BAS report

Alex Radcliffe

September 20, 2021

1 Overview

In this project, we sought to use an ensemble Kalman filter to model the behaviour of the Earth's radiation belt. The ensemble Kalman filter is a Monte Carlo method for combining observations we can make about a system with a physical model of how the system evolves, assuming that all the errors in both our model and our observations are normally-distributed. In our model we want to predict the phase space density of geomagnetically-trapped radiation in the outer Van Allen Belt in different L-shells, so for our model we use a one-dimensional radial diffusion equation to model our physical system, and use the Kalman filter to include data from NASA's Van Allen Probes to try to build accurate predictions.

2 Van Allen Probe Data

For the observations in this model, we have downloaded CDF files from NASA's RBSP Gateway containing data from NASA's Van Allen Probe B about the phase space density of radiation in the Van Allen Belts using the TS04 model of the Earth's magnetic field. I have written a script that processes this data as follows:

1. It extracts a list of all the data points from all of the CDF files we have, and filters them to extract all of the points within 15% of the particular μ and K at which we want to model the phase space density.

2. It then separates these points into orbits of the probe. Within each orbit we take a set of equally spaced points $(L_i)_{0\leq i<n_L}$ within the L-range in which we want to model the density, and average all of the points wihin some tolerance L_{tol} of that L-value to get a density at all of the L_i's where we have any data. We also average the times of all of the observations contributing to each averaged points to get a time associated with each of them.

3. We then linearly interpolate between these points to get a full phase space density for any time in between them. We cannot interpolate to get data from before the first data point for a given L-value, nor after the last one, so we only consider the largest time-range over which we can. We discretize this time range into n_t equally spaced points $(t_i)_{0\leq i<n_t}$ and interpolate to each of these points and put this into a 2D-array.

4. We also create another discretization with the same time steps, but discretizing L into $(L'_i)_{0\leq i<n_{L'}}$ such that $\Delta L' = 2L_{\text{tol}}$, as this means that every data point can only contribute to one of our averaged points. We then once again average our points within L_{tol} of each point in our new discretizaton, and find the time associated with it as before. After that, we snap each point of this data to the nearest time t_i to create another 2D-array, each row of which represents the density wherever we have it, and we populate it with NaN's wherever we don't.

5. We then export these arrays to a JSON file so that we don't have to do this every time we want to run the model on the same data.

3 Diffusion Model

The key equation behind the model is a one-dimensional radial diffusion equation:

$$\frac{\partial f}{\partial t} = L^2 \frac{\partial}{\partial L}\left(\frac{1}{L^2} D_{LL} \frac{\partial f}{\partial L}\right) - \frac{f}{\tau}$$

where f represents the phase space density at a given value of L at a given time t; D_{LL} is a diffusion coefficient that is a function of the Kp-index and L; and $\tau(Kp)$ is a function describing the lifetime of particles in the

radiation belts. We set $D_{LL}(Kp, L) = 10^{0.506Kp - 9.325}L^{10}$ as in Brautigam and Albert 2000 and $\tau(Kp) = 3/Kp$ as in Shprits et al. 2005.
We then turn this equation into an implicit scheme by once again discretizing our t-range into $(t_i)_{0 \leq i < n_t}$, and our L-range into $(L_j)_{0 \leq j < n_L}$, which gives us:

$$\frac{f_j^{n+1} - f_j^n}{\Delta t} = \frac{L_j^2}{\Delta L}\left[\frac{D_{j+1/2,LL}^{n+1}}{L_{j+1/2}^2}\left(\frac{f_{j+1}^{n+1} - f_j^{n+1}}{\Delta L}\right) - \frac{D_{j-1/2,LL}^{n+1}}{L_{j-1/2}^2}\left(\frac{f_j^{n+1} - f_{j-1}^{n+1}}{\Delta L}\right)\right] - f_j^{n+1}/\tau^n$$

where $f_j^n = f(t_n, L_j)$; $L_{j+1/2} = (L_j + L_{j+1})/2$; $D_{j+1/2,LL}^n = D_L L(t_n, L_{j+1/2})$; $\tau^n = \tau(Kp(t_n))$.

At every time-step $1 \leq j < n_t - 1$ we get $n_L - 2$ of these equations. We can rearrange each equation to get:

$$f_j^{n+1} - f_j^n = \frac{\Delta t L_j^2}{\Delta L^2}\left[\frac{D_{j+1/2,LL}^{n+1}}{L_{j+1/2}^2}\left(f_{j+1}^{n+1} - f_j^{n+1}\right) - \frac{D_{j-1/2,LL}^{n+1}}{L_{j-1/2}^2}\left(f_j^{n+1} - f_{j-1}^{n+1}\right)\right] - \frac{\Delta t}{\tau}f_j^{n+1}$$

$$\iff f_j^n = f_j^{n+1} + \frac{\Delta t L_j^2}{\Delta L^2}\left[\frac{D_{j-1/2,LL}^{n+1}}{L_{j-1/2}^2}\left(f_j^{n+1} - f_{j-1}^{n+1}\right) - \frac{D_{j+1/2,LL}^{n+1}}{L_{j+1/2}^2}\left(f_{j+1}^{n+1} - f_j^{n+1}\right)\right] + \frac{\Delta t}{\tau}f_j^{n+1}$$

$$= -\frac{\Delta t L_j^2}{\Delta L^2}\frac{D_{j-1/2,LL}^{n+1}}{L_{j-1/2}^2}f_{j-1}^{n+1} + f_j^{n+1}\left(1 + \frac{\Delta t}{\tau} + \frac{\Delta t L_j^2}{\Delta L^2}\left(\frac{D_{j-1/2,LL}^{n+1}}{L_{j-1/2}^2} + \frac{D_{j+1/2,LL}^{n+1}}{L_{j+1/2}^2}\right)\right) - \frac{\Delta t L_j^2}{\Delta L^2}\frac{D_{j+1/2,LL}^{n+1}}{L_{j+1/2}^2}f_{j+1}^{n+1}$$

To make notation simpler, I'm going to define:

$$X_i = -\frac{\Delta t L_{i+1}^2}{\Delta L^2}\frac{D_{i+1/2,LL}^{n+1}}{L_{i+1/2}^2}$$

$$Y_i = 1 + \frac{\Delta t}{\tau} + \frac{\Delta t L_{i+1}^2}{\Delta L^2}\left(\frac{D_{i+1/2,LL}^{n+1}}{L_{i+1/2}^2} + \frac{D_{i+3/2,LL}^{n+1}}{L_{i+3/2}^2}\right),$$

$$Z_i = -\frac{\Delta t L_{i+1}^2}{\Delta L^2}\frac{D_{i+3/2,LL}^{n+1}}{L_{i+3/2}^2}$$

which makes the equation into

$$f_{j+1}^n = X_j f_j^{n+1} + Y_j f_{j+1}^{n+1} + Z_j f_{j+2}^{n+1}$$

We have one of these equations each for each $0 \leq j \leq n_L - 2$, and we can express them all as a matrix equation as follows:

$$\begin{pmatrix} Y_0 & Z_0 & 0 & 0 & \cdots & 0 & 0 \\ X_1 & Y_1 & Z_1 & 0 & \cdots & 0 & 0 \\ 0 & X_2 & Y_2 & Z_2 & \cdots & 0 & 0 \\ 0 & 0 & X_3 & Y_3 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & Y_{n-2} & Z_{n-2} \\ 0 & 0 & 0 & 0 & \cdots & X_{n-1} & Y_{n-1} \end{pmatrix}\begin{pmatrix} f_1^{t+1} \\ f_2^{t+1} \\ f_3^{t+1} \\ f_4^{t+1} \\ \vdots \\ f_{n-2}^{t+1} \\ f_{n-1}^{t+1} \end{pmatrix} = \begin{pmatrix} f_1^t - X_0 f_0^{t+1} \\ f_2^t \\ f_3^t \\ f_4^t \\ \vdots \\ f_{n-2}^t \\ f_{n-1}^t - Z_{n-1} f_n^{t+1} \end{pmatrix}$$

We also need data about the Kp-index to feed into the model, so we download this data from NASA's OMNI-Web, and then interpolate it to every time in the model.
We then solve this equation to update the system, and for a simple one-dimensional diffusion model, we just keep updating until we have calculated our Phase Space Density at every time that we are interested in.

4 Ensemble Kalman Filter

Before we consider the Ensemble Kalman filter, let's first consider the Kalman filter itself. In the Kalman filter we start with an initial condition, and we incrementally update this to get predictions at every time-step we are interested in, just like we did in the diffusion model, however during every time-step, after we have used the model to create a prediction, we compare any observations we have about the state of the system to our model's prediction and use that to update our predicted model state. In order to work out how much to

change our model based off of an observation, we need to have some idea of how reliable our model is, and of how reliable our observation is. We do this by keeping track of an error covariance matrix for our model, and another associated with our observations.
The equations for the Kalman filter are reasonably simple, however, one of them (the covariance update equation, which works out how our model error covariance matrix evolves) only works if the equation for updating the state of the system is a linear transformation. There are fortunately two ways to deal with non-linearity though — the extended Kalman filter and the ensemble Kalman filter.
The extended Kalman filter linearizes the covariance update equation by expanding it as a Taylor series, but the model that we use is the ensemble Kalman filter, which sidesteps the problem of the non-linearity by not explicitly calculating the model covariance, and instead treating the state of the system as an ensemble of possible states for the system, and letting the covariance of the state naturally evolve as we update each vector in the ensemble.
Here is a brief step-by-step description of the way that our implementation of the algorithm works:

1. First we initialize the system. We only try to run the model over the time interval that we were able to interpolate the Phase Space Density, so that we can use this to get both boundary conditions and an initial condition for our model. To make things even easier, we use the same discretization of L and t as we did when we were interpolating the density, so we have $(t_i)_{0\leq i<n_t}$, $(L_j)_{0\leq j<n_L}$.
   First we want to get an estimate for the initial state of the system. Throughout the model, (except for the part using the diffusion equation to update the system) we use the log of the phase space density as the state of the system throughout the model, because then the errors are closer to normal (which is the fundamental assumption of the Kalman filter). Hence, for the initial state vector, we take

$$u_{\text{initial}} = \begin{pmatrix} \log(f_0^0) \\ \log(f_1^0) \\ \log(f_2^0) \\ \vdots \\ \log(f_{n_L}^0) \end{pmatrix}$$

   where the values of f_i^0 are lifted from the interpolated Van Allen Probe data. To get an ensemble of n_{runs} initial states, we take an initial covariance matrix, which we estimate to be $P_a = 0.3I_{n_L}$ where I_{n_L} is an $n_L \times n_L$ identity matrix, and take a sample of n_{runs} error terms from $\mathcal{N}(\mathbf{0}, P_a)$ and add these to our initial state to get an ensemble of initial states.

2. The one other last bit of set up that we have to do before we start iterating through the times and updating the state is that we need to create a different perturbed set of Kp-indices for each run in the ensemble. Once again we start with the same Kp-indices as in the simple diffusion model, and we for each run we take a copy of this data, and perturb each value by adding a perturbation drawn from $\mathcal{N}(0, 0.5Kp)$.

3. Now we are finally ready to try updating our model. We have $n_t - 1$ update steps so that we can update our model to each time $(t_i)_{1\leq i<n_t}$. Each update step is composed of two parts:

   (a) First we have a prediction step, where we try updating each initial state in our ensemble through one time step using the same update step as in the diffusion model. We do this for each possible initial state in the ensemble using the corresponding Kp-index from our ensemble of perturbed Kp data. This then gives us an ensemble of predictions for the current time-step. We then also calculate the covariance of this ensemble of predictions, to be used later.

   (b) We then have an analysis step where we process any points of data we have from that time. We take the array of points (non-interpolated) from the Van Allen Probe data and find any points we have from this particular time. These points came from a different discretization of our L-range (L_i'), so in order to combine these observations with our model, we have to relate the two different discretizations. We do this by defining an operator $H_{L'}$ that maps the state of our model onto the observation space for each possible L' at which we have can have a data point. As we are considering individual points of Van Allen data, the observation space is one-dimensional which means that H will be a row vector.
   We do this by (for any state of the system $\mathbf{u}$) letting $H\mathbf{u}$ be the average of all of the model points at L-values within L_{tol} of the L' of the Van Allen probe data point (in other words all of the points from the model space that could have been aggregated into that point in our rolling average). Using this, for each run we calculate the innovation associated with this data point, which is given by:

$$d_{\text{run}} = \mathbf{v}_{\text{obs}} - H_{L'}\mathbf{u}_{\text{run}}$$

where $\mathbf{v}_{\text{obs}}$ is the log of the observed phase space density at that point, and $H_{L'}\mathbf{u}$ is the average of all the predicted log phase space densities within L_tol of L' where our observation was taken. This ends up as a scalar and it is essentially the difference between this model run's prediction and the Van Allen Probe's data point.
To work out how to update our model run based on this, we need to calculate what's called the Kalman gain, and for this we first need to calculate the covariance P of our ensemble.
Once we have this, we calculate the Kalman gain as

$$\mathbf{K} = PH^t/(HPH^t + R)$$

where R is the variance from the error in the Van Allen Probe data.
We can then update the state of the system (according to this run) by

$$\mathbf{u}_{\text{new}} = \mathbf{u}_{\text{old}} + \mathbf{K}d$$

After we've finished this for all runs, we take the average of all of these updated space, and this is our state estimate for this time.

Once we have then iterated through this at every time step, we should have a full PSD over our time range, and then our work is complete.

5 Example

As an example of what the model can do, we have tried running it over a period through August and September 2017, at $\mu = 700MeV/G$, $K = 0.11R_EG^{1/2}$. From top to bottom, Figure 1 (on the next page) shows a graph of the Kp-index over time; a scatter plot of the Phase Space Density from the Van Allen Probe data; a pseudocolour plot of this same data but interpolating between points; a pseudocolour plot of the phase space density as predicted by our diffusion model; a pseudocolour plot of the phase space density as predicted by our Ensemble Kalman Filter; and finally a plot of the average innovation vector at each L value in our ensemble Kalman Filter.
We can see from the innovation vector, that in general our predictions are slightly higher than the points we observe from the Van Allen Probe data, especially at L values between 4 and 5. I suspect that this is mainly because of the non-linearity of our diffusion coefficients, which means that when we perturb Kp symmetrically we end up increasing the mean of the diffusion coefficients, especially at high L. Another issue is that the Van Allen Probes do not always reach our maximum L-value of 5.3, and so as we are using this (interpolated) as our upper boundary condition this means that our data is somewhat unreliable at these values as we are interpolating from reasonably sparse data.
On the other hand, the ensemble Kalman filter is clearly doing a reasonable job, as the average difference between the (natural) log of the phase space density as seen in the points of Van Allen Probe data, and the predictions of the ensemble Kalman filter (before including the point) is only 0.631, whereas with the diffusion model the average difference is 1.85. This corresponds to the ensemble Kalman filter's predictions of the PSD being off by a factor of 1.88, where as the diffusion model's predictions are off by a factor of 6.36, so assimilating the data, clearly makes a big difference to the accuracy of our predictions.

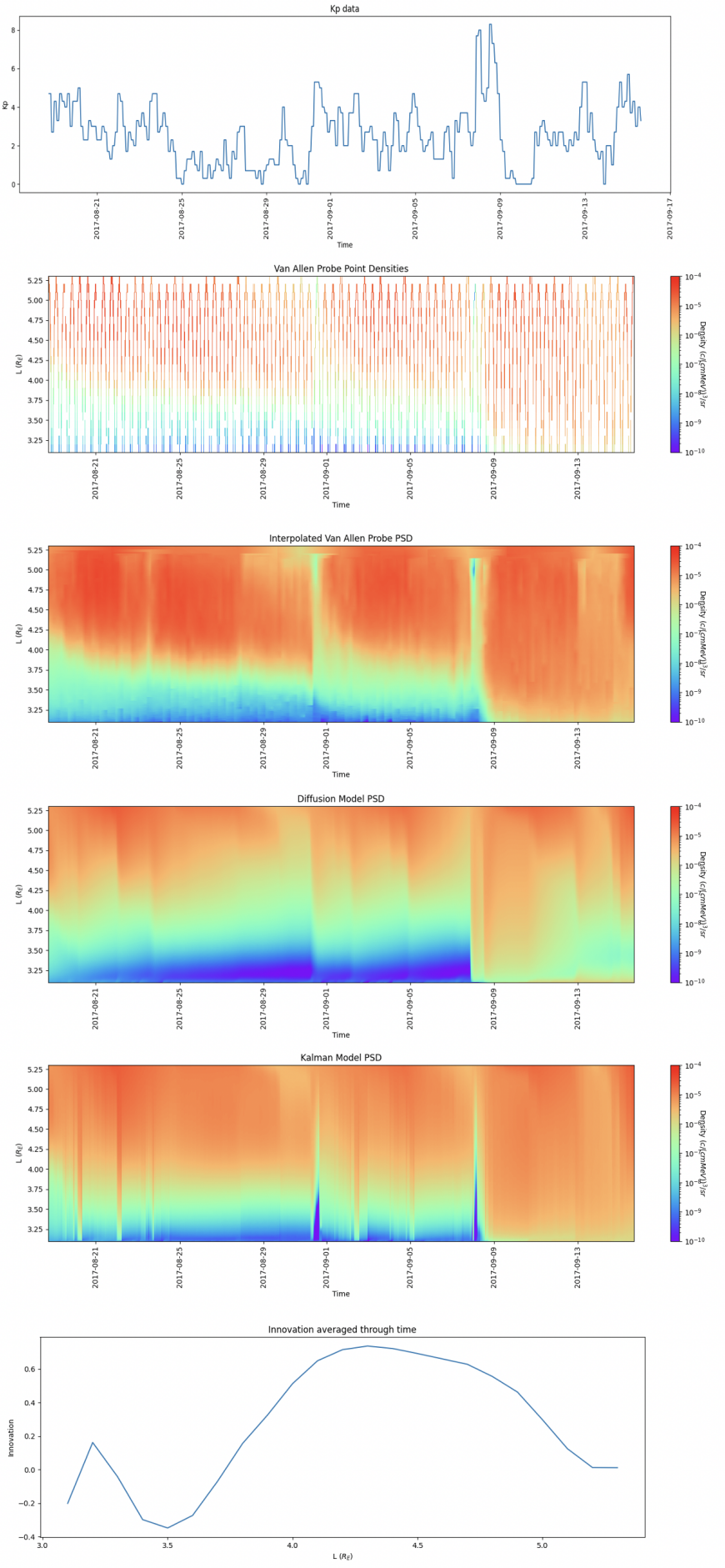

Figure 1: Figure 1