

Generation of AE9/AP9/SPM V1.0 Runtime Tables

December 30, 2013

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

Prepared for:

Air Force Research Laboratory
Kirtland AFB NM 87117-5776

Contract No. FA8802-09-C-0001

Authorized by: Engineering and Technology Group

Distribution Statement A: Approved for public release; distribution is 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:

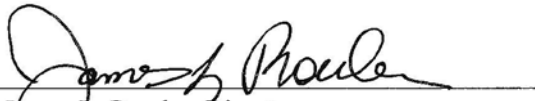
Electronics and Photonics Laboratory: Microelectronics, VLSI reliability, failure analysis, solid-state device physics, compound semiconductors, radiation effects, infrared and CCD detector devices, data storage and display technologies; lasers and electro-optics, solid-state laser design, micro-optics, 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.

Generation of AE9/AP9/SPM V1.0 Runtime Tables

Approved by:



James L. Roeder, Director
Space Sciences Department
Space Science Applications Laboratory
Physical Sciences Laboratories

© The Aerospace Corporation, 2013.

All trademarks, service marks, and trade names are the property of their respective owners.

Abstract

This report describes the algorithms employed to generate the AE9/AP9 V1.0 runtime data tables. These algorithms begin with time series in-situ particle data that have already been cleaned and calibrated. The algorithms produce maps of the radiation and plasma environment as a function of magnetic field coordinates, matrices needed to perturb those maps to represent uncertainty, and matrices needed to evolve dynamic (Monte Carlo) scenario environments in time.

Acknowledgments

The author acknowledges many useful discussions with the other members of the AE9/AP9 team at AFRL, MIT/LL, and AER. Matlab is a trademark of The Mathworks Inc.

This report will be incorporated into a larger omnibus documentation package, which will be issued as an AFRL technical report entitled “AE9/AP9/SPM V1.1 Radiation Environment Model: Technical Documentation,”

Contents

1.	Introduction.....	1
2.	Generating Flux Maps.....	3
2.1	Loading and Filtering (prepdata).....	3
2.2	Defining the Bins (definebins).....	4
2.3	Binning (assign2bins).....	5
2.4	Averaging (binavg).....	5
2.5	Computing Theta (calctheta).....	6
2.6	Filling in Each Sensor with Each Template (fill_maps).....	8
2.7	Combining Filled-in Maps and Estimating Global Errors (combinetheta).....	9
3.	Time Evolution Matrices.....	14
3.1	Computing Spatial (calccov covs) and Lag Correlations (calccov lagcovs).....	15
3.2	Filling in the Spatial Covariance Matrix (makecov cov).....	15
3.3	Computing Principal Components (makeQ).....	16
3.4	Filling in Lag Correlation Matrices (makecov lagcov).....	17
3.5	Computing Time Evolution Matrices (buildmc).....	17
4.	Diagnostic Figures (figs).....	18
	References.....	19
	Appendix A—Templates Used in Each Model.....	20
	Appendix B—Smoothing parameters.....	22

Figures

1. Graphical depiction of the "Turnkey System" that generates the runtime tables from cleaned and calibrated time series data.....	2
2. A graphical depiction of the spatial filling technique.....	10
3. A "binspectra" figure for AE9V10 on the K- Φ grid.....	13

1. Introduction

The AE9/AP9 runtime tables are generated by what is known within the team as the “turnkey system.” The turnkey system consists of a set of Matlab™ codes and scripts to run them. These codes progress through a series of stages in the processing from cleaned and calibrated time series data to the data tables used at run time by the AE9/AP9 library and application. The turnkey system is run separately for each of the various models: AE9V10 (electron radiation), AP9V10 (proton radiation), SPMEV10 (electron plasma), SPMHV10 (proton plasma), SPMHEV10 (helium plasma), SPMOV10 (oxygen plasma).

The stages of the turnkey system are:

prepdata	Read in the data from each sensor and, if necessary, apply flags and break up into daily, sorted time series files.
definebins	Define the coordinate bins to be used (e.g., E , K , Φ).
assign2bins	Bin the time series data from each sensor into coordinate bins.
binavg	Compute time averages (e.g., 24 hours or 7 days) within bins.
calctheta	Compute θ_1 and θ_2 (statistical moments/percentiles in each bin) from binavg results.
fill_maps	Use templates to fill in gaps in theta maps for each sensor.
combinetheta	Combine filled, sensor-specific theta maps into one consensus map with errors (Stheta).
<i>calccov covs</i>	Compute a set of spatial covariances at zero lag.
<i>calccov lagcovs</i>	Compute a set of spatial covariances at various time lags.
<i>makecov cov</i>	Use calccov results to fill in spatial covariance matrix.
<i>makeQ</i>	Compute principal components.
<i>makecov lagcov</i>	Use calccov and makeQ results to fill in spatial covariance matrices at various time lags.
buildmc	Compute the final Monte Carlo scenario quantities, if needed, and save the runtime tables to a single file (e.g., AE9V10_runtime_tables.mat).
figs	Generate an extensive set of diagnostic figures.

The turnkey process is depicted graphically in Figure 1. The items above in *italics* are not applicable to the plasma models, which do not have Monte Carlo scenario capabilities. Each stage in the processing is controlled by a set of “global” variables that describe the model and set parameter values. Thus, it is possible to restart the entire processing chain with a single command, and it will run unattended through to completion. However, as the algorithms themselves were often being updated dur

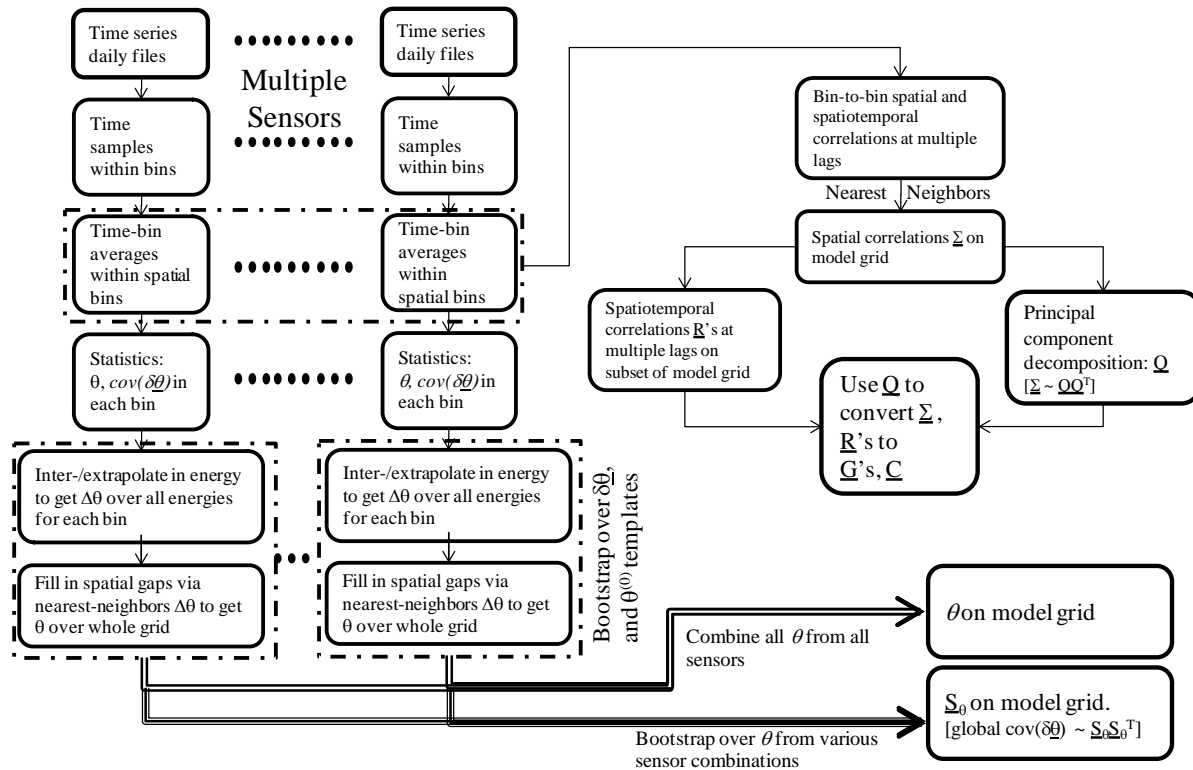


Figure 1. Graphical depiction of the "Turnkey System" that generates the runtime tables from cleaned and calibrated time series data.

ing the V1.0 development process, it is also possible to run only part of the turnkey process, starting and stopping at an arbitrary stage using partial results from a prior run.

It is important to note, since it will come into play later in this document, that most of the models assume (based on visual inspection) that the long-term statistical distribution of data within a given spatial bin has a log-normal distribution. However, for the energetic electrons in AE9, the shape is assumed to be Weibull. The properties of these distributions will be discussed in more detail in later sections.

2. Generating Flux Maps

A flux map is a tabular representation of the statistical properties of the flux in each coordinate bin. In AE9/AP9 V1.0, the flux map has two parameters at each location, which are expressed in terms of the median (m_{50}) and 95th percentile (m_{95}) flux within the bin:

$$\theta_1 = \ln m_{50} \quad (1)$$

$$\theta_2 = \ln(m_{95} - m_{50}) \quad (2)$$

This transformation from flux percentiles to θ guarantees that, for any real values of θ_1, θ_2 , the fluxes will satisfy $m_{95} > m_{50} > 0$. Further, for $m_{95} \gg m_{50}$, as is often the case, θ_2 becomes approximately the natural log of m_{95} . Thus, a Gaussian distribution of errors in θ translates to approximately log-normal errors in m_{50} and m_{95} . Log-normal errors are common in space particle fluxes, and so we assume the errors in θ are Gaussian. The errors in θ are given by $cov(\delta\theta)$, which is the local 2x2 matrix that represents the uncertainty in θ as measured by a single sensor on a single spacecraft. The entries in this matrix shrink with the square root of the sample size within the bin, and the off-diagonal term reflects the fact that errors in θ_1 and θ_2 are necessarily correlated. When we combine data across sensors, this $cov(\delta\theta)$ will feed into how we combine them and into the global error representation S_0 .

For each sensor, we generate a separate tabulation of θ and $cov(\delta\theta)$ on the spatial grid (a flux map). That process involves several steps.

2.1 Loading and Filtering (prepdata)

The first step in generating a flux map is reading the time series data, filtering it and reorganizing as needed, and saving it into a file with a standard structure. The filtering typically involves removing times flagged for possibly having contamination (e.g., from solar energetic particles), and possibly applying *ad hoc* filters. *Ad hoc* filters originated because it was technically less cumbersome to include a few last-minute filters to the data at the start of the turnkey process rather than to continually generate new versions of the turnkey input data files. Reorganizing the data involves unwrapping an angular axis, if one is present, so that the flux table has only two dimensions: time and energy. Further, variable name translation may be necessary, such as converting “Hmin” to “h_min,” etc.

At the completion of the prepdata step, each sensor is organized into daily files of filtered, calibrated, and cleaned particle flux, as well as associated coordinate and error information [see Ginet et al., 2013, AE9/AP9 Tech Doc, Guild et al., 2009, and O’Brien 2012b]. The flux errors are provided in terms of the standard deviation of the natural log of flux, either using an *ad hoc* value or as a result of an inversion and/or intercalibration.

2.2 Defining the Bins (definebins)

Before we can bin the data, we must define the bins, specifically, the bins in the 2nd and 3rd coordinates (such as K , Φ and others described in Ginet et al. [2013]). The data are not binned in energy; rather, they are handled at their original energy channels provided to the turnkey system and will be interpolated onto the model energy grid at a later stage. The coordinate grid itself defines the nominal spatial bins. However, because the coordinate grid is rectangular, it is often the case that a large portion (half) of the nominal grid falls inside the loss cone. Therefore, the definebins step not only describes the bin limits for each spatial bin, it also applies a loss cone filter to remove bins that fall in the loss cone.

For example, the AP9V10 high-altitude grid (K - Φ) uses a loss cone defined by the maximum value of K as a function of Φ . This maximum value is derived from the largest K at a given Φ that corresponds to an h_{min} of 200 km (that is, the K for which the particle's drift-bounce orbit eventually intersects the atmosphere at 200 km geodetic altitude). The maximum K thus defined is given by a polynomial in Φ (rounded off):

$$\text{Log}_{10} K = -21.8 135.0\Phi^7 + 135.0\Phi^6 - 342.0\Phi^5 + 458.0\Phi^4 - 348.0\Phi^3 + 150.8\Phi^2 - 36.1\Phi + 4.7 \quad (3)$$

For K in $G^{1/2}R_E$ and Φ in $G R_E^2$.

Bins whose center falls in the loss cone are removed from further consideration. Bins that partially overlap the loss cone are limited such that the bin edges in the second coordinate (e.g., K) are entirely outside the loss cone. Equivalently, bins whose upper or lower edges fall outside the valid range for one of the coordinates will have the edge adjusted to be at the limit of the valid range (e.g., $K < 0$ is replaced by $K = 0$).

A bin consists of a lower, middle, and upper value for each spatial coordinate, and the lower and upper values typically split the difference between adjacent bin centers. The “full” grid of bins spans the full range of the second and third coordinates, and is thus referred to as the full Q_2 - Q_3 grid. It can be referenced either by the row/column type subscripts or by a 1-D index that unwraps the 2-D grid onto a 1-D list. The “reduced” grid is only referenced by a 1-D index, which spans only the N_{red} bins that are not in the loss cone. Tables *ired2full* and *ifull2red* are used to convert 1-D indices between the two grids. Matlab's *sub2ind* and *ind2sub* routines are used to convert between 2-D subscripts and 1-D indices. In a table with N_1 rows and N_2 columns the coordinate (i_1, i_2) maps to a 1-D coordinate j according to:

$$j = N_1(i_2 - 1) + i_1. \quad (4)$$

That is, Matlab uses a 1-based, column-major indexing system, and therefore, so does AE9/AP9.

An additional consideration is that AE9 and AP9 support two different grids, identified in the code as subgrids. The calculations described in this section are applied separately to the subgrids and combined only at the very end of generating the runtime tables.

2.3 Binning (assign2bins)

The assign2bins step is fairly straightforward: each spectrum from a given sensor at the original time resolution is assigned to a bin in the two spatial coordinates. The result is a set of files, each one containing all the time samples from a specific sensor in a specific bin. The files are identified by the model, sensor, subgrid, and the reduced grid index for the bin.

For the AE9 and AP9 models, there are two subgrids, so the assign2bins step is done twice, once for each subgrid. Because the subgrids overlap, some time samples may appear once in each subgrid. This is allowed because it is believed to help ensure a smooth interface of the two grids. It does not lead to over-counting since at runtime the grids are spliced, not summed.

2.4 Averaging (binavg)

Time averaging is used to suppress (and sometimes assess) random measurement errors, and to put the data from different sources on a consistent time index.

First, the data are separated into “passes” within a bin. A pass is defined as any set of time samples in the bin that have no temporal gaps longer than 15 minutes. This definition allows for vehicles that physically move in and out of a bin, but also for spinning sensors where the field of view moves in and out of the bin. Successive spins would usually count as a single pass through the bin. Data within each pass are combined using a weighted flux average.

The weighted flux average is a bit complicated because it combines two different approaches to computing the error in the average. If the raw flux data is marked as x with error dx (which is the standard error of the natural log flux, sometimes denoted $dlnj$ or $dlogflux$) and the average is given as y with error dy , then the weighted flux average

$$y = \frac{\sum_{i=1}^N \frac{x_i}{dx_i^2}}{\sum_{i=1}^N dx_i^{-2}} \quad (5)$$

$$dy = \max \left(\sqrt{\frac{\sum_{i=1}^N \frac{x_i^2}{dx_i^2}}{y^2 \sum_{i=1}^N dx_i^{-2}} - 1}, \left(\frac{1}{N} \sum_{i=1}^N dx_i^{-2} \right)^{-1/2} \right) \quad (6)$$

The resulting dy is further limited to be between 0.1 and $\ln(10)$, which is to say that the resulting uncertainty is constrained to be between 10% and a factor of 10. Note that we do not apply a square-root-N factor to shrink dy . We experimented with this, but determined that there was sufficient correlation within a single pass that the square-root-N was not justified.

After the pass averaging, we then proceed to average all the passes within time bins, using the same formulas used for pass averaging. For AP9, the time bin size is 1 week; for all other models, it is one day.

2.5 Computing Theta (calctheta)

We require at least 50 time-averages for a given energy channel for a given sensor in a bin to proceed with the calculation of θ . Otherwise, the bin is designated as empty for the particular sensor and channel (we will fill in these empty bins by interpolation and extrapolation in a later step).

In the alpha and beta versions of AE9/AP9, the calculation of theta (θ) was fairly straightforward: the median and 95th percentiles were calculated directly from the averaged samples within bins. That approach guarantees good agreement between the model and the data at the 50th and 95th percentiles. However, a desire to guarantee good agreement with the *mean* and the 95th percentile motivated a change to a more indirect approach.

The approach used in V1.0 fits θ_1 and θ_2 using one of three approaches depending on the model. In all models, the θ fit must match the observed 95th percentile in the bin (m_{95}). For the plasma model, the θ fit must also match the observed mean $\langle x \rangle$ in the bin. For AE9 and AP9, the θ fit must match $\langle x \rangle_{95}$ the mean of the flux up to the 95th percentile. We chose $\langle x \rangle_{95}$ because, in the radiation belt datasets, we are still concerned that the tail of the distribution might be affected by instrumental effects. To do the fitting, we need several new relationships. For the log Normal, we have:

$$P[X < x] = \Phi\left(\frac{x-\mu}{\sigma}\right) \quad (7)$$

$$\Phi(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^z e^{-t^2/2} dt \quad (8)$$

$$\langle x \rangle = \exp\left(\mu + \frac{\sigma^2}{2}\right) \quad (9)$$

$$\mu = \ln m_{50} \quad (10)$$

$$\sigma = \frac{\ln m_{95}/m_{50}}{\Phi^{-1}(0.95)} = \frac{\ln m_{95} - \mu}{\Phi^{-1}(0.95)} \quad (11)$$

$$\langle x \rangle_{95} = \frac{\sum_{x_i < m_{95}} x_i}{\sum_{x_i < m_{95}} 1} = \frac{\exp(C)\Phi(v_{95})}{0.95} \quad (12)$$

$$C = \frac{1}{2}\left(a^2 - \left(\frac{\mu}{\sigma}\right)^2\right) \quad (13)$$

$$a = \sigma + \frac{\mu}{\sigma} \quad (14)$$

$$v_{95} = \ln m_{95} - a \quad (15)$$

Here $\Phi()$ is the cumulative unit normal (standard Gaussian) distribution, with mean 0 and standard deviation 1, and $\Phi^{-1}()$ is its inverse. We begin with the observed m_{95} and an guess for $\mu = \ln m_{50}$, and

perform a 1-D optimization with respect to μ , fitting either $\langle x \rangle$ or $\langle x \rangle_{95}$, depending on the model (AP9 or a plasma model). With μ and m_{95} , we can obtain θ_1 and θ_2 directly using Eq. (10) then Eqs. (1) and (2). We will later need to transform from flux x to a unit normal z using this relation:

$$z = \frac{\ln x - \mu}{\sigma} \quad (16)$$

For AE9, we are working with a Weibull distribution, and we use a different set of formulas:

$$P[X < x] = 1 - \exp\left(-\left(\frac{x}{\sigma}\right)^\gamma\right) \quad (17)$$

$$\langle x \rangle = \sigma \Gamma(1 + 1/\gamma) \quad (18)$$

$$\Gamma(z) = \int_0^\infty t^{z-1} e^{-t} dt \quad (19)$$

$$\gamma = \frac{\ln(\ln 0.05 / \ln 0.5)}{\ln m_{95} / m_{50}} \quad (20)$$

$$\sigma = \frac{m_{50}}{(\ln 2)^{1/\gamma}} = \frac{m_{95}}{(\ln 20)^{1/\gamma}} \quad (21)$$

$$\langle x \rangle_{95} = \frac{\sum_{x_i < m_{95}} x_i}{\sum_{x_i < m_{95}} 1} = \sigma \gamma^* \left(\ln 20, 1 + \frac{1}{\gamma} \right) / 0.95 \quad (22)$$

$$\gamma^*(z, v) = \int_0^v t^{z-1} e^{-t} dt \quad (23)$$

Here $\Gamma()$ is the gamma function, and $\gamma^*()$ is the incomplete gamma function (this notation varies slightly from some conventions because we needed to distinguish between the gamma function and the statistical parameter γ). Given m_{95} , we can compute σ using Eq. (21). We then perform a 1-d search for γ in the range 0.01 to 10 so as to reproduce $\langle x \rangle_{95}$ according to Eq. (22). With γ and m_{95} , we can obtain θ_1 and θ_2 directly using Eq. (20) then Eqs. (1) and (2). For the Weibull, the conversion between x and z is:

$$\Phi(z) = 1 - \exp\left(-\left(\frac{x}{\sigma}\right)^\gamma\right) \quad (24)$$

Whichever statistical distribution we use, we compute θ for every sensor and every energy channel in every spatial bin using the entire dataset. We then compute an error covariance for θ by bootstrapping over resamples (with replacement) from the data within the bin. For each time sample in a bootstrap, we perturb the flux using a log Normal error distribution consistent with the error computed in the binavg step (d), which is the standard error for the natural log of the averaged flux). We bootstrap 200 times, with different error perturbations for the selected time averages in each bootstrap. From

these 200 bootstraps, we compute one estimate of θ . We can then compute the 2x2 error covariance for θ using these 200 bootstrap estimates. We denote this error covariance $cov(\delta\theta)$ as described in the introduction to this section. We will use θ and $cov(\delta\theta)$ in the next step, when we fill in and combine the flux maps.

2.6 Filling in Each Sensor with Each Template (fill_maps)

We fill in the gaps in the flux map from each sensor using a set of masks and templates. A mask is simply an arbitrary set of points that are manually selected for removal. Masks affect a very small number of points that did not pass visual inspection of the resulting flux maps from a prior build of the model. Masks are only used for SPMHE (helium plasma) and SPMO (oxygen plasma).

A template is a global specification of the shape of the radiation belt as a function of the model coordinates (e.g., E, K, Φ). Templates are generated by manually stitching together select datasets, models, and ad hoc extrapolations. The templates represent human input into the system, such as specifying how the spectrum should fall off past the last energy channel. Because humans may be uncertain, or different humans may disagree, there are many templates for each subgrid within each model. The templates do not specify an absolute flux level, but merely how flux varies across the model domain. The actual templates used in AE9/AP9 are described in the AE9/AP9 Tech Doc and O'Brien [2013]. The templates themselves are used only in developing the runtime tables; they are not used at runtime, and so are not distributed with the model. A list of templates used for each model is provided in Appendix A.

Templates are stored on disk in terms of the common log of flux (arbitrary units), but are used in the model as natural log flux (for compatibility with θ). The conversion is performed on load. In this report, we will denote an arbitrary template, in natural log flux, as $\theta^{(0)}$. Since there are many templates for any given subgrid, we denote a particular template as $\theta^{(0,k)}$.

Every template will be used to fill in the entire flux map for every sensor for several instantiations of errors (10 bootstraps) in the original sensor θ , represented by local $cov(\delta\theta)$.

First, at each grid point, we perturb the local 2-element θ using a random 2-element unit-normal perturbation $\underline{\eta}$, processed through a conditioning matrix \underline{s} . The conditioning matrix is nominally the square root of the local $cov(\delta\theta)$, except that the $cov(\delta\theta)$ is adjusted so that the variance (diagonal elements) are never less than 0.1^2 ; that is, the local error in θ is never allowed to be less than 10%. The perturbed θ is denoted θ' .

$$\vec{\theta}' = \vec{\theta} + \underline{s}\vec{\eta} \quad (25)$$

$$\underline{s}\underline{s}^T = cov(\delta\theta) \quad (26)$$

This process is repeated at every grid point with a unique perturbation.

From this point forward, the filling in is done separately for θ_1 and θ_2 , and it is assumed that θ_2 has the same shape on the grid as a log flux would, even though it is not exactly a log flux.

The next step is to perform a log-log interpolation from the sensor channel energies in θ' to the grid energies at each spatial grid point (e.g., for each K, Φ pair). This applies only to energy grid points that are at or between the lowest and highest sensor channel energies. The low- and high-energy extrapolations are handled separately by adjusting either tail of the template spectrum to match the lowest/highest interpolated energy grid point. When only one energy channel is present, the template spectrum at the local grid point is log-log interpolated onto that one channel, and then the entire template spectrum is adjusted up or down to match that interpolated value. In either case, the template shape is mainly used only for extrapolating the spectrum. At this point, at each spatial grid point, we either have no data or we have a complete spectrum (for the sensor we are working with). We will call this partially filled map θ'' .

Spatial interpolation and extrapolation (e.g., in K, Φ) is considerably more complicated. First, we subtract the template we are using from the θ'' to create a map of deviations:

$$\Delta\theta = \theta'' - \theta^{(0)} \quad (27)$$

Then, we step through all the spatial grid points to find cases where there is no data from the sensor we are working with. At those “gaps” we compute a composite $\Delta\theta'$, which is the average of nearby $\Delta\theta$. “Nearby” is defined in a Pythagorean sense, but on a transformed coordinate grid. Namely, grid coordinates (which are often transformed from the physical units, e.g., Φ is actually $\log_{10} \Phi$ on the grid) are rescaled to cover the range from 0 to 1. The number of nearest neighbors used is scaled to 1/7 the number of points on the reduced grid; this scale factor was arrived at through trial and error against human judgment of what was adequately smooth. The final, filled in map, is obtained by adding the template back to the composite $\Delta\theta'$.

$$\theta''' = \Delta\theta' + \theta^{(0)} \quad (28)$$

This nearest-neighbors process ensures that there are no artificial spatial features added in at the interfaces between original and gap-filled points. The spatial filling in is depicted graphically in Figure 2.

At this stage, for each sensor and for each template, we have many (10 bootstraps) completely filled in flux maps θ''' . We will next combine these together to obtain a global best estimate flux map and to obtain the new local error in that filled flux map for each sensor. The best estimate $\underline{\theta}$ for the sensor is obtained by computing an average over all templates and all bootstraps of θ''' . The local error, again denoted $cov(\delta\theta)$, is computed as the covariance over all templates and all bootstraps of θ''' . Now we have a filled-in flux map and its local error for each sensor.

2.7 Combining Filled-in Maps and Estimating Global Errors (combinetheta)

The many filled sensor-specific flux maps will be averaged together to obtain a best estimate of θ and its global error.

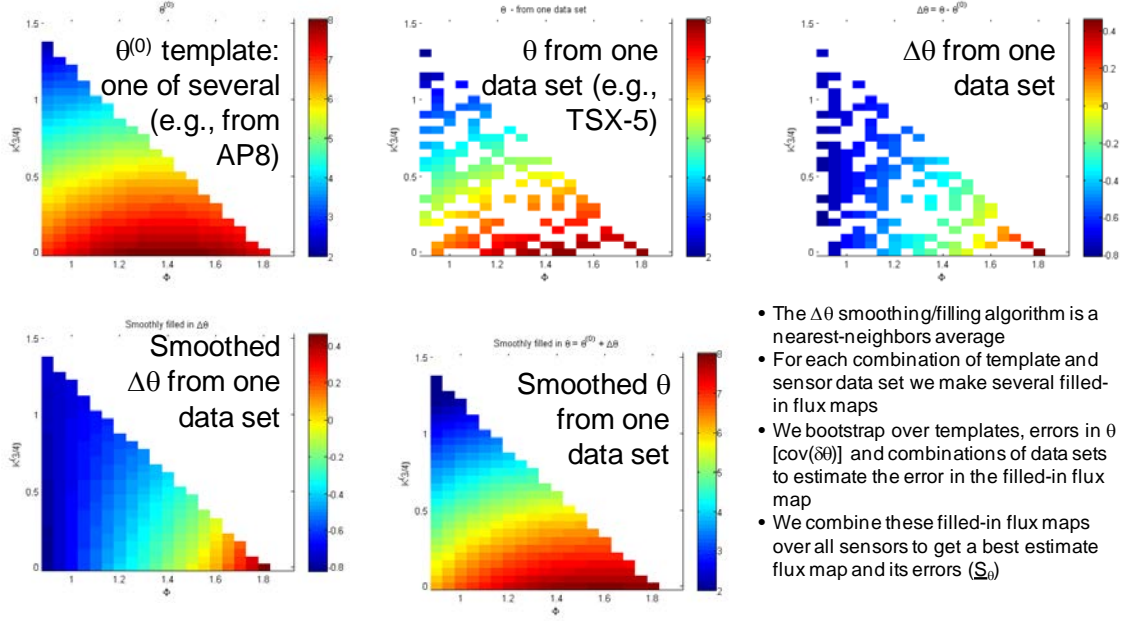


Figure 2. A graphical depiction of the spatial filling technique.

The best estimate of θ is obtained by a weighted average of the individual sensor-specific $\underline{\theta}$. The weighting is given by the reciprocal of the diagonal elements of each local $cov(\delta\theta)$ (i.e., the weighting is inverse square of the local standard error). Thus, locally, sensors with large error (due either to measurement error, poor sampling, or lots of interpolation/extrapolation uncertainty) are weighted less than those with small error.

This combination process allows some roughness, due to possible roughness in $cov(\delta\theta)$, and so an ad hoc smoothing is applied. The smoothing is defined by a 3-d smoothing mask. Smoothing parameters are given in Appendix B.

Next, for AE9 and AP9 only, we “stitch” the K - h_{min} grid and the K - Φ grid together. This stitching process finds the grid interface at $h_{min} = 1000$ km, and adjusts the θ at all h_{min} for a given E, K to match the ratio of the θ value at 1000 km between the two grids. Because the relationship between Φ and h_{min} is epoch dependent, the stitching is done based on the epoch 1-Jan-2010 00:00:00 UTC. The adjustments are linear in θ , which means they are effectively multiplicative in flux. For K - h_{min} points where the K - Φ value does not exist (i.e., it is outside the grid), the corresponding K - h_{min} points are flagged for removal. (This stitching relegates the data used to populate the K - h_{min} grid to representing only the h_{min} gradients, and it will have to be replaced with something better in future versions.)

Next, we determine the “activepoints” filter. For the plasma model, this filter simply removes any points for which there remains a bad data flag after all the filling and smoothing. For the radiation models, this filter also removes points whose median flux is less than an ad hoc factor (10^9 for AE9, 10^{10} for AP9) less than the maximum median flux anywhere on the grid. These factors were estimated from corresponding values in AE8 and AP8. This filter is applied because some of the energy extrapolations lead to fluxes that are so small as to be neglected (or, simply, not credible given the limitations of the measurements). The activepoints filter maps between the $N_{Ex}N_{red}$ set of all q and the

$N_{act} \leq N_E \times N_{red}$ domain. Arrays *active2all* and *all2active* translate between these domains in the same ways as their counterparts for the reduced grid.

Now we have our final theta (θ) for use in the runtime tables. The version of theta stored in the runtime tables is concatenated such that all elements of a subgrid are “unwrapped” to produce an $N_{act} \times 2$ matrix, and then each subgrid is stacked to produce a new, larger matrix, still with two columns.

To compute S_θ (Stheta), which represents the global errors in θ , we repeat the entire process of building the flux map 50 times, but each time we resample the set of sensors used to compute the flux map and we perturb each $\underline{\theta}$ using its local $cov(\delta\theta)$. This bootstrapping represents different combinations of sensor datasets, which captures the uncertainty that the model has with regard to adding new sensor data, and it represents the error in each dataset when combined with the others. We recognize that some sensors should be grouped together because they have the same host vehicle or are very similar in orbit, design, or calibration. The bootstrapping keeps those groups together. For example, CRRES MEA and HEEF should be kept together, as should all the LANL-GEO data. Initially, we concatenate the multiple estimates of θ into a large $(N_{act} \times 2) \times 50$ matrix $\underline{\Theta}$. We have to unwrap the individual $N_{act} \times 2$ θ matrices by stacking the two original columns into a single column of $\underline{\Theta}$. The initial estimate of S_θ is then given by:

$$\underline{S}_\theta = \frac{1}{\sqrt{49}} \left(\underline{\Theta} - \underline{\bar{\Theta}} \right). \quad (29)$$

Here $\underline{\bar{\Theta}}$ is a matrix whose columns are the average of the rows of $\underline{\Theta}$. This relation gives

$$\underline{S}_\theta \underline{S}_\theta^T = \frac{1}{49} \left(\underline{\Theta} - \underline{\bar{\Theta}} \right) \left(\underline{\Theta} - \underline{\bar{\Theta}} \right)^T, \quad (30)$$

which means that S_θ is the square root of the error covariance of θ , as estimated from 50 bootstrap samples. The runtime algorithms will use this matrix to generate perturbations on θ in order to represent uncertainty in the flux maps. Those perturbations are generated from the square root of the error covariance matrix, just as we did with \underline{s} in Eqs. (25) and (26).

Next we “clip” this initial S_θ so that no entry is larger than $\ln(X)/2/\sqrt{49}$, where $X = 100$ for AE9 and $X=10$ for all the other models. This imposes a maximum uncertainty on any flux as having a 95% confidence interval of a factor of X .

Next, we compress S_θ using singular value decomposition. Any real matrix can be decomposed into a column space (and its null space), a set of singular values, and a row space (and its null space):

$$\underline{S}_\theta = \underline{U} \underline{S} \underline{V}^T \quad (31)$$

$$\underline{U} \underline{U}^T = \underline{V} \underline{V}^T = \underline{I}. \quad (32)$$

The singular values are the diagonal elements of the diagonal matrix S . We are actually interested in the properties of $\underline{\underline{S}}_\theta \underline{\underline{S}}_\theta^T$, which is:

$$\underline{\underline{S}}_\theta \underline{\underline{S}}_\theta^T = \underline{\underline{U}} \underline{\underline{S}} \underline{\underline{S}}^T \underline{\underline{U}}^T. \quad (33)$$

We can dispense with V entirely, and we can approximate $\underline{\underline{S}}_\theta \underline{\underline{S}}_\theta^T$ by retaining only a subset of the singular values in S ; namely, we retain enough diagonal elements of S to retain 90% of the variance (the sum of the squares of the diagonals of S). If the original S_θ is $N_S \times 50$, and we retain only $N_{S'}$ singular values, then S' is $N_S \times N_{S'}$, and so is S'_θ :

$$\underline{\underline{S}}'_\theta = \underline{\underline{U}} \underline{\underline{S}}'. \quad (34)$$

Typically, this compression requires only about 10 columns in S'_θ rather than 50 in S_θ . The compression saves substantially (~80%) on disk space and memory because S_θ is by far the largest item used by the runtime tables.

Finally, we clip S'_θ again, using the same value of X as above to ensure that the compression did not introduce any new, very large errors.

This completes the calculation of the flux map θ and its global error S_θ for the runtime tables. Figure 3 shows several diagnostics of the process described above.

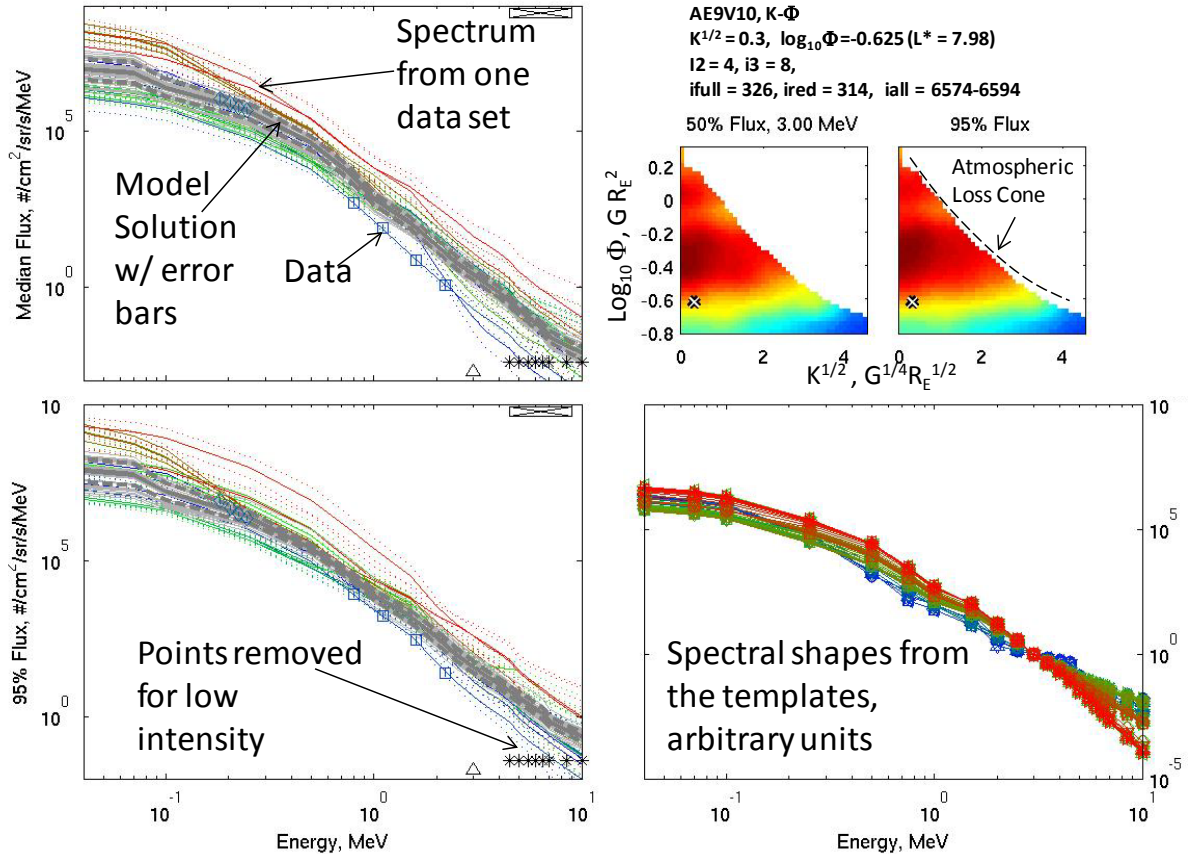


Figure 3. A "binspectra" figure for AE9V10 on the K - Φ grid. It shows the median and 95th percentile flux (left) for every sensor data set (colored thin curves) and for the model fit (thick gray curves) in the selected K - Φ bin. Error bars are shown as dotted lines. Original sensor data in the bin is shown with symbols. Energies that are removed in the bin for low flux are marked with black asterisks. On the top right is a set of bin identification information as well as color contours of the \log_{10} median and 95th percentile flux at the selected energy. The atmospheric loss cone is identified on the 95th percentile flux map. At bottom right are all the templates used, normalized to have a value of 1 at 3 MeV in this bin.

3. Time Evolution Matrices

O'Brien [2012a, hereafter TOR-2012(1237)-3] presents the equations that govern time evolution in AE9 and AP9 Monte Carlo scenarios. That evolution is controlled by a set of principal component amplitudes:

$$\vec{q}_t = \sum_{i=1}^{N_G} \underline{G}_i \vec{q}_{t-\tau_i} + \underline{C} \vec{\eta}_{t+\delta t}. \quad (35)$$

There are N_G lag persistence factors, and at each time step there is a unit Gaussian white noise innovation η (a series of uncorrelated Gaussian random variables with mean zero and unit variance). The G and C matrices condition the persistence and innovation, respectively. The τ_i are prescribed integer multiples of the fundamental time step δt .

To derive G and C according to TOR-2012(1237)-3, we need to obtain from the data spatial covariance matrix across the entire grid, and spatiotemporal covariances (lag covariances) on a decimated grid. The time lags that we will need correspond to the unique set of time offsets between every time lag used in the time evolution equation. For example, an equation that involved states at times 1, 3, and 6 would require covariances at lags of 2 (= 3-1), 3 (=6-3), and 5 (= 6-1).

A (lag) covariance is defined as

$$\underline{\hat{R}}_M = \langle \vec{z}_t \vec{z}_{t-M\delta t}^T \rangle = \underline{\hat{R}}_{-M}^T, \quad (36)$$

where z is a vector of normalized fluxes over an entire subgrid. The normalization is such that all the time-averaged points within a bin are replaced according to:

$$z_i = \Phi^{-1} \left(\frac{i}{N+1} \right), \quad (37)$$

where i represents the order the point would have in a sorted list such that z_1 replaces the smallest flux, and z_N replaces the largest. Thus, with a bin, the set of z 's has a Gaussian distribution with mean 0 and variance 1. The time offset is M time steps of size δt , and the angle brackets represent an average, effectively an average over time. For $M=0$, we have the spatial covariance matrix Σ :

$$\underline{\hat{R}}_0 = \langle \vec{z}_t \vec{z}_t^T \rangle = \underline{\Sigma} = \underline{Q} \underline{Q}^T. \quad (38)$$

The matrix Q defines the relationship between the principal component amplitudes q 's and the z 's:

$$\underline{\underline{\vec{z}}}_t = \underline{\underline{Q}}\underline{\underline{\vec{q}}}_t \quad (39)$$

Like the z 's, the q 's are Gaussians with zero mean and unit variance.

3.1 Computing Spatial (calccov covs) and Lag Correlations (calccov lagcovs)

We first build a database of covariances at various lags. We do this by repeatedly selecting two spatial bins at random from two randomly selected sensors, and a random energy channel for each sensor. We then find the intersection of the binavg data in the first bin with binavg data in the second bin at the appropriate lag. If there are enough intersecting points (100), we compute a covariance between the z 's for two datasets and store it in a database for that lag. We repeat this until we have 3×10^5 points for the spatial covariance and 10^4 points for each lag covariance (we will see below why we need far less lag covariance information because of the principal component transform).

For spatial covariance (lag zero), we also do this across subgrids, where we select one point from the primary subgrid ($K-\Phi$) and one from the subordinate subgrid ($K-h_{min}$).

3.2 Filling in the Spatial Covariance Matrix (makecov cov)

With on the order of 10^4 – 10^5 grid points per subgrid, the full covariance matrix Σ is huge. Further, as we will see below, it contains a lot of “noise” correlations that we will not retain anyway, and we do not have enough simultaneous data to actually constrain such a huge covariance matrix. Therefore, to keep compute times to a reasonable level, we decimate the spatial subgrids for computing the spatial covariance matrix. This has the effect of reducing the fine detail of the spatial correlation structure, but that is the part we believe is least credible given the limitations of our observations. We decimate the subgrid iteratively over the three dimensions, increasing the decimation factor from 1 (meaning keep all points in that dimension) to 2 (meaning skipping every other point), etc. The dimension that is decimated is whichever is longest given the decimation up to that point. We iterate this process until the implied covariance matrix has less than about 200 million entries. Typically, the energy dimension is not decimated, while each spatial dimension is decimated by a factor of 2. Special care is taken to retain the end points of decimated dimensions.

Next, we obtain all the covariances from the lag-zero database. We are going to use a nearest-neighbors averaging to populate the covariance matrix on the decimated grid. First, we rescale their coordinates of the points in the database. The energy dimension is scaled first with a logarithm, and then linearly scaled to span 0 to 1. The spatial dimensions are simply linearly scaled to span 0 to 1 (although this already includes the coordinate transforms inherent in the grid, such as taking the logarithm of Φ).

If we are working with the primary grid, then we are creating a symmetric, square matrix, and we can save calculations by only doing the diagonal and one triangle, and copying the results to the other triangle.

To compute a point in one of the covariance matrices, we average the 100 nearest neighbors of that point's grid coordinates with the neighbors selected from the database for that lag. The set of nearest

neighbors is determined by Pythagorean distance in the rescaled coordinates domain (a hypercube from 0 to 1 in 6 dimensions).

It is worthwhile to note how the subgrids relate to each other. If we have two subgrids, the covariance matrix $\underline{\Sigma}$ can be represented in block matrix notation as:

$$\underline{\Sigma} = \begin{bmatrix} \underline{\Sigma}_{11} & \underline{\Sigma}_{12} \\ \underline{\Sigma}_{12}^T & \underline{\Sigma}_{22} \end{bmatrix}, \quad (40)$$

where $\underline{\Sigma}_{11}$ is the spatial covariance matrix on subgrid 1, $\underline{\Sigma}_{22}$ is the spatial covariance matrix on subgrid 2, and $\underline{\Sigma}_{12}$ is the spatial covariance matrix between subgrids 1 and 2. We will see below that we only need the column for the primary grid, which in the case of AE9 and AP9 is subgrid 2 (K- Φ). Thus, we only need $\underline{\Sigma}_{12}$ and $\underline{\Sigma}_{22}$.

3.3 Computing Principal Components (makeQ)

The principal components that drive the Monte Carlo scenarios are related to the spatial covariance according to Q (suppressing the subgrid subscripts for the moment):

We determine Q via eigenvalue decomposition of $\underline{\Sigma}$:

$$\underline{\Sigma} = \underline{V}\underline{\Lambda}\underline{V}^T \quad (41)$$

$$\underline{V}\underline{V}^T = \underline{I}, \quad (42)$$

where the eigenvalues are the diagonal elements of the diagonal matrix $\underline{\Lambda}$. We only retain enough entries in $\underline{\Lambda}$ such that QQ^T represents most of the variance in $\underline{\Sigma}$. The decision on how many entries in $\underline{\Lambda}$ to retain is based on two criteria: we remove all entries in $\underline{\Lambda}$ that represent less than 1% of the variance in $\underline{\Sigma}$. Because of numerical limitations, we also ensure that we stop retaining entries in $\underline{\Lambda}$ after we have stored more variance than exists in $\underline{\Sigma}$ (this happens because sometimes $\underline{\Lambda}$ has negative entries, which are numerical noise). We compute a preliminary Q from the truncated $\underline{\Lambda}'$ as:

$$\underline{Q} = \underline{V}\sqrt{\underline{\Lambda}'}. \quad (43)$$

While Q has as many rows as there are grid points, it typically only has about 10 columns; thus, all the temporal variation is controlled by changing amplitudes about 10 principal components q of spatial variation. The columns of Q are the principal components of spatial variation on the decimated (primary sub)grid.

The expressions for Q above give the principal components on the decimated primary subgrid. To obtain Q for a decimated subordinate grid, we have to relate Q to $\underline{\Sigma}_{12}$ via $\underline{\Sigma}_{22}$ (when the primary subgrid is number 2). Specifically:

$$\underline{\underline{\Sigma}} = \begin{bmatrix} \underline{\underline{Q}}_{11} \\ \underline{\underline{Q}}_{22} \end{bmatrix} \begin{bmatrix} \underline{\underline{Q}}_{11} \\ \underline{\underline{Q}}_{22} \end{bmatrix}^T = \begin{bmatrix} \underline{\underline{Q}}_{11} \underline{\underline{Q}}_{11}^T & \underline{\underline{Q}}_{11} \underline{\underline{Q}}_{22}^T \\ \underline{\underline{Q}}_{22} \underline{\underline{Q}}_{11}^T & \underline{\underline{Q}}_{22} \underline{\underline{Q}}_{22}^T \end{bmatrix} = \begin{bmatrix} \underline{\underline{\Sigma}}_{11} & \underline{\underline{\Sigma}}_{12} \\ \underline{\underline{\Sigma}}_{12}^T & \underline{\underline{\Sigma}}_{22} \end{bmatrix}. \quad (44)$$

Thus:

$$\underline{\underline{Q}}_{11} \underline{\underline{Q}}_{22}^T = \underline{\underline{\Sigma}}_{12}. \quad (45)$$

which means

$$\underline{\underline{Q}}_{11} = \underline{\underline{\Sigma}}_{12} \underline{\underline{Q}}_{22}^{-T}. \quad (46)$$

The matrix $\underline{\underline{Q}}_{22}^{-T}$ is computed as the transpose of the pseudoinverse of $\underline{\underline{Q}}_{22}$ (again, via singular value decomposition). This approach allows a single set of principal components on the primary subgrid to drive temporal variation on both subgrids via $\underline{\underline{Q}}_{11}$ and $\underline{\underline{Q}}_{22}$.

Next, we interpolate the $\underline{\underline{Q}}$'s from the decimated subgrids to the full subgrids. We then rescale the rows of $\underline{\underline{Q}}$ for each subgrid such that each has a sum-of-squares that is 1—this is important, as it guarantees the validity of the statistical transforms from q to z retain the property that z has zero mean and unit variance, which, in turn, guarantees that the flux x , when produced from z according to Eq. (16) or Eq. (24) has the right distribution.

The remainder of the operations can be computed only on the primary subgrid since the fluxes on the subordinate subgrid can be computed from the same q used on the primary grid.

3.4 Filling in Lag Correlation Matrices (makecov lagcov)

As described in TOR-2012(1237)-3, we can compute the lag covariance matrices on a further decimated grid. We select the points in the new decimated grid at random from the original undecimated grid. The number of points is the larger of $5N_q^2$ and $50N_q$, where N_q is the number of principal components. We then proceed through this new decimated grid to compute all the needed lag covariance matrices using the same nearest neighbors strategy used for the spatial covariance matrix in Subsection 3.2.

3.5 Computing Time Evolution Matrices (buildmc)

The equations and algorithms for computing G and C in Eq. (35) from the R 's are given in TOR-2012(1237)-3 and will not be repeated here (they are complicated).

Once the $\underline{\underline{Q}}$'s, G 's, and C are computed, we can save the entire set of runtime tables to a single file, such as AE9V10_runtime_tables.mat, which is both a Matlab save set and an HDF5 file.

4. Diagnostic Figures (figs)

A large number of diagnostic figures are generated. Manually browsing this set is a required visual validation of the model. Further empirical validation is performed by comparison to other datasets and models [AE9/AP9 Validation Doc].

Briefly, the diagnostic figures are:

thetafigs	Theta maps, i.e., flux maps, for each sensor
filledthetafigs	Theta maps, filled, for each sensor
fluxmaps	Flux maps for combined model
binspectra	Energy spectra & fit in spatial bins (similar to Figure 3)
pcfigs	Principal Components figures
radial_profile	Equatorial radial profiles vs AE8/AP8
SAA_profile	SAA latitude profiles vs AE8/AP8
FieldLine_profile	Field line profiles vs AE8/AP8
LEO_map	Flux contour at multiple altitudes

References

- _____ [AE9/AP9 Tech Doc] AE9/AP9/SPM V1.1 Radiation Environment Model: Technical Documentation, Air Force Research Lab Technical Report, *in preparation*.
- _____ [AE9/AP9 Validation Doc] AE9/AP9/SPM V1.1 Radiation Environment Model: Validation and Model Comparison Report, Air Force Research Lab Technical Report, *in preparation*.
- Ginet., G.P., T.P. O'Brien, S.L. Huston, W.R. Johnston, T.B. Guild, R. Friedel, C.D. Lindstrom, C.J. Roth, P. Whelan, R.A. Quinn, D. Madden, S. Morley, and Y.J. Su, AE9, AP9 and SPM: New models for specifying the trapped energetic particle and space plasma environment, *Space Sci. Rev.*, 2013, doi:10.1007/s11214-013-9964-y.
- Guild, T., T.P. O'Brien, J. Mazur, and M. Looper, On-orbit inter-calibration of proton observations during solar particle events, Aerospace Report No. TOR-2007(3905)-22, Aerospace Corporation, 2009.
- O'Brien, T. P., 2012a, Adding multiple time lags to AE9/AP9 V1.0, TOR-2012(1237)-3, The Aerospace Corporation, El Segundo, CA, 2012, *in press*.
- O'Brien, T. P., 2012b, Data Cleaning Guidelines for AE-9/AP-9 Data Sets, TOR-2012(1237)-4, The Aerospace Corporation, El Segundo, CA, 2012, *in press*.
- O'Brien, T. P., 2013, AE9V1.0 Templates Derived from AE8, TOR-2014-00292, The Aerospace Corporation, El Segundo, CA, 2013, *in press*.

Appendix A—Templates Used in Each Model

- AE9 Templates for K - Φ grid
 - AE9_crrespoststorm\$X_kf\$Yphif\$Z_KPhi_V3.mat
 - \$X = 50, 95
 - \$Y = 25, 30, 35
 - \$Z = 15, 17, 19
 - AE9V10_ae8\$X_KPhi_PLn\$Y_extrap\$Z.mat
 - \$X = min, max
 - \$Y = 3, 5, 8
 - \$Z = 1, 2, 3, ..., 9
- AE9 Templates for K - h_{min}
 - AE9V10_ae8\$X_Khmin_PLn\$Y_extrap\$Z.mat
 - \$X = min, max
 - \$Y = 3, 5
 - \$Z = 1, 2, 3, ..., 9
- AP9 Templates for K - Φ grid
 - AP9V10_ICT_KPhi_\$Xpctl_v3_lo\$Y_hi\$Z.mat
 - \$X = 25, 50, 75, 95
 - \$Y = 07, 10, 15
 - \$Z = 25, 40
 - AP9V10_ICT_KPhi_\$Xpctl_v3_nom.mat
 - \$X = 25, 50, 75, 95
- AP9 Templates for K - h_{min} grid
 - AP9V10_SCT_KHmin_\$Xpctl_bca_PLn\$Y.mat
 - \$X = 25, 50, 75, 90
 - \$Y = 3, x
- SPME Templates for α_{eq} - L_m grid
 - hydra2011_e_\$X_pl\$Y_\$Zpctl.mat
 - \$X = alls, dawn, dusk, mdnt, noon
 - \$Y = 10, 20, 30, 40
 - \$Z = 50, 75, 95, mean
- SPMH Templates for α_{eq} - L_m grid
 - milillo2001_h_\$X.mat
 - \$X = alls, dawn, dusk, mdnt, noon
 - \$X_h_\$Y_\$Z.mat
 - \$X = roeder2005, niehof2011
 - \$Y = alls, dawn, dusk, mdnt, noon
 - \$Z = 50pctl, 75pctl, 95pctl, mean
- SPMHE Templates for α_{eq} - L_m grid
 - \$X_he_\$Y_mean.mat
 - \$X = roeder2005, niehof2011
 - \$Y = alls, dawn, dusk, mdnt, noon

- SPMO Templates for $\alpha_{eq}-L_m$ grid
 - niehof2011_o_\$X_mean.mat
 - \$X = alls, dawn, dusk, mdnt, noon

Appendix B—Smoothing parameters

For each grid or subgrid a 3-D smoothing mask is defined. The mask defines relative weights for a local average, and the extent of the local average is given by the size of the mask in each dimension (i.e., the window size). Except where noted, equal weight is given to all points within the window.

- AE9
 - $E = 3, K = 7, \Phi = 5$
 - $E = 3, K = 3, h_{min} = 10$
- AP9
 - $E = 3, K = 3, \Phi = 5$
 - $E = 3, K = 5, h_{min} = 5$
- SPME, SPMH, SPMHE, SPMO
 - $E = 3, \alpha_{eq} = 3, L_m = 3^*$
 - (*Note L_m weighting is only 30% for neighbors)