1 + n\delta$. That is, the magnitude drifts away from 1 as the power increases. We have implemented the eigenvalue test by trying successive integer powers of $\underline{\underline{A}}$ until the eigenvalue algorithm returns an accurate result. If the eigenvalue is larger than 1, then we start the loop again.

Once we have a solution that passes the eigenvalue test, we perform a final stability check by evolving the principal components through 50 years and 100 scenarios. If two or more scenarios produce unlikely large values of $\vec{q}_t$ at the end of 50 years, the test fails, and we begin the loop again. This second test ensures that any latent numerical issues in the calculation of the AR coefficients will not lead to unrealistic Monte Carlo states.

The next two tests are performed on another long simulation, with 1000 time points randomly spaced over a 50-year period and 100 scenarios. The resulting time series includes time interpolation effects. From this simulation, we confirm that, over time, 99 times the variance of principal components across scenarios follows approximately a chi-squared distribution with 99 degrees of freedom. This is the distribution one expects for the sample variance of 100 samples from a Gaussian variable with variance 1. For each principal component (there are ~10 of these), we have 1000 time steps of the sample variance across scenarios. The distribution of these 1000 variances will not exactly follow the chi-squared distribution because of temporal correlation in the Monte Carlo process. However, we require that they have a cumulative distribution that never deviates more than 10% from the ideal, which means they closely approximate the expected chi-squared. Similarly, we compute the variance

over time for each scenario and for each principal component. We confirm that the average across scenarios of this variance is within 10% of 1 for all principal components. We cannot perform a more rigorous test because of the temporal correlation effects. If either of these two variance tests fails, we begin the loop again with more tapering.

6. Recommendations for New Test of the Monte Carlo Machinery

We recommend several new Monte Carlo tests aimed at uncovering any other latent inadequacies in the Monte Carlo formulation.

6.1 Confirm that Monte Carlo Fluence Nearly Matches Perturbed Mean Fluence After 10 Years

To accomplish this goal, we will run GEO, GPS, POES, GTO, and HEO orbits for 10 years in 40 Monte Carlo scenarios and in 40 perturbed mean scenarios. We will examine how the 50^{th}, 75^{th}, 90^{th}, and 95^{th} percentile fluence after 10 years changes between the Monte Carlo (MC) and perturbed mean (PM) scenarios. We will perform this comparison for integral and differential proton and electron fluence as well as for ionizing dose and equivalent neutron fluence. Specifically, we will plot the ratio of the fluence percentile from the Monte Carlo runs to the corresponding fluence percentile for the perturbed mean runs as a function of particle energy or depth of shielding. Figure 1 shows an example comparison for a GPS orbit. For this example, the largest difference between any percentile in the MC versus PM calculation is about 20%. Across all five sample orbits, the largest deviation is about 30%. These differences are acceptable: they are smaller than the error in the percentiles due to using only 40 scenarios, and it can take longer than 10 years for the dynamics to truly average out in the MC scenarios.

6.2 Confirm That the Statistical Distribution in Monte Carlo Runs Approximately Matches Observed Statistical Distributions

Using the same set of reference orbits, we will make quantile-quantile (QQ) plots of the statistical distribution of each scenario against the observed statistical distribution. We will use HEO and GEO data as the reference for this test.

6.3 Confirm Lag Correlation at Fixed Energy and Location for Electrons Approximately Matches Observed Values

To perform this test, we will use HEO and GEO data. The HEO data will be binned into daily averages in Φ bins. The GEO data will be simple daily averages.

6.4 Create a Reference Set of Engineering Specifications So That Each New Revision of the Model Can Be Compared in the Way an Engineer Would See the Results

For each of the five reference orbits listed above in 6.1, we will create the following specifications using the 10-year Monte Carlo runs:

- 1, 5, 10-year 95^{th} percentile dose vs depth
- 1, 5, 10-year 95^{th} percentile integral proton fluence
- 1, 5, 10-year 95^{th} percentile integral electron fluence
- 1, 5, 10-year 95^{th} percentile integral electron 24-hour worst-case boxcar average
- 1, 5, 10-year 95^{th} percentile integral proton 1-minute worst-case boxcar average

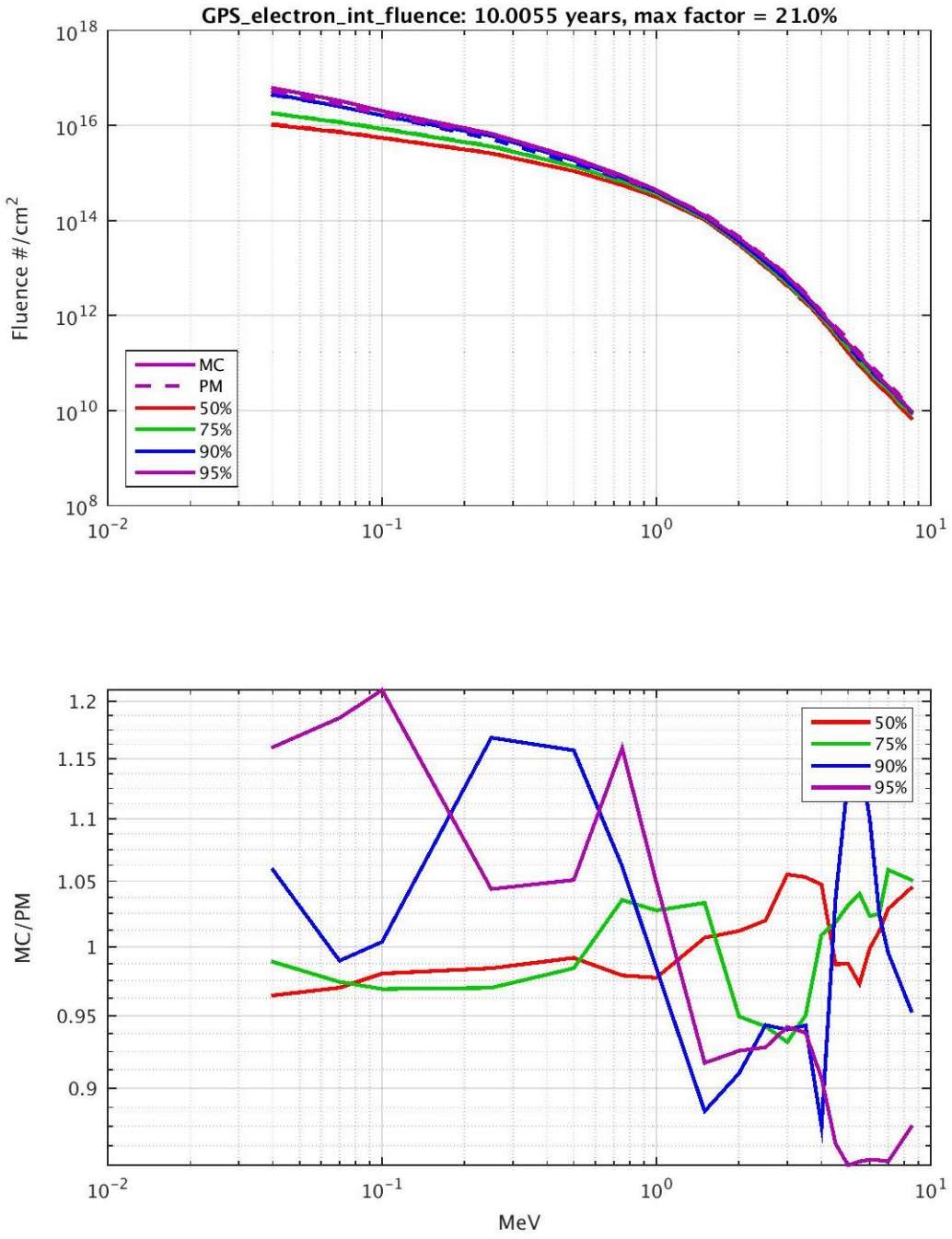

Figure 1. Comparison of Monte Carlo and Perturbed Mean fluences. Top: Monte Carlo (MC) and perturbed mean (PM) fluence spectra for 10 years in a GPS orbit at different confidence levels. Bottom: Ratio of MC to PM fluence at different confidence levels versus energy.

The first three of these specifications address cumulative effects, like total dose, displacement damage, or fatal single-event effects. The fourth addresses internal charging, and the fifth addresses single-event effects. Each test consists of three curves that can be combined into one plot, and the same curves for the most recent prior AE9/AP9 release can also be made. Figure 2 shows an example internal charging specification for a GPS orbit. The improved Monte Carlo simulations in v1.3 produce a 30–100% higher worst-case internal charging flux for a 10-year mission, depending on energy, evidence that the older version was missing some of the flux variance.

Together, these tests will help us identify any shortcomings of the Monte Carlo approach, and they will better prepare us to anticipate concerns from engineers about changes between versions of the model.

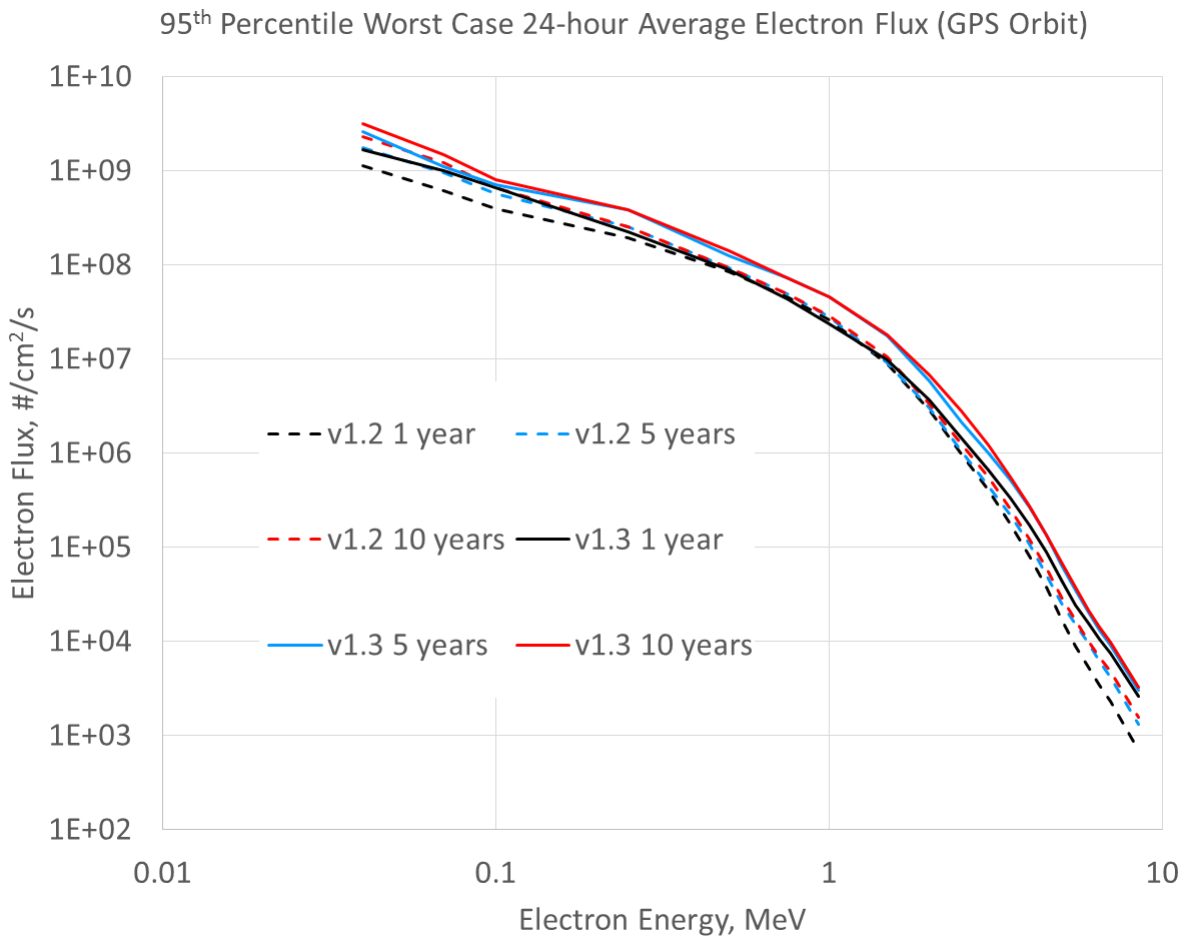

Figure 2. Example internal charging specification for a GPS orbit for different mission durations, comparing v1.2 and v.13

7. References

Ginet, G. P. et al., "AE9, AP9 and SPM: New models for specifying the trapped energetic particle and space plasma environment," *Space Sci. Rev.*, doi:10.1007/s11214-013-9964-y, 2013

Neumaier, A. and T. Schneider, "Estimation of parameters and eigenmodes of multivariate auto-regressive models," *ACM Trans. Math. Soft.,* **27**(1), 27-57, 2001.

O'Brien, T. P., *Adding Multiple Time Lags to AE9/AP9 V1.0*, TOR-2012(1237)-3, The Aerospace Corporation, El Segundo, CA, August 10, 2012 (corrected June 2015).

O'Brien, T. P., *AE9/AP9/SPM V1.0 Runtime Algorithms*, TOR-2014-00294, The Aerospace Corporation, El Segundo, CA, December 30, 2013a.

O'Brien, T. P., *Generation of AE9/AP9/SPM V1.0 Runtime Tables*, TOR-2014-00295, The Aerospace Corporation, El Segundo, CA, December 30, 2013b.