Modeling the Plasmasphere Dynamics with Data Assimilation

A. M. Jorgensen
Electrical Engineering Department
New Mexico Tech
USA

Outline

- Part 1 - Introduction
  - New Mexico Tech
  - My research
- Part 2 - Lecture
  - Introduction to data assimilation
  - Kalman filter
  - Ensemble Kalman Filter
- Part 3 - Science
  - Plasmasphere model
  - Observations
  - Data assimilation approach
  - Results
- Summary and Conclusions

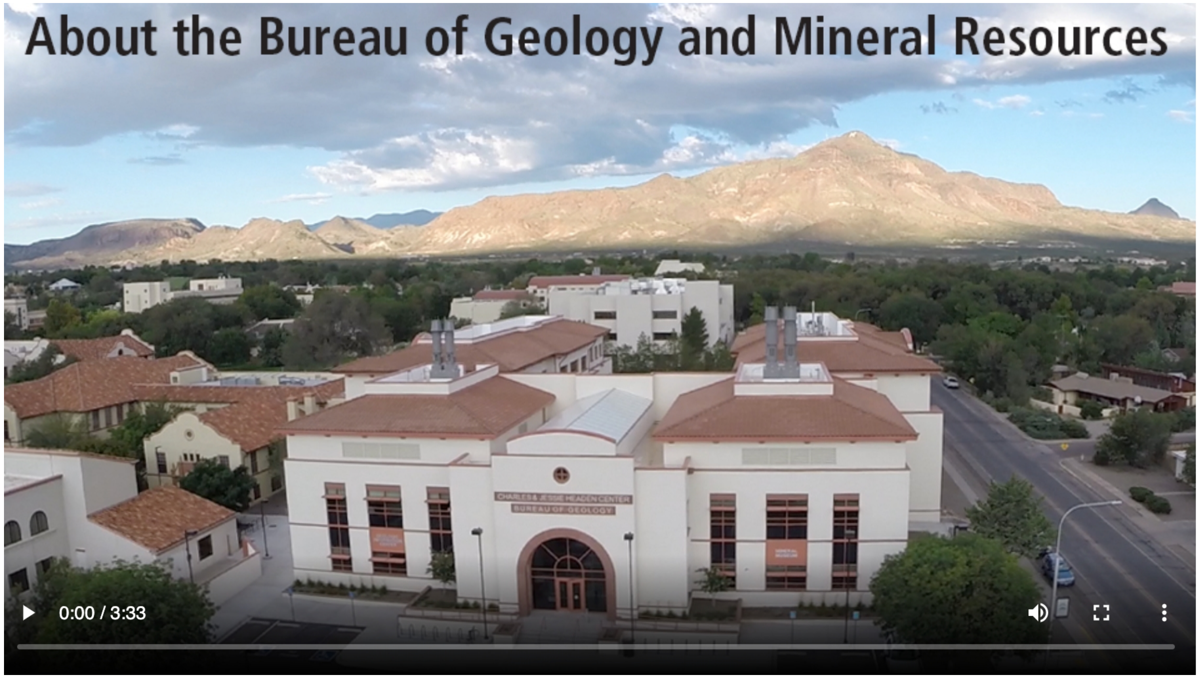
New Mexico Tech

- 2000 students
- 150 faculty
- Technical University
- EE department:

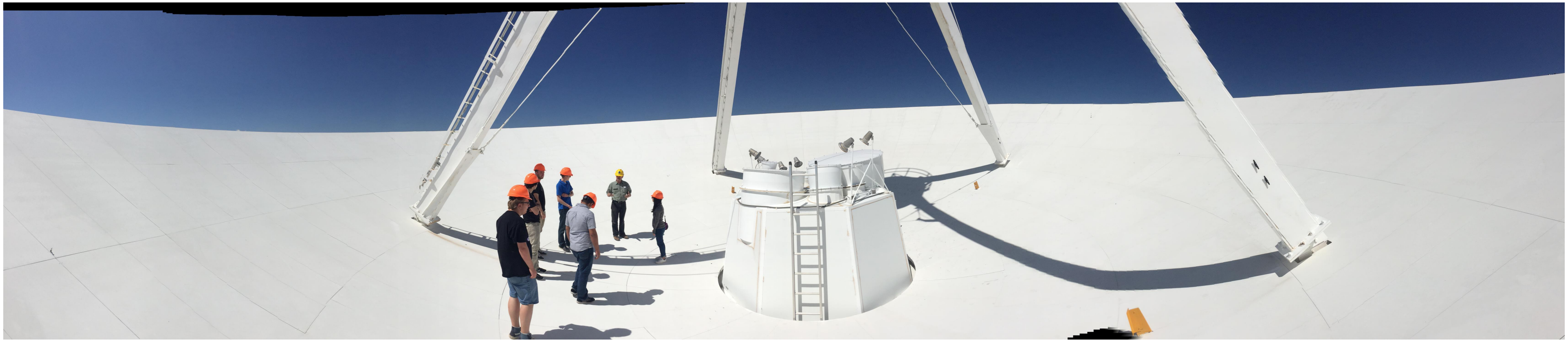
  150 students
- #1 public school in USA for: 10% of undergraduates earn PhD within 10 years of graduating.
- Princeton review: best value education
- USA Today: 8 on top 10 Engineering Schools in USA
- Etc....

New Mexico Tech – not just a small university

New Mexico Tech – not just a small university



New Mexico Tech – not just a small university



Me

B.S. Physics and Chemistry

PhD Astronomy

Postdoc
Scientist

Associate Professor

Ring current
Energetic Neutral Atoms

Wireless networking

Astronomical Interferometry
(Imaging stars)

Plasmasphere

Cubesats

Space Elevators

Instrumentation/controls

Cave meteorology

Software defined radios

Me (2)

- Associate Professor
- Teaching
  - Electrical Engineering courses
  - Electricity and Magnetism
  - Electrodynamics
  - RF and antennas
  - Analog Electronics
  - Instrumentation
  - Optics
  - Modeling and Simulation

Outline

- Part 1 - Introduction
  - New Mexico Tech
  - My research
- **Part 2 - Lecture**
  - **Introduction to data assimilation**
  - **Kalman filter**
  - **Ensemble Kalman Filter**
- Part 3 - Science
  - Plasmasphere model
  - Observations
  - Data assimilation approach
  - Results
- Summary and Conclusions

Data Versus Model

- “All models are wrong, some are useful.” --George Box, 1976
- Models are wrong.
- Data are sparse, sporadic, and noisy.
- What is one to do?

Mathematical Descriptions

- We begin with a model

$$\psi = \{\psi_0, \ldots, \psi_k\}$$

- ... which evolves

$$\psi_i = f\left(\psi_{i-1}\right)$$

- .. or if we want to call the drivers out explicitly

$$\psi_i = f\left(\psi_{i-1}, q_i\right)$$

- It's the same thing, but sometimes it is easier to think of the drivers as external and sometimes as internal to the model.

What is ψ_i?

- For weather modeling it is grids of temperature, pressure, wind velocities, humidity, etc etc.
- Perhaps also an array of the drivers (which are inserted by the function f()), although that is not strictly necessary.

Bayesian Likelihood

(from Evensen, *Ocean Dynamics*, 53, 343-367, 2003)

- The probability distribution of the model given the (uncertain) observations

$$f\left(\psi|d\right) = \frac{f\left(d|\psi\right) f\left(\psi\right)}{\int \left(\ldots\right)\, d\psi}$$

- But where did the prior probability distribution of ψ come from?
- It comes from the fact that we acknowledge that the model is not exact and that some "adjustment" which is not described by the numerical equations of the model is allowed.
- We will return to this later.

Sequential Likelihood Evaluation

- Remember that

$$\psi = \{\psi_0, \ldots, \psi_k\}$$

- So we can re-write the previous equation as

$$f\left(\psi_0, \ldots, \psi_k | d_0, \ldots, d_k\right) = \frac{f\left(\psi_0, \ldots, \psi_k\right) f\left(d_1, \ldots, d_k | \psi_0, \ldots, \psi_k\right)}{\displaystyle\int (\ldots)\, d\psi}$$

- ... or as

$$f\left(\psi_0, \ldots, \psi_k | d_0, \ldots, d_k\right) = \frac{f\left(\psi_0\right) f\left(\psi_1 | \psi_0\right) \ldots f(\psi_k | \psi_{k-1}) f(d_1 | \psi_1) \ldots f(d_k | \psi_k)}{\displaystyle\int (\ldots)\, d\psi}$$

- ... when assuming independent measurements

Sequential Likelihood Evaluation

- ... which can be decomposed sequentially like this

$$f\left(\psi_0, \psi_1 | d_1\right) = \frac{f(\psi_0) f(\psi_1|\psi_0) f(d_1|\psi_i)}{\int (\ldots)\, d\psi}$$

$$f\left(\psi_0, \psi_1, \psi_2 | d_1, d_2\right) = \frac{f(\psi_0) f(\psi_1|\psi_0) f(\psi_2|\psi_1) f(d_1|\psi_1) f(d_2|\psi_2)}{\int (\ldots)\, d\psi}$$

$$= \frac{f\left(\psi_0, \psi_1 | d_1\right) f(\psi_2|\psi_1) f(d_2|\psi_2)}{\int (\ldots)\, d\psi}$$

Sequential Likelihood Evaluation

- ... and in general

$$f\left(\psi_0, \ldots, \psi_k\right) = \frac{f(\psi_0, \ldots, \psi_{k-1}) f(\psi_k | \psi_{k-1}) f(d_k | \psi_k)}{\int (\ldots)\ d\psi}$$

- The conclusion is that Bayesian estimation can be decomposed sequentially.
  - We do not need to fit the entire time-sequence simultaneously.
  - Instead we can run the model forward one step at a time and incorporate observations multiplicatively.
  - The latter is much simpler.
  - Step-at-a-time assimilation is MUCH simpler than fitting (assimilating) to an entire time sequence at once.

The State Transition Probabilities

- We still need to understand the state transition probabilities

$$f(\psi_k|\psi_{k-1})$$

- ... because the model is supposed to be exact, so why are there probabilities involved?
- Because instead of the model evolution

$$d\psi = f(\psi)dt$$

  we should imagine a stochastic evolution - because the model is not exact -

$$d\psi = f(\psi)dt + g(\psi)w\,dt$$

  where w is a random variable

- Obviously the model probability will diverge in this case unless we apply constraints – from observations.

A Simple Linear Model

System evolution

Driver effect

External driver

$$x_{k+1} = F\,x_k + G u_k + w_k$$

Model noise

System state

Measurement

Measurement noise

$$z_k = H\,x_k + r_k$$

Measurement operator

Example: a car accelerating linearly

Example: A car accelerating linearly

State

Acceleration

$$\begin{bmatrix} x \\ v \end{bmatrix}_{k+1} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ v \end{bmatrix}_{k} + \begin{bmatrix} \frac{\Delta t^2}{2} \\ \Delta t \end{bmatrix} a_k$$

Evolution

Driver effect

Expressed as a noisy process

$$\begin{bmatrix} x \\ v \end{bmatrix}_{k+1} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ v \end{bmatrix}_k + \begin{bmatrix} \frac{\Delta t^2}{2} \\ \Delta t \end{bmatrix} (a_k + w_k)$$

Set or measured acceleration

Noise/inaccuracy

$$z_k = \begin{bmatrix} 1 & 0 \end{bmatrix} \times \begin{bmatrix} x \\ v \end{bmatrix}_k + r_k$$

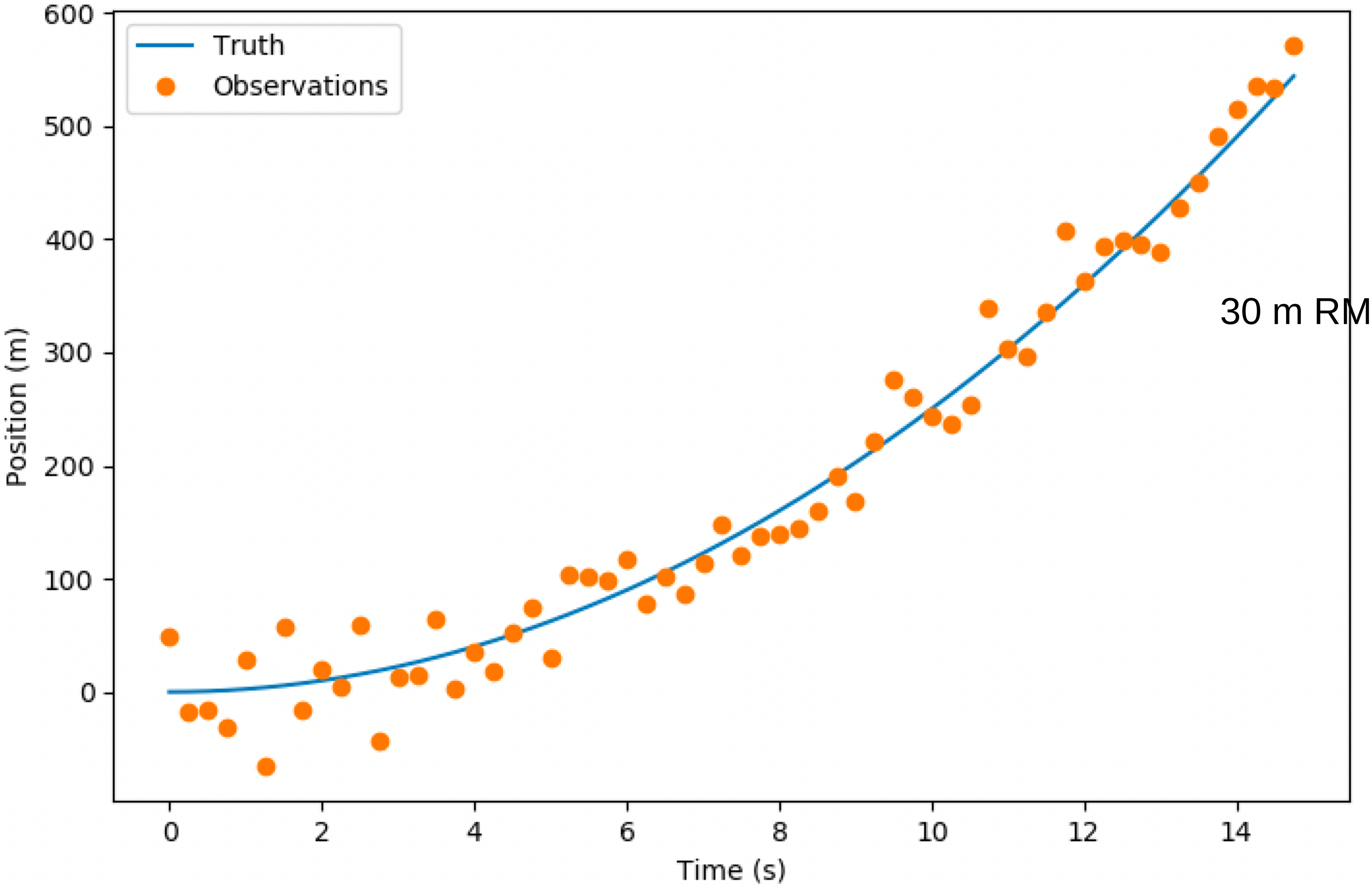
Position measurement with noise

Measurement noise (error)

We assume the model is correct, except for the real acceleration being different from the set or measured acceleration by w_k. We are able to measure the position from time to time with an uncertain r_k.

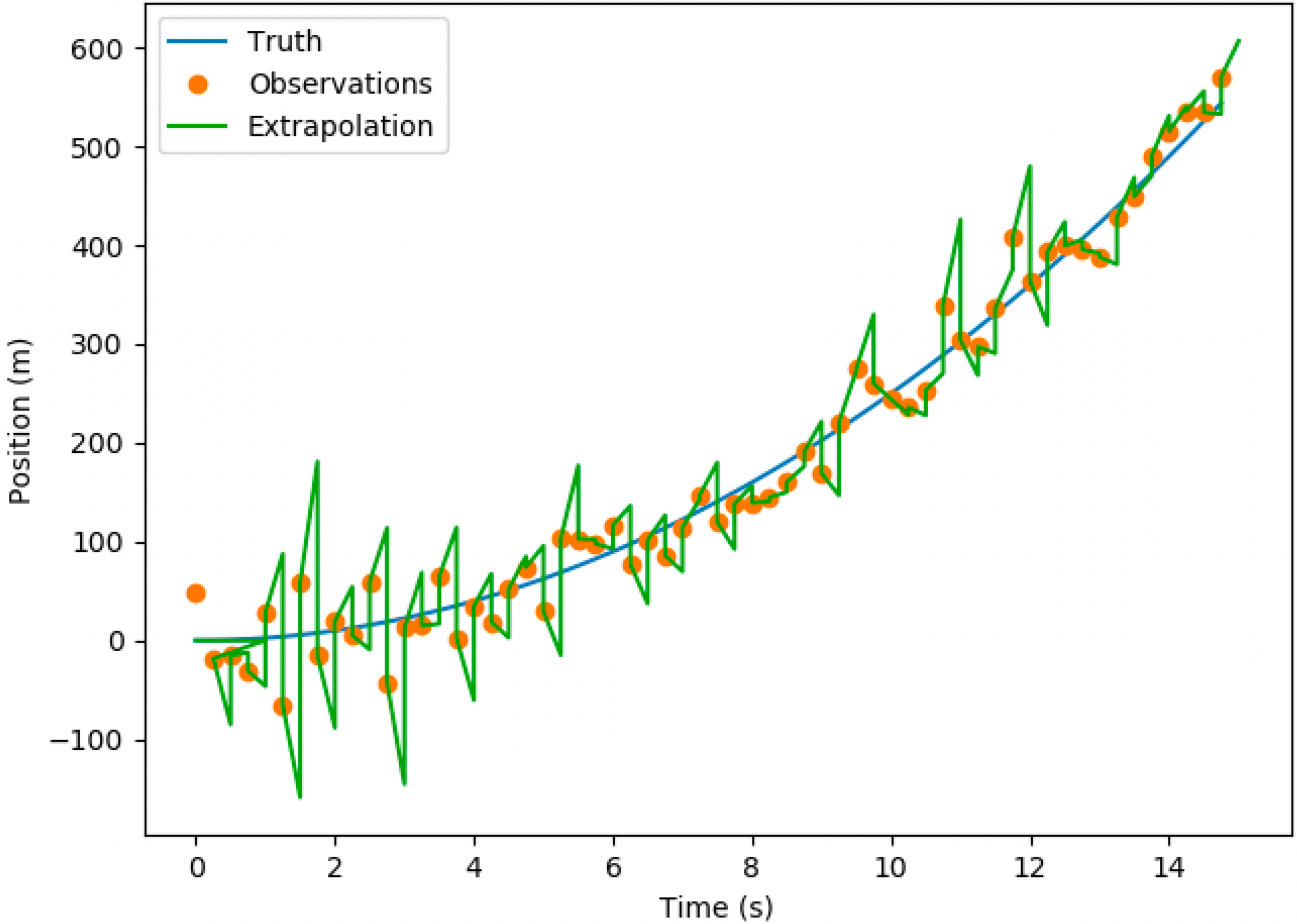
The challenge: estimate the position (and velocity) of the car as a function of time when the equations of motion are nearly correct and we have noisy measurements of position.

Noisy Observations

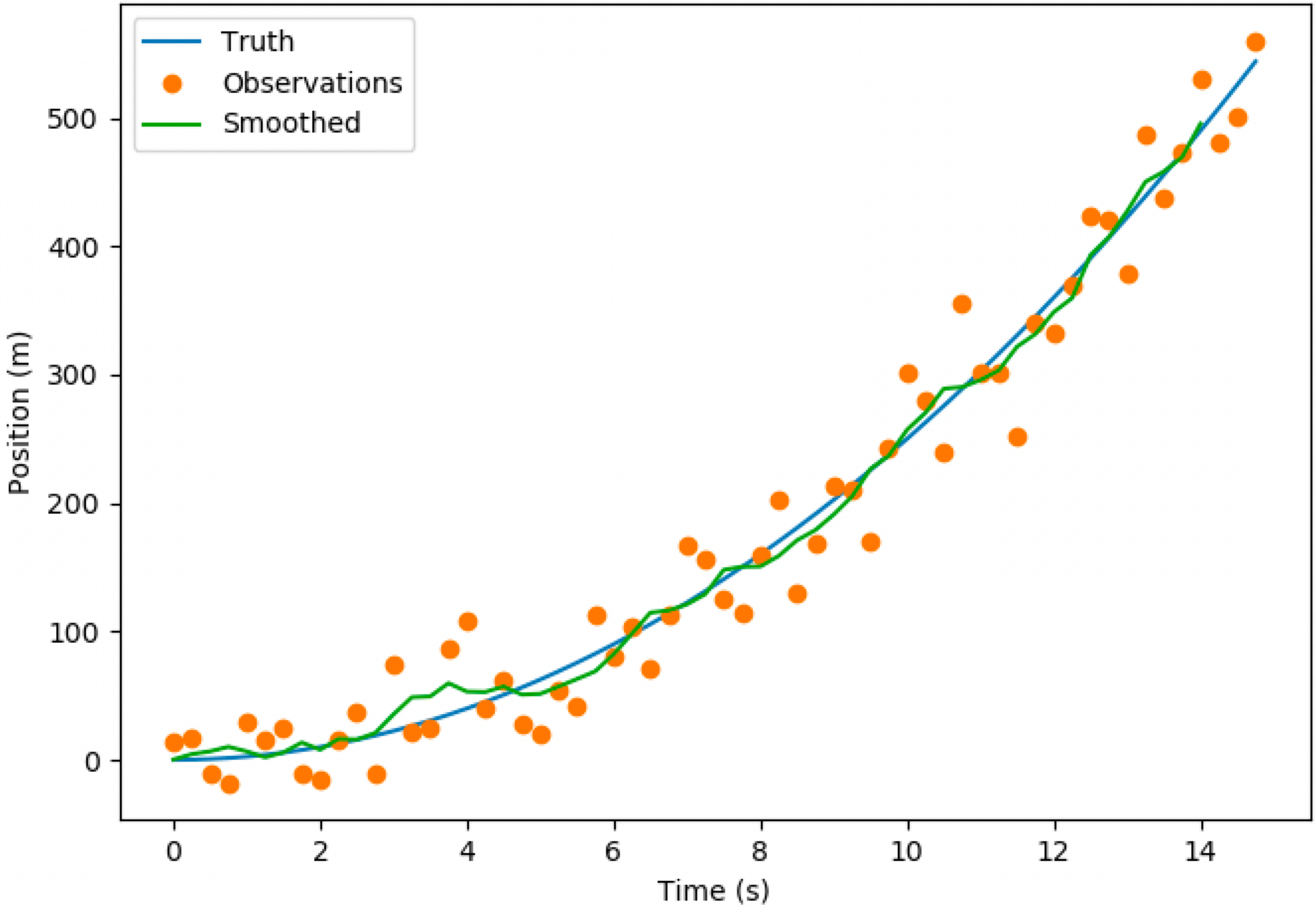


30 m RMS

First attempt: Extrapolate From Last Two Data Points



Second Attempt: Smooth over seven data points



Better Estimation

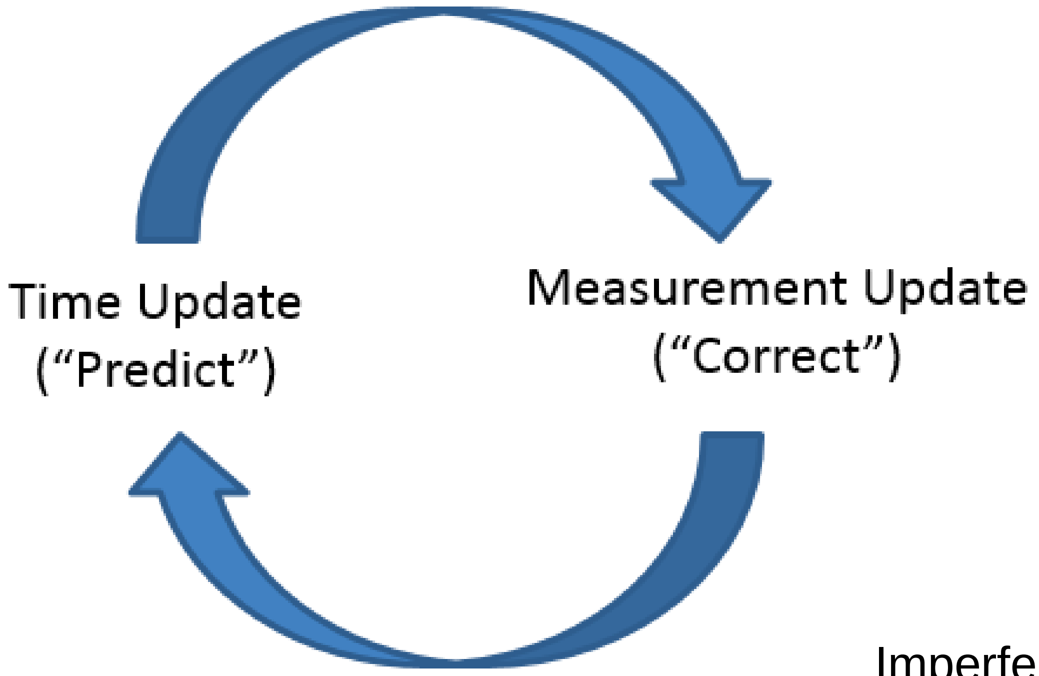
- So far we have estimated based on observations only
- We have not used prior known information about the system
  - There is a set of equations which relate position, speed, acceleration

$$\begin{bmatrix} x \\ v \end{bmatrix}_{k+1} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ v \end{bmatrix}_{k} + \begin{bmatrix} \frac{\Delta t^2}{2} \\ \Delta t \end{bmatrix} a_k$$

- How to incorporate the physical model into the estimation?
- Model-based filtering
- The Kalman Filter
  - Linear quadratic/Gaussian filter/estimator

The Kalman Filter Approach to Estimation

Advance the model according to the model equations.



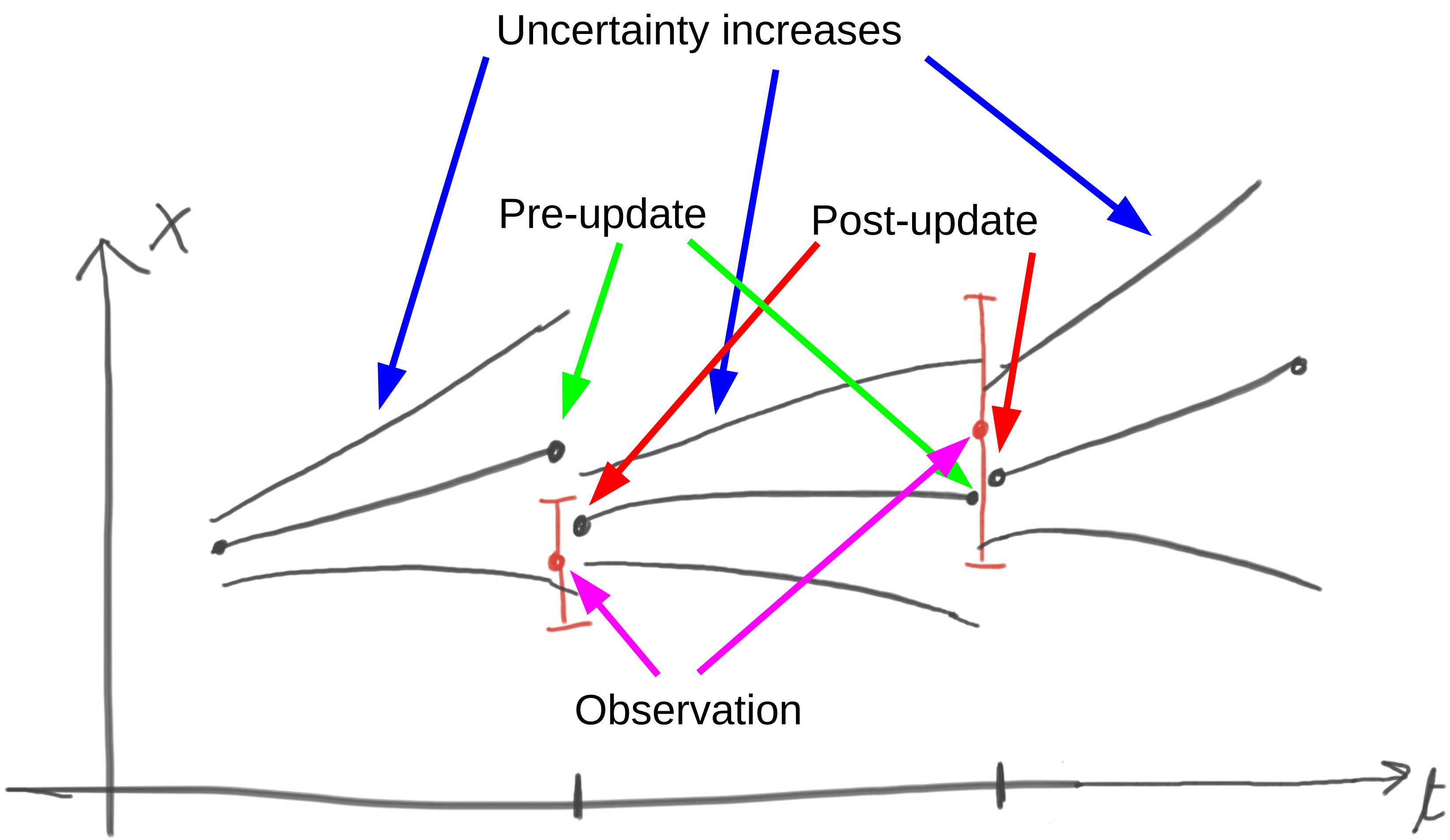
Correct the model by adjusting x and v, weighting the relative uncertainty of the model and the observations.

Imperfect observations of positions.

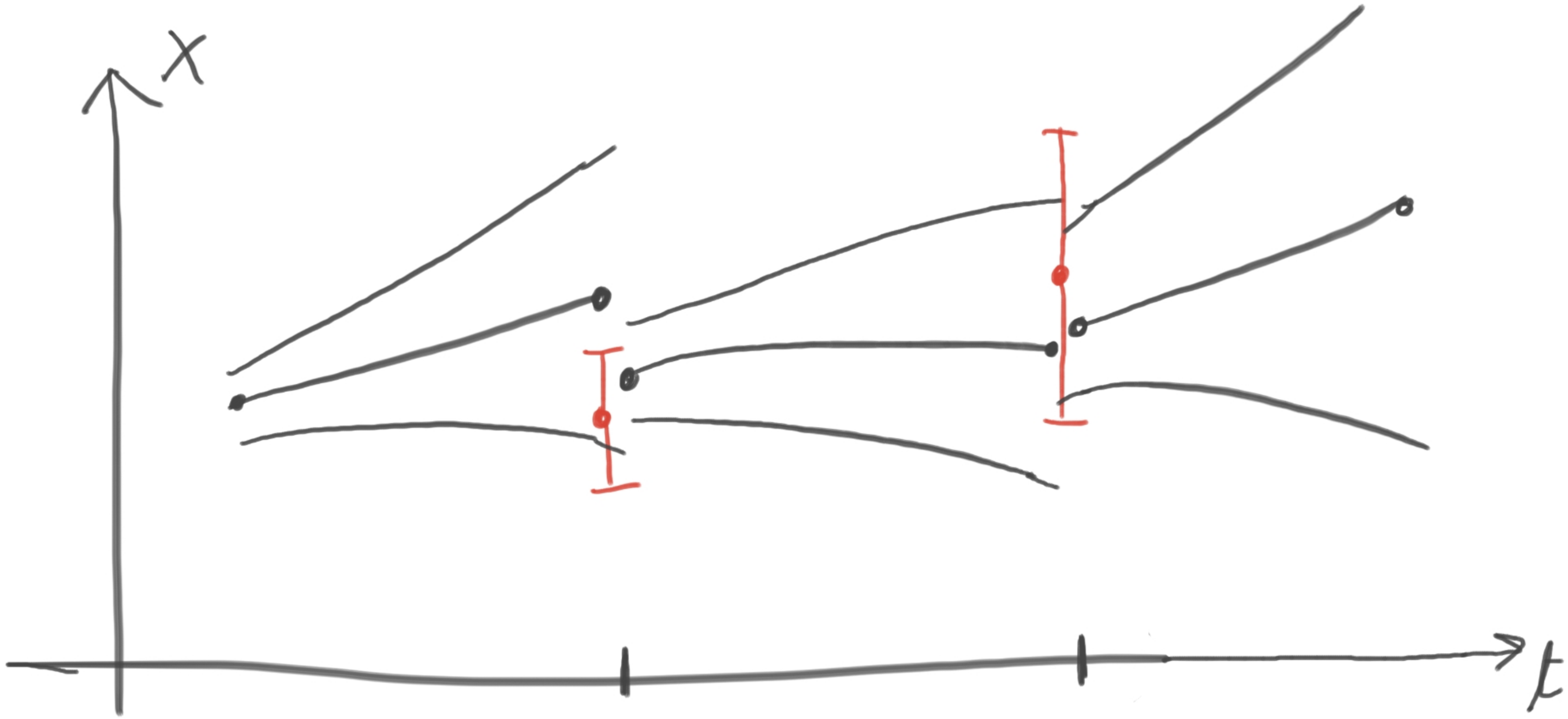
Imperfect knowledge of the acceleration

(Kalmanfilter.net)

Kalman Filter Output



Kalman Filter Output



Kalman Filter (with the math)

Predict

$$\hat{x}_k = \boldsymbol{F}_k \bar{x}_{k-1} + \boldsymbol{B}_k \bar{u}_k$$
$$\hat{\boldsymbol{P}}_k = \boldsymbol{F}_k \boldsymbol{P}_{k-1} \boldsymbol{F}_k^T + \boldsymbol{Q}_k$$

$$\bar{x}_0 \;\; \boldsymbol{P}_0$$

Advance

$$\bar{x}_k = \hat{x}_k$$
$$\boldsymbol{P}_k = \hat{\boldsymbol{P}}_k$$

$$\bar{z}_k$$

Update

$$\boldsymbol{K}_k = \hat{\boldsymbol{P}}_k \boldsymbol{H}_k^T \left(\boldsymbol{H}_k \hat{\boldsymbol{P}}_k \boldsymbol{H}_k^T + \boldsymbol{R}_k \right)^{-1}$$
$$\bar{x}_k = \hat{x}_k + \hat{\boldsymbol{K}}_k \left(\bar{z}_k - \boldsymbol{K}_k^T \hat{x}_k \right)$$
$$\boldsymbol{P}_k = \left(\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}_k \right) \hat{\boldsymbol{P}}_k$$

$\hat{\square}$ A predicted value

$\bar{\square}$ An updated value

The Accelerating Car in a Kalman Filter

$$\underbrace{\begin{bmatrix} x \\ v \end{bmatrix}_{k+1}}_{\bar{x}} = \underbrace{\begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}}_{\boldsymbol{F}} \times \begin{bmatrix} x \\ v \end{bmatrix}_k + \underbrace{\begin{bmatrix} \frac{\Delta t^2}{2} \\ \Delta t \end{bmatrix}}_{\boldsymbol{B}} (a_k + w_k)$$

$$z_k = \underbrace{\begin{bmatrix} 1 & 0 \end{bmatrix}}_{\boldsymbol{H}} \times \begin{bmatrix} x \\ v \end{bmatrix}_k + r_k$$

$$\bar{x}_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \qquad \boldsymbol{R} = \begin{bmatrix} 30^2 \end{bmatrix}$$

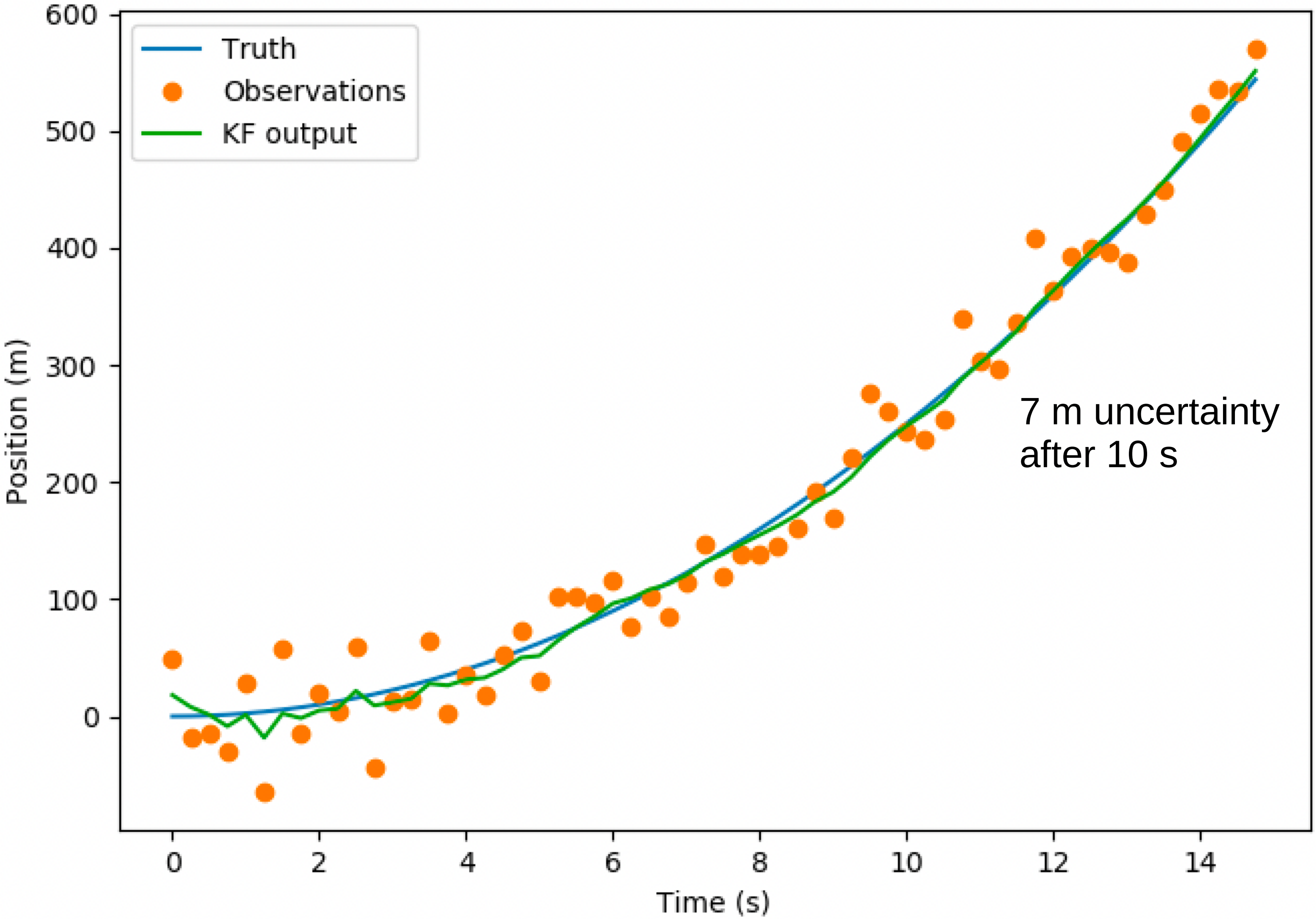
$$\boldsymbol{P}_0 = \begin{bmatrix} 500 & 0 \\ 0 & 500 \end{bmatrix} \qquad \boldsymbol{Q} = \begin{bmatrix} \frac{\Delta t^4 w^2}{4} & \frac{\Delta t^3 w^2}{3} \\ \frac{\Delta t^3 w^2}{3} & \Delta t^2 w^2 \end{bmatrix}$$

Python to the rescue

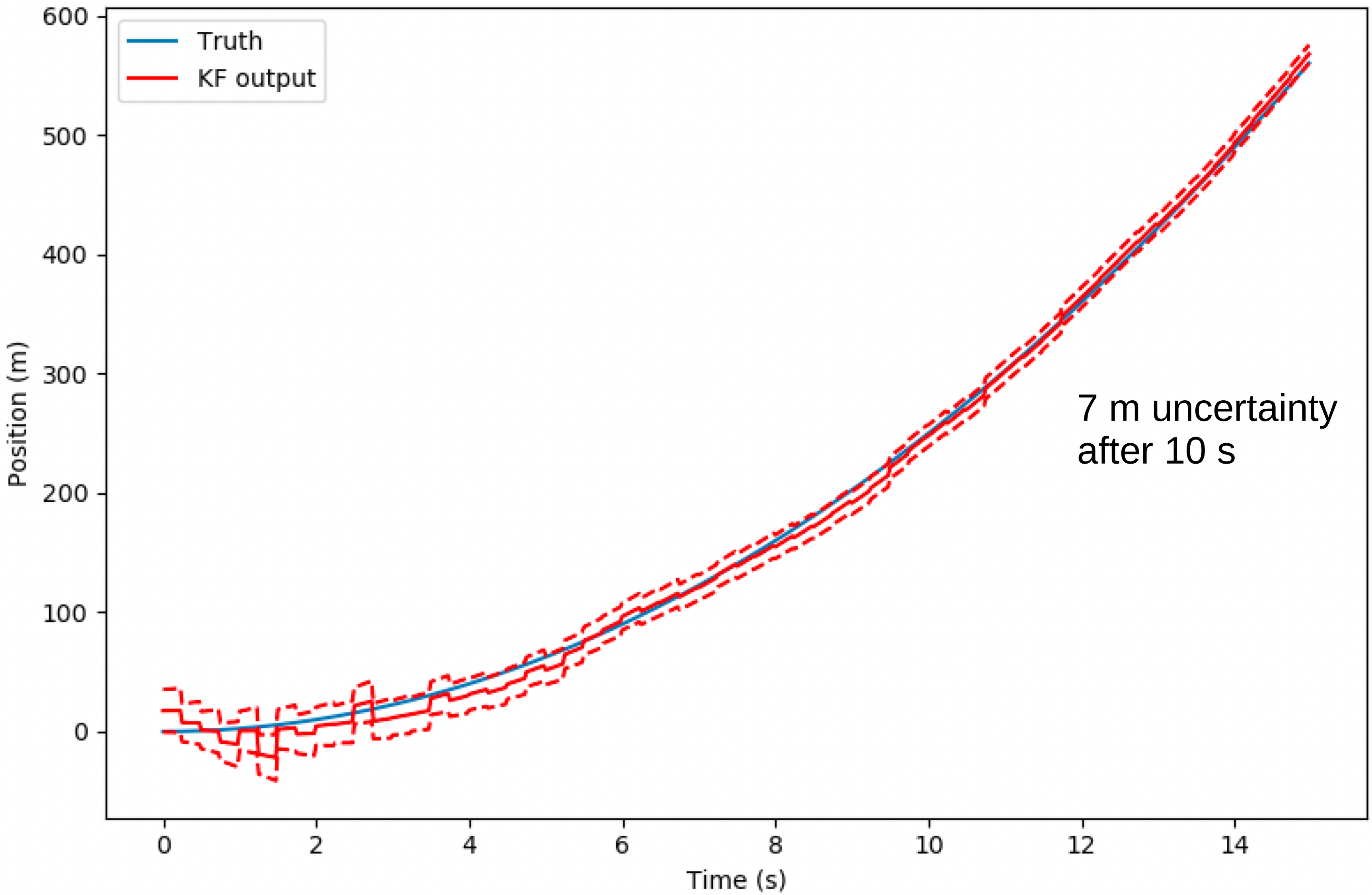
```
#!/usr/bin/python3

#
# Accelerating car and noisy observations
#

from filterpy.kalman import KalmanFilter
import numpy as np
import matplotlib.pyplot as plt

n=60
dt=0.25
a0=5
r=30.
w=.1
x0=0.
v0=0.

t=np.arange(n)*dt
x=0.5*a0*np.square(t)
np.random.seed(1)
z=np.random.normal(0,r,n)+x
a=np.random.normal(0,w,n)+a0
```

```
f=KalmanFilter(dim_x=2,dim_z=1)
f.x=np.array([[x0],[v0]])
f.F=np.array([[1,dt],[0,1]])
f.B=np.array([[dt*dt/2],[dt]])
f.H=np.array([[1.,0.]])
f.P=np.array([[500.,0],[0,500.]])
f.R=np.array([[r*r]])
f.Q=np.array([[dt*dt*dt*dt*w*w/4,dt*dt*dt*w*w/2],\
              [dt*dt*dt*w*w/2,dt*dt*w*w]])
xp=np.empty(n,dtype=float);

for i in np.arange(n):
    f.predict(a[i])
    f.update(z[i])
    xp[i]=f.x[0]

plt.plot(t,x,label='Truth')
plt.plot(t,z,'o',label='Observations')
plt.plot(t,xp,label='KF output')
plt.xlabel('Time (s)')
plt.ylabel('Position (m)')
plt.legend()

plt.show()
```

Kalman Filter Result

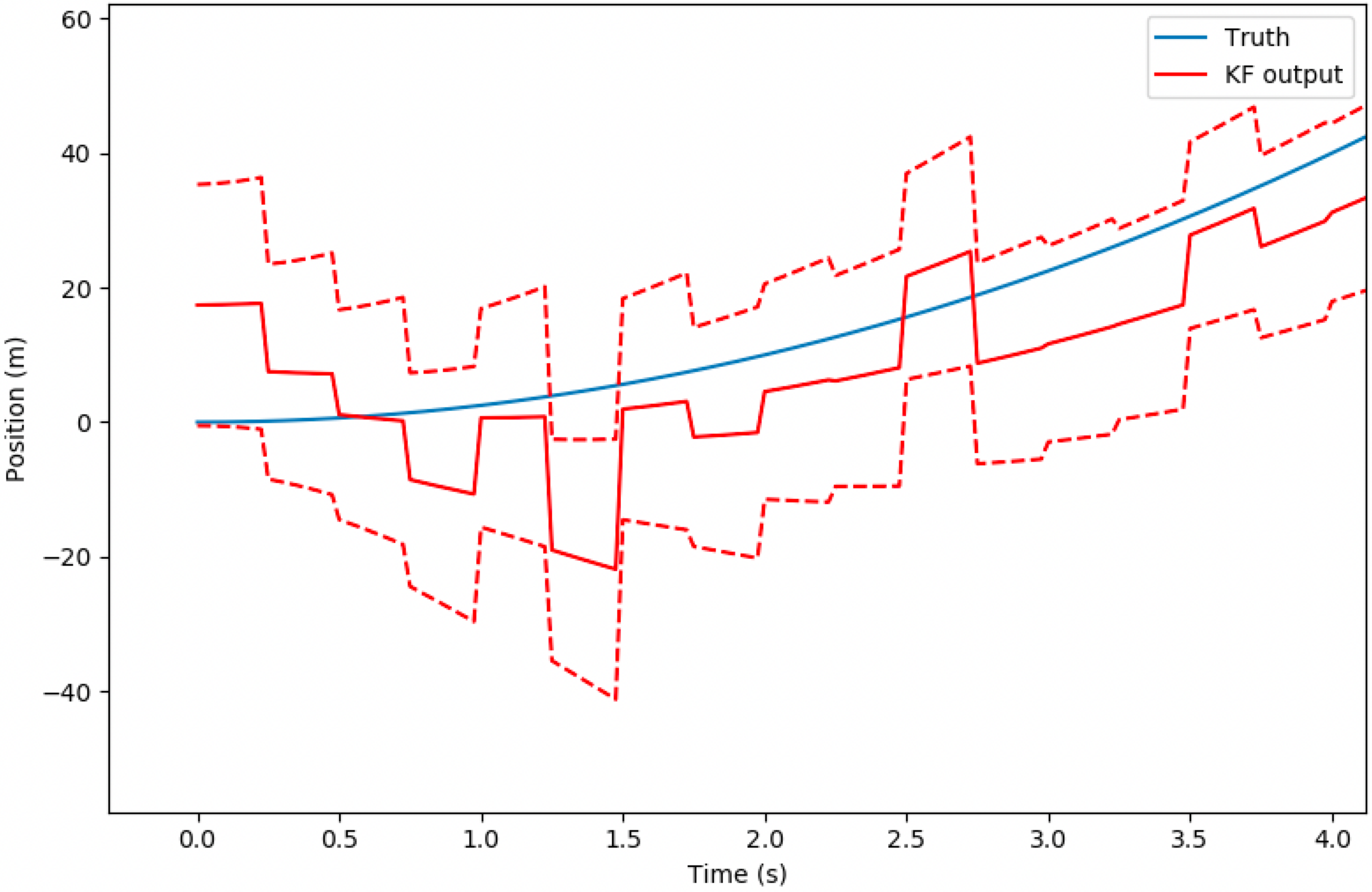


Smaller Steps with Intermittent Data



7 m uncertainty
after 10 s

Smaller Steps with Intermittent Data



What About Non-Linear Models?

What About Non-Linear Models?

- Anytime we encounter non-linear models:
- Attempt to linearize them
- The Extended Kalman Filter (EKF) is used for non-linear models
- Modify some of the equations of the linear Kalman Filter:

$$\hat{x}_k = f\left(\bar{x}_{k-1}, \bar{u}_k\right)$$

$$\boldsymbol{H}_k = \frac{\partial h}{\partial \bar{x}}_{\bar{x}_{k-1}, \bar{u}_k}$$

Jacobians

$$\boldsymbol{F}_k = \frac{\partial f}{\partial \bar{x}}_{\bar{x}_{k-1}, \bar{u}_k}$$

$$\bar{x}_k = \hat{x}_k + \hat{\boldsymbol{K}}_k \left[\bar{z}_k - h\left(\hat{x}_k\right)\right]$$

Ensemble Methods

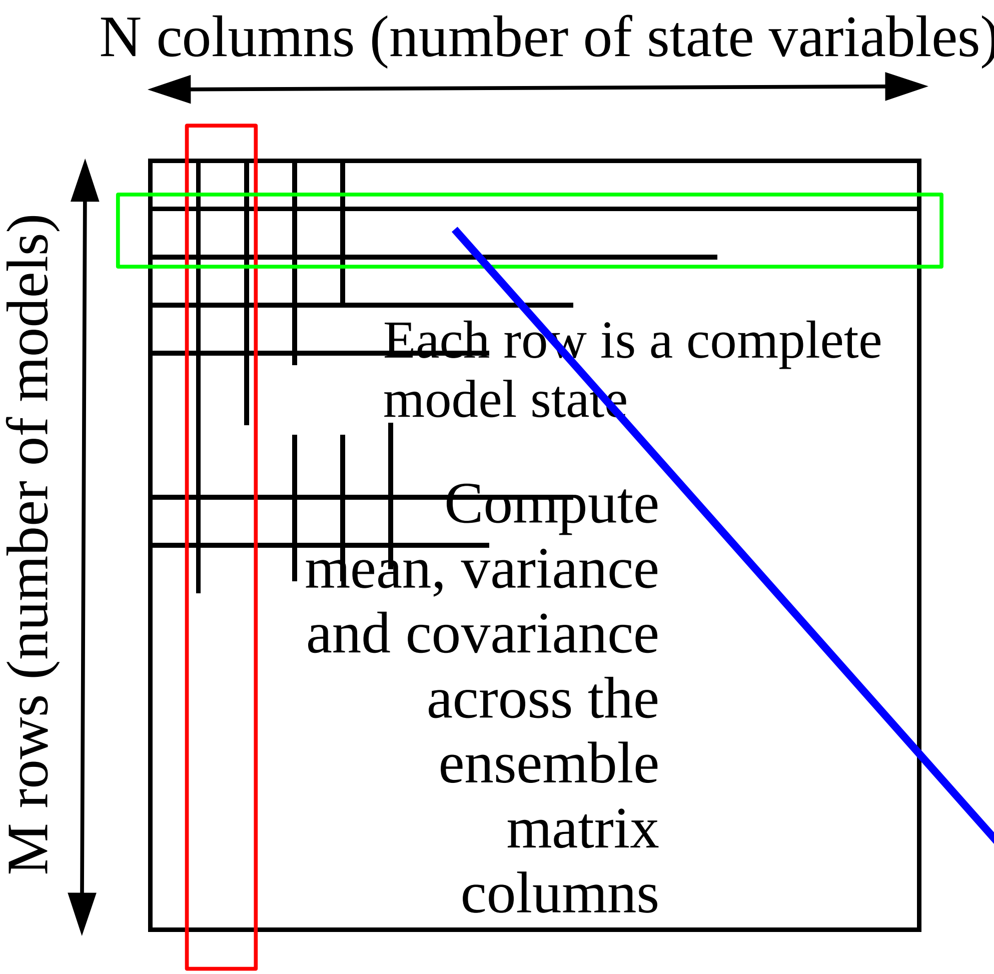
$$\begin{bmatrix} x \\ v \end{bmatrix}_{k+1} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ v \end{bmatrix}_{k} + \begin{bmatrix} \frac{\Delta t^2}{2} \\ \Delta t \end{bmatrix} (a_k + w_k)$$

$$\bar{\psi}_k = \begin{pmatrix} \psi_{k1} \\ \vdots \\ \psi_{kM} \end{pmatrix}$$

Instead of a covariance matrix to represent uncertainty, use an ensemble of models which has the same covariance

Ensemble methods are useful for large models:
E.g.: a weather model with 10^6 state variables:
A co-variance matrix will have $(10^6)2=10^{12}$ elements
In an ensemble with 102 models the ensemble matrix will have $10^6 x 10^2 = 10^8$ elements
Saving a factor 10^4 in memory and possibly computation

Ensemble Methods



Evolve each row as before

$$\psi_{kj} = f\left(\psi_{k-1,j}\right)$$

Each row evolves differently because of model noise. Here is “red noise”

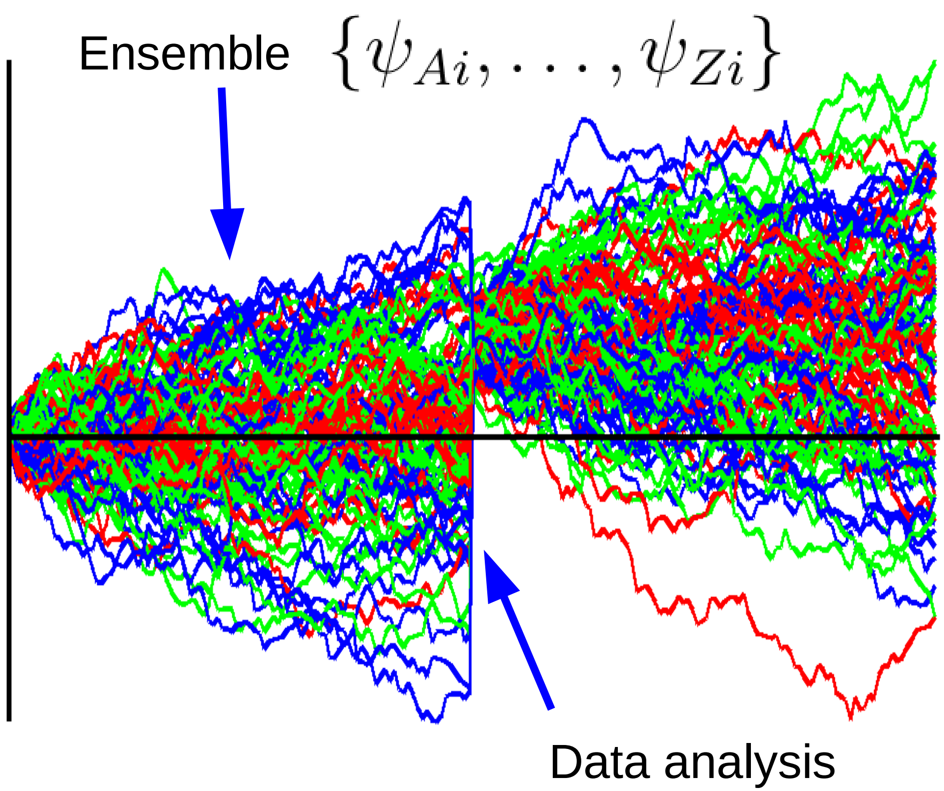
$$\bar{q}_{kj} = \alpha \bar{q}_{k-1,j} + \sqrt{1-\alpha^2}\,\bar{w}_{kj} \quad \boldsymbol{\alpha} = \frac{\mathbf{1}}{\boldsymbol{\tau}}$$

Noise can be applied to all elements of the state or just to some driver variables, forming an “enhanced” state

$$\psi_{kj} = \left(\bar{q}, \psi_{kj0}, \ldots, \psi_{kjN}\right)$$

Ensemble Methods

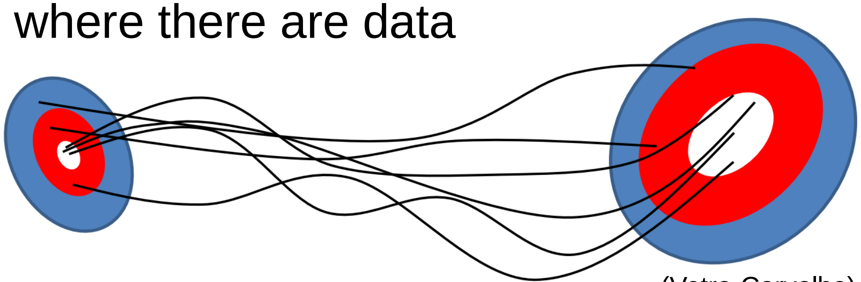
- Particle filters for very non-linear models
- Ensemble Kalman Filter (EnKF) for very large models which are “not too non-linear”..........



Particle filter analysis:
The new ensemble members are sampled members of the old ensemble.

Ensemble Kalman Filter analysis:
The new ensemble members are linear combinations of the old ensemble members.

Distribution:
In both cases done such that the variance represents the data variance where there are data



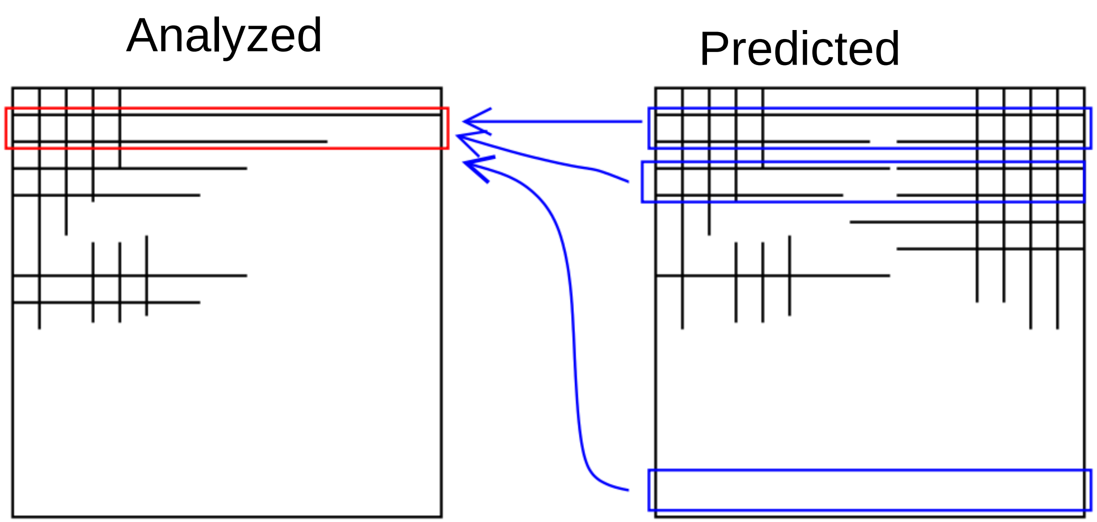
(Vetra-Carvalho)

The Ensemble Kalman Filter

- The Ensemble is conveniently represented as a matrix in which each column is a model Ensemble member.

$$\overline{\overline{\psi}}_a = \overline{\overline{X}}\,\overline{\overline{\psi}}_f$$

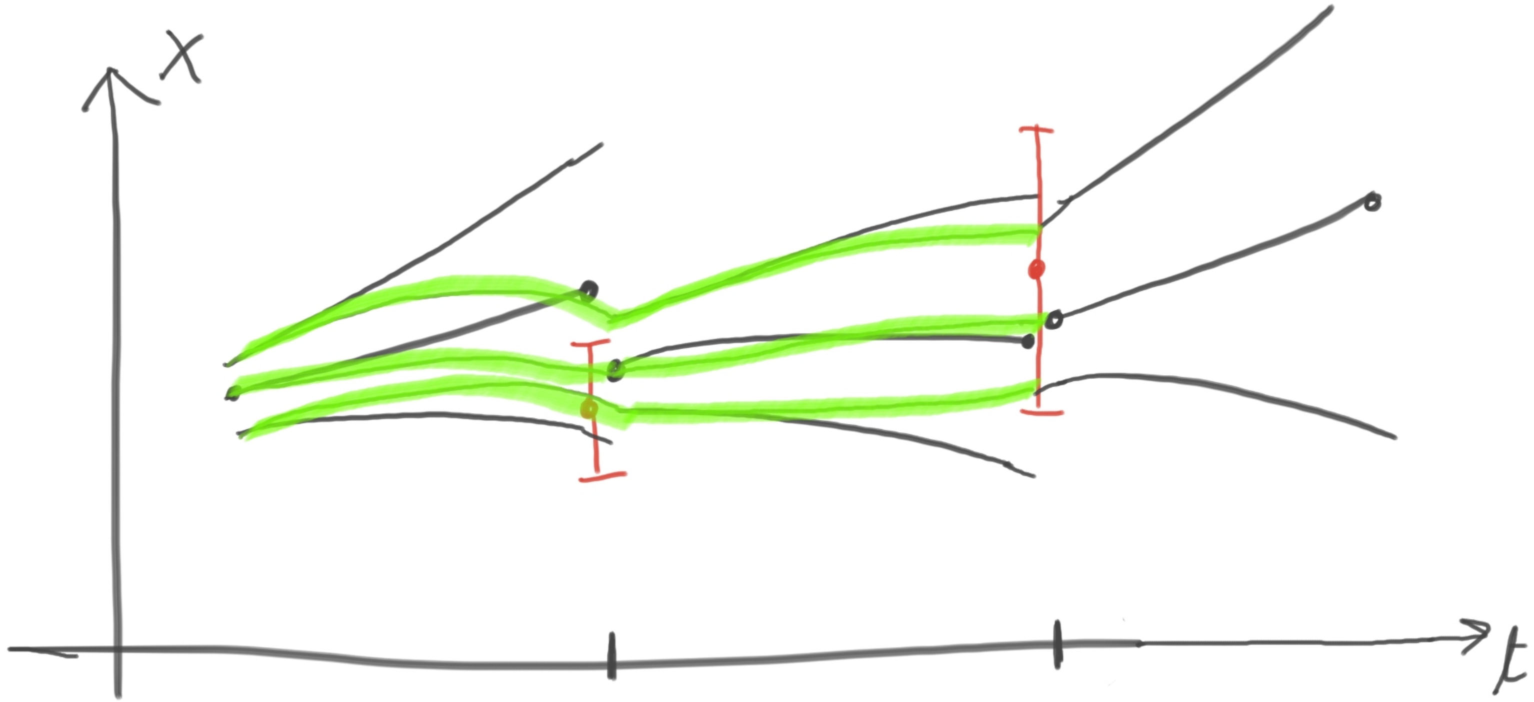
$$\overline{\psi}_{ai} = \sum_j x_{ji}\,\overline{\psi}_{fj}$$



- The complications of computing the matrix X are left out here, but make use of the differences between Ensemble members and observations.
- Multiplication by X, the analysis, make the new Ensemble have the variance of the observations at the observation points.

Hindcasting (Opposite of Forecasting)

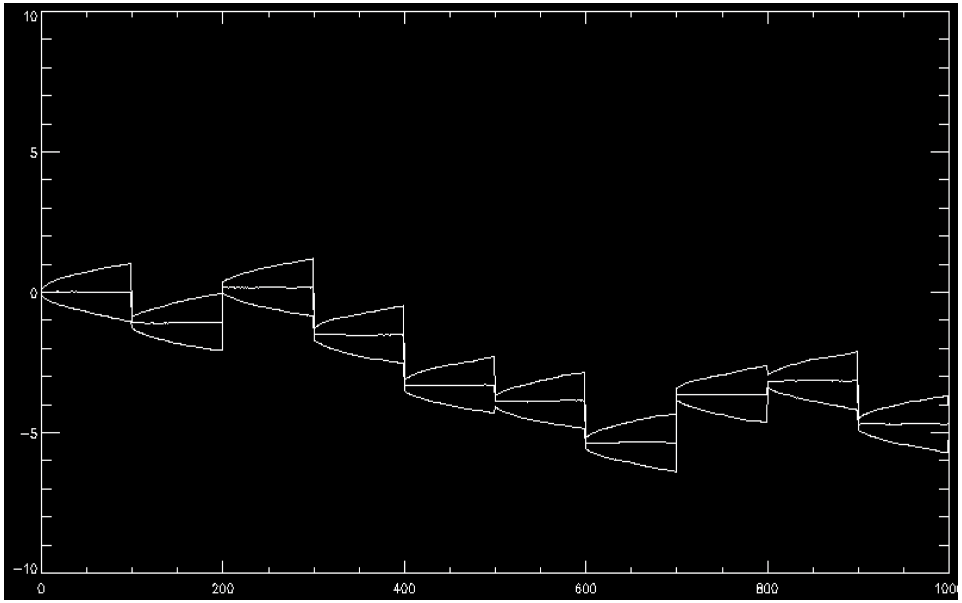
- Save past model states and perform the analysis on those past states.
- Produce more accurate time-histories of the model

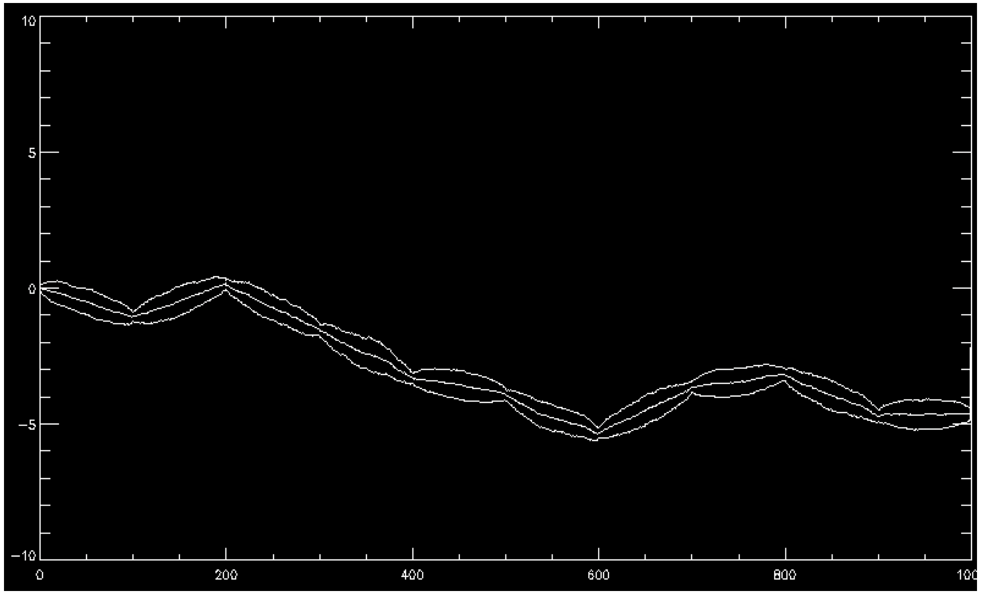


Tip: Hindcasting is normally called Kalman Smoothing

Hind-casting (Opposite of Forecasting)

- Useful in some circumstances, not useful in other circumstances
- Is hind-casting useful when...
  - Tracking an incoming nuclear missile which needs to be destroyed?
  - In preparing the daily weather forecast?
  - In re-running weather models for a research archive?

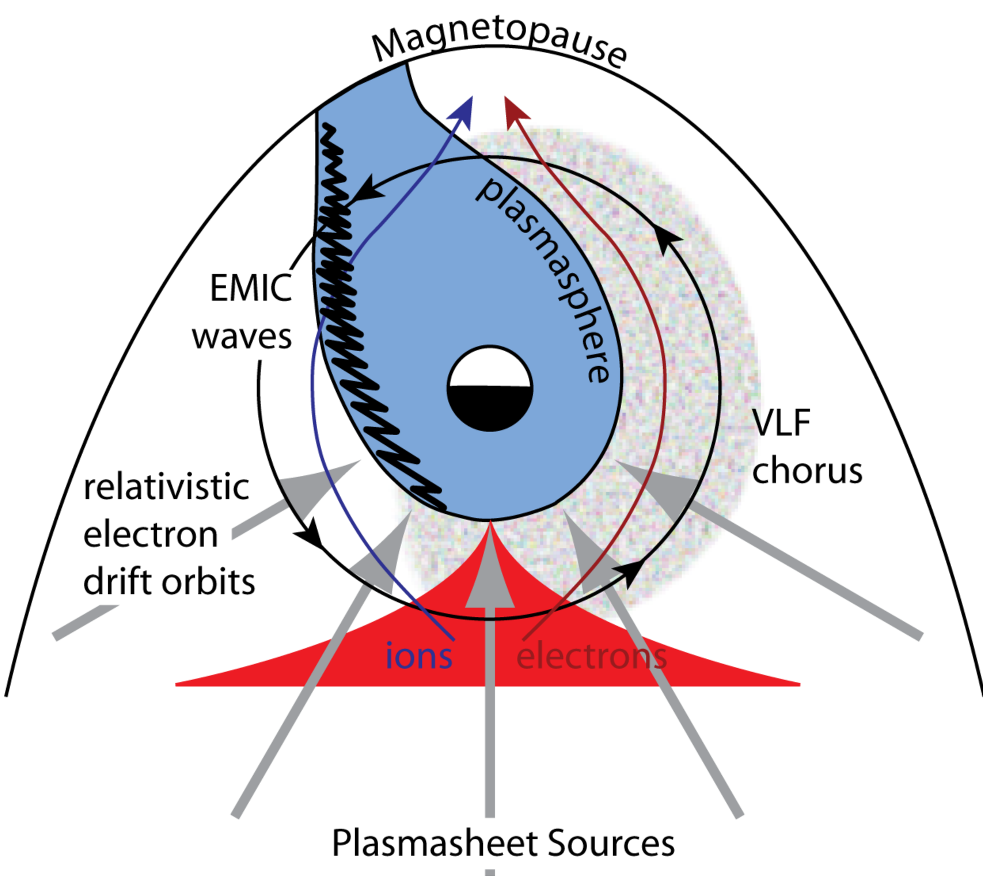




Outline

- Part 1 - Introduction
  - New Mexico Tech
  - My research
- Part 2 - Lecture
  - Introduction to data assimilation
  - Kalman filter
  - Ensemble Kalman Filter
- **Part 3 - Science**
  - **Plasmasphere model**
  - **Observations**
  - **Data assimilation approach**
  - **Results**
- Summary and Conclusions

The Plasmasphere



- Filling on the dayside – ionospheric source
- Loss on the nightside – ionospheric sink
- Co-rotation electric field due to Earth rotation
- Dawn-to-dusk Electric field

(This version from Reiner Friedel)

The Plasmasphere Model

- The Dynamic Global Core Plasma Model (DGCPM) (Ober et al. 1997)
- 2D Single species model which incorporates the three process of filling, loss, and transport

Magnetic field

$$\vec{B}\left(\vec{r}\right)$$

Electric field

$$\vec{E}\left(\vec{r}\right)$$

Flux tube content

$$N\left(\vec{r}\right)$$

Sunlit ionosphere

$$F_d = \frac{n_{\text{sat}} - n}{n_{\text{sat}}} F_{\text{max}}$$

Shadowed ionosphere

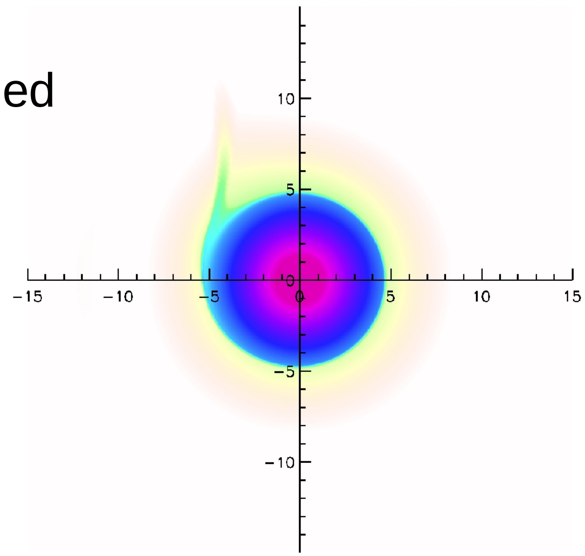
$$F_n = -\frac{NB_i}{\tau}$$

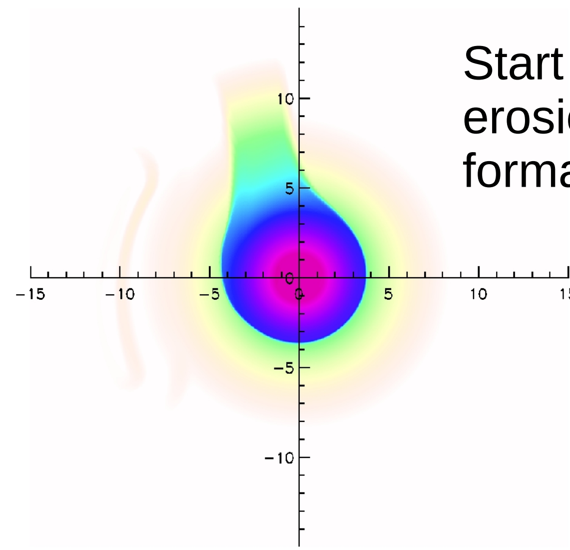
Convection + two hemispheres

$$\frac{D_\perp N}{Dt} = \frac{F_N + F_S}{B_i}$$

Model Run Example

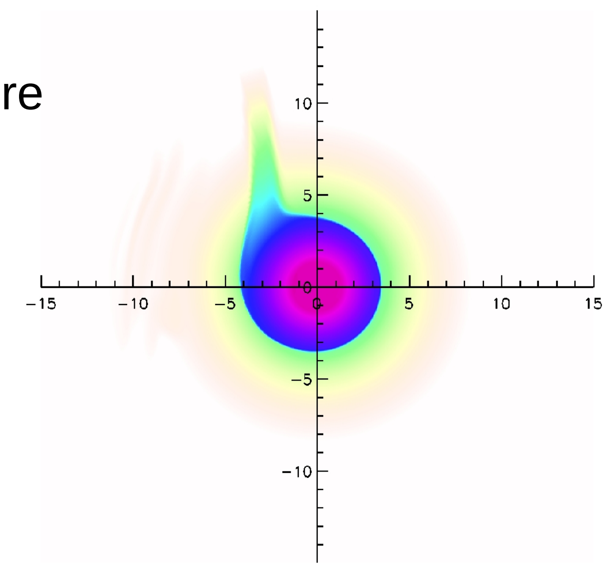
Quiet time extended plasmasphere to L=5

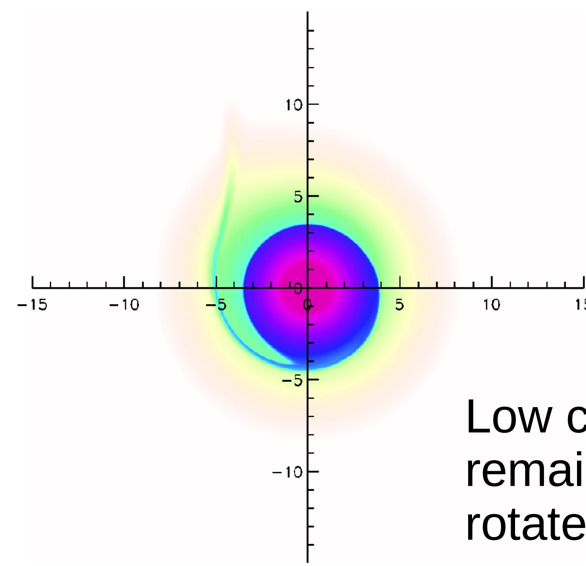




Start of high convection erosion and plume formation

Outer plasmasphere drained

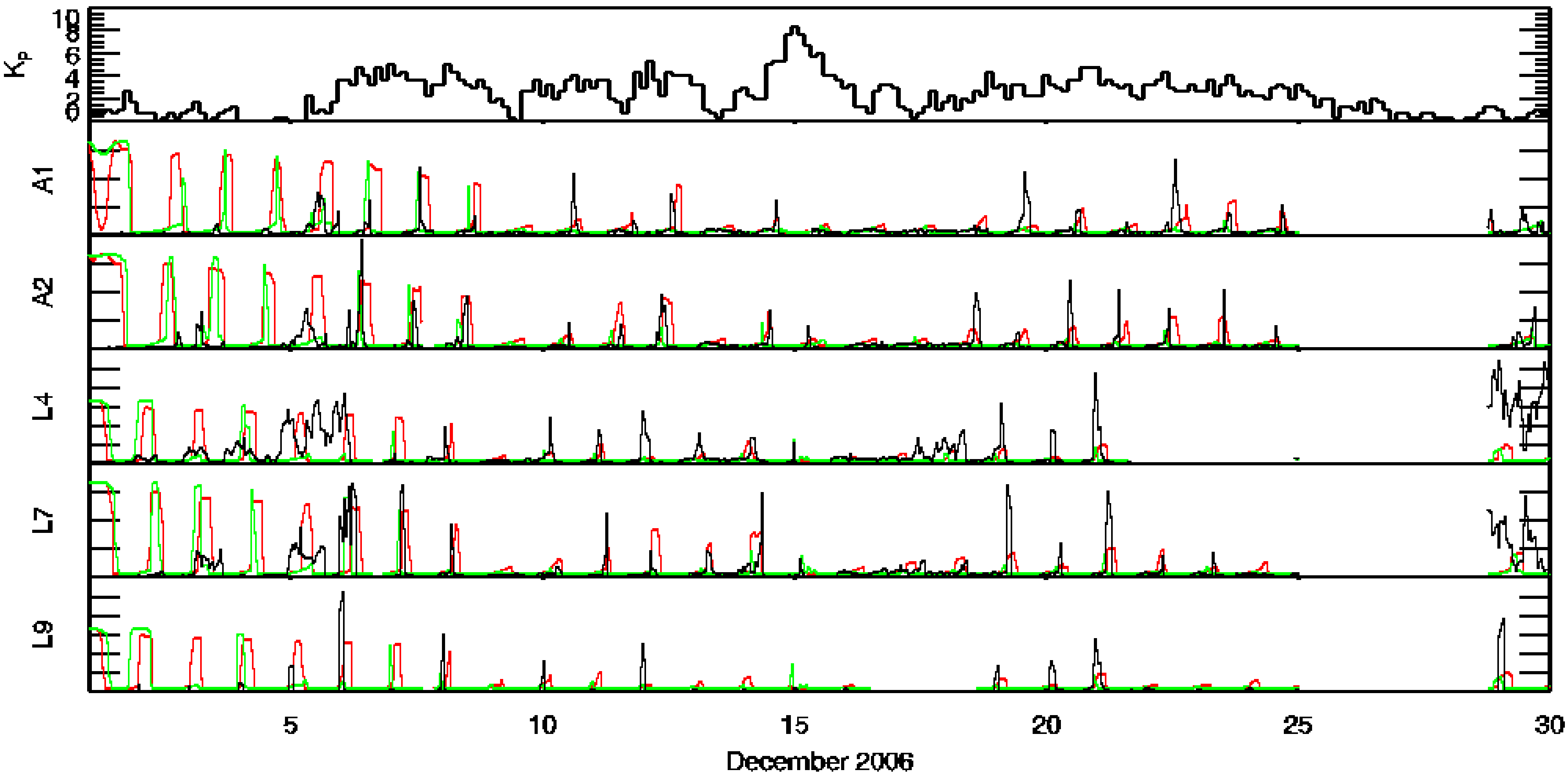




Low convection resumes, remainder of plume co-rotates

Model Data Comparisons

Gallagher et al. (1995) Sojka et al. (1986)



The Plasmasphere Model in an Ensemble Kalman Filter

Adjustable parts in the Ensemble Kalman Filter

Magnetic field

$$\vec{B}\left(\vec{r}\right)$$

Electric field

$$\vec{E}\left(\vec{r}\right)$$

Flux tube content

$$N\left(\vec{r}\right)$$

Sunlit ionosphere

$$F_d = \frac{n_{\mathrm{sat}} - n}{n_{\mathrm{sat}}} F_{\mathrm{max}}$$

Shadowed ionosphere

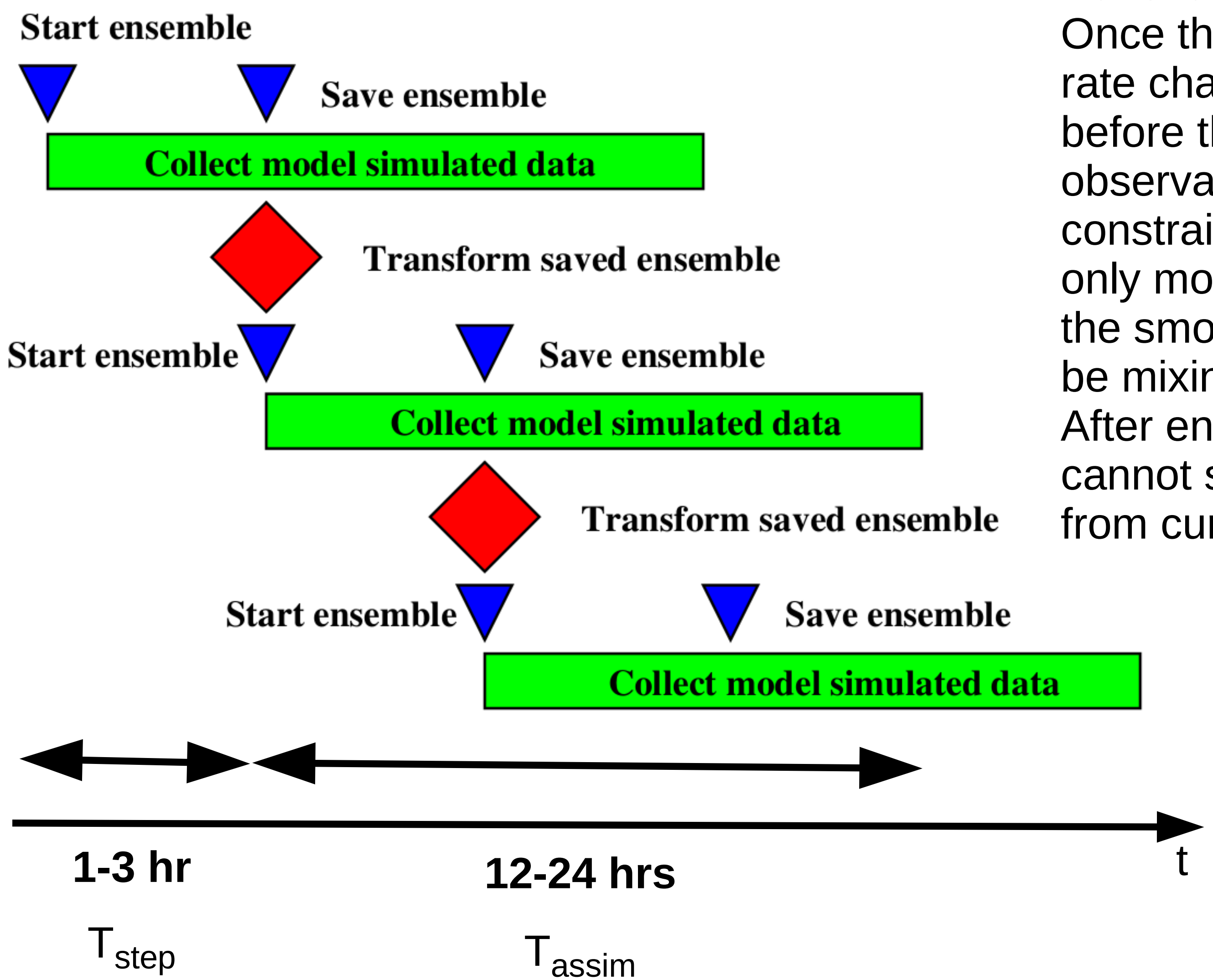
$$F_n = -\frac{N B_i}{\tau}$$

Convection + two hemispheres

$$\frac{D_\perp N}{Dt} = \frac{F_N + F_S}{B_i}$$

$$n_{\mathrm{sat}} = 10^{A+BL}\ \mathrm{cm}^{-3}$$

Ensemble Kalman Smoother



Rationale for using a smoother:
Once the electric field or the refilling rate changes it can take many hours before the effect is seen in the observations. The observations do not constrain the current model inputs, only model inputs some hours ago. If the smoother runs for too long we will be mixing in irrelevant observations. After enough time has elapsed we cannot say anything about the past from current observations.

All of the code parallelized with MPI, including parallel matrix operations.

Comparison with Numerical Weather Prediction

Operational forecasting

The UK Met Office runs its global NWP models on a 6-hourly basis. Forecasts for the first 6 hours of the last run are combined with all the **observational data** received over that period to produce the starting point for the next forecast.

In order to meet deadlines, there has to be a data cut-off time. Data arriving after that time are used later to re-compute the first 6 hours' forecast so getting the best start possible for the following run.

Limited area or meso-scale models are run in the same way. All this is on a "best endeavours" basis and users should always bear in mind the **limitations to forecast accuracy** .

Calculating it

Physicists call weather forecasting an initial value problem. At the initial time, T=0, the rates of change of each weather element can be calculated. That allows estimates to be made a short time, a few minutes, ahead, of winds, temperatures, pressures, water vapour and liquid water. These new values then provide a starting point to calculate for the next few minutes.

The UK Met Office global grid (2017) has a spacing of about 11 km, 0.1 degree lat/lon in the horizontal. There are 4,916,200 grid point with, 70 levels at each from about 20 m to 80 km above the surface of the earth.

Chaos and Forecast Ensembles

Chaos is always a problem – it has been suggested that a single butterfly flapping its wings could lead to major weather systems. In fact, it cannot but small disturbances can and often do grow into large ones.

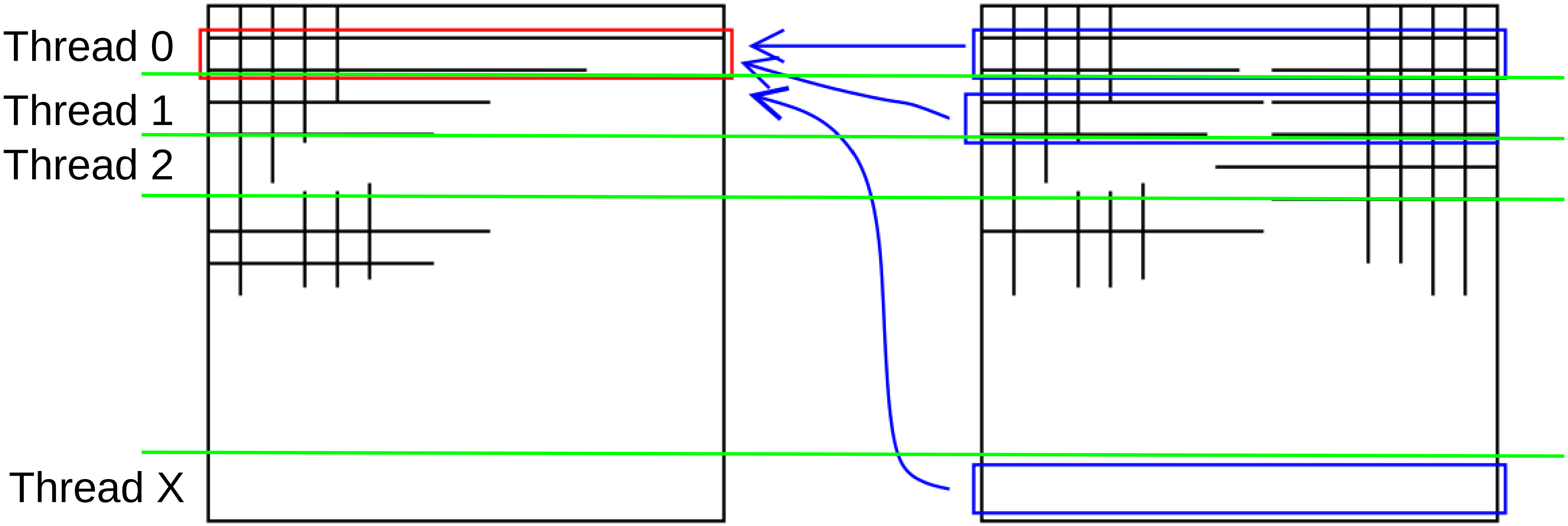
There are always uncertainties in detail of weather analyses. These lead to uncertainties in the deterministic forecast.

Model ensembles are a way of tackling this problem. After running the forecast model, small variations can be put into the analysis that are compatible with the original data. The forecast can be run many times and a spread of results obtained. The spread of these indicates the degree of uncertainty in the deterministic forecast.

(https://weather.mailasail.com/Franks-Weather/Numerical-Weather-Prediction)

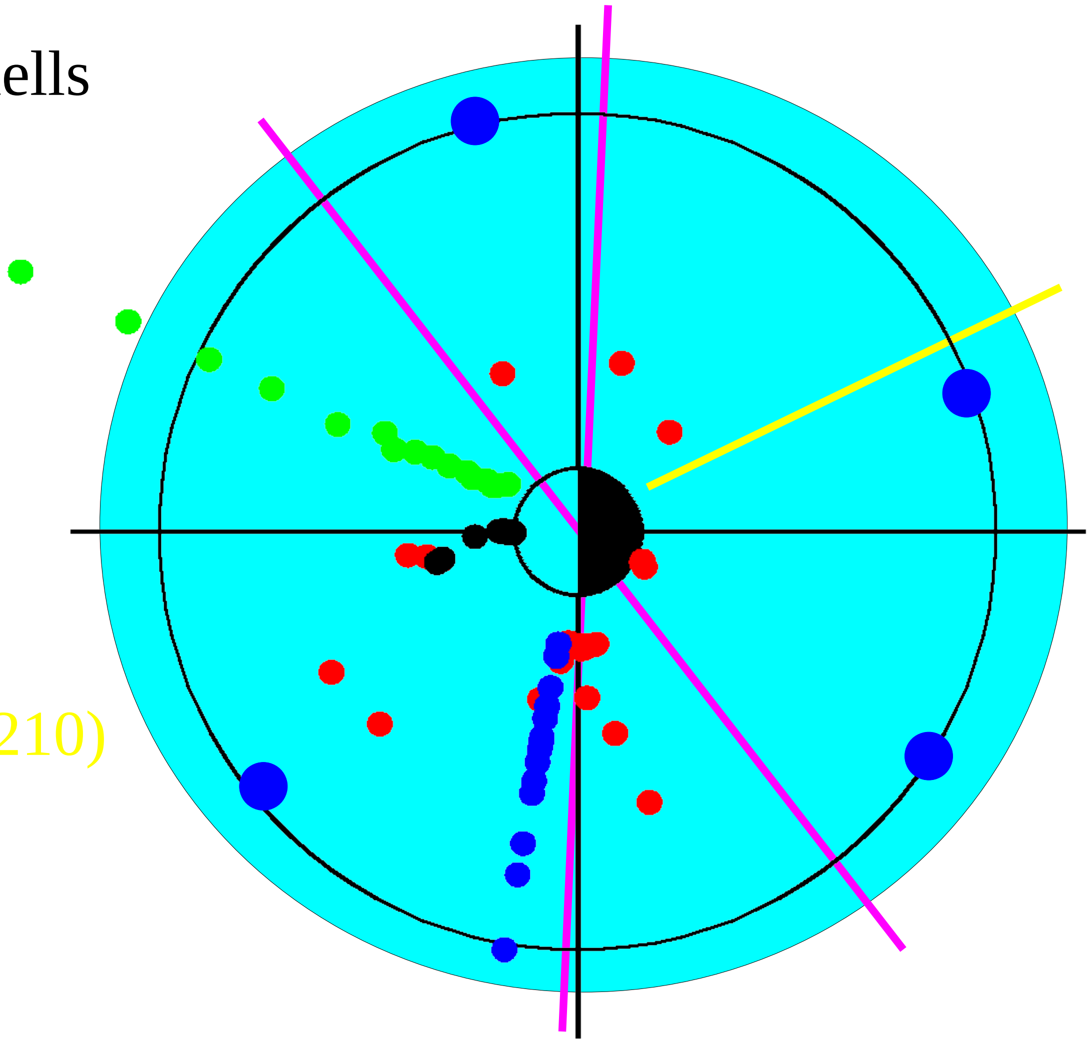
Parallelization

- A typical run may involve an ensemble of hundreds of models
- To make efficient use of computing resources the code is parallelized with MPI. Each individual model resides on one thread. Analysis operations, involving matrix multiplications, decompositions, and inversions, are parallelized across all treads.

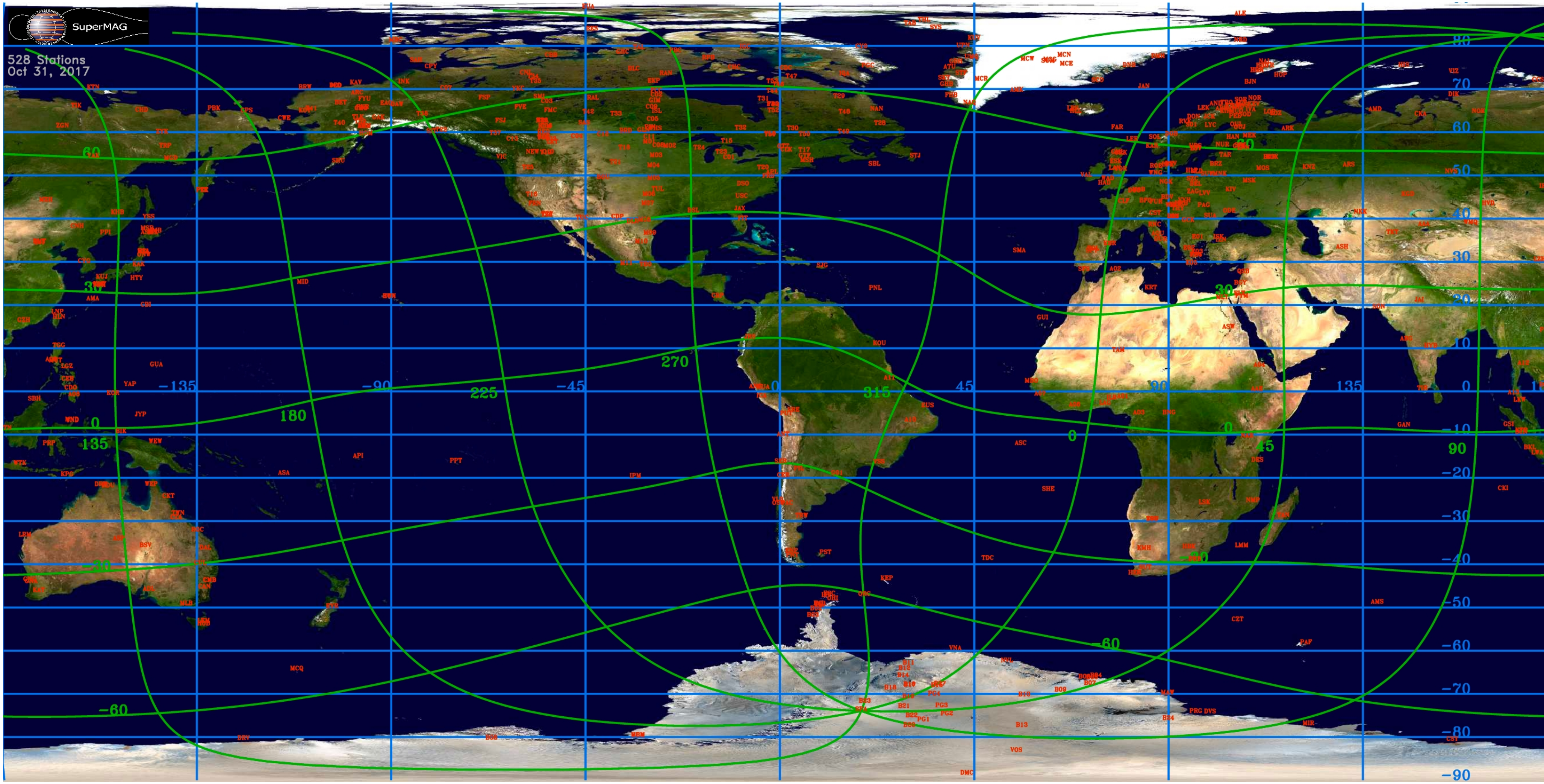


Observations – part 1 - the dream

- AWDANet – VLF whistlers electron density multiple L-shells at each station. Nightside.
- EMMA – FLR mass density. Dayside. L=1.6-6.7
- McMac and Canopus – FLR mass density. Dayside. L=1.6-12
- SAMBA – FLR mass density Dayside. L=1.0-3 (5-6)
- Asia-pacific magnetometers (210)
- DMSP
- LANL-GEO
- EUV



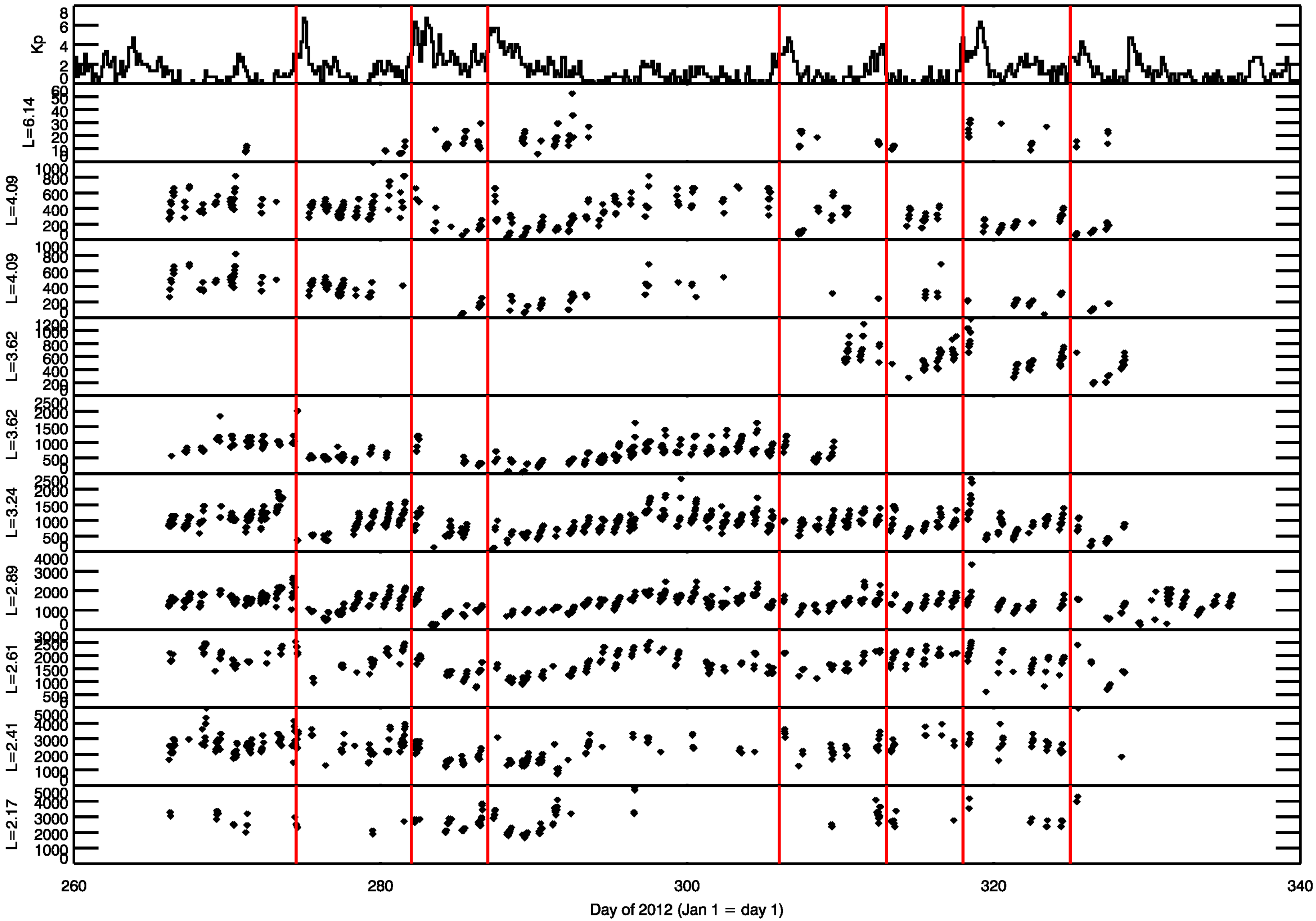
Observations – part 2 - the dream continues



Observations – part 3

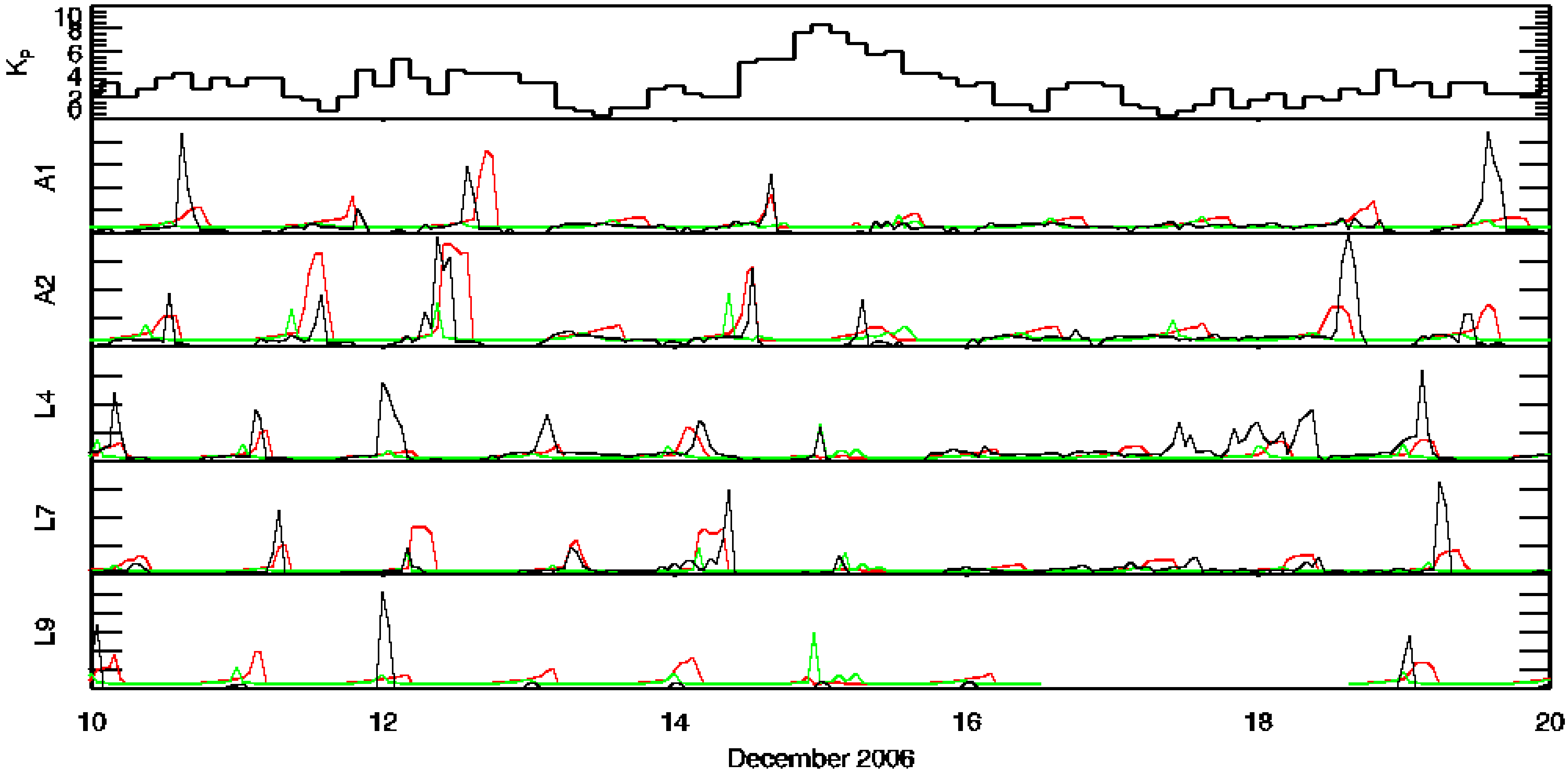
- Summary of possible data types:
- In-situ density measurements from satellites
- Ground-based ULF measurements of mass density
- Ground-base VLF measurements of electron density
- LEO satellites detections of the plasmapause
- Upward-looking GPS on LEO satellites
- Other types of data:
- Detections of the plasmapause from satellite data

Observations – current reality

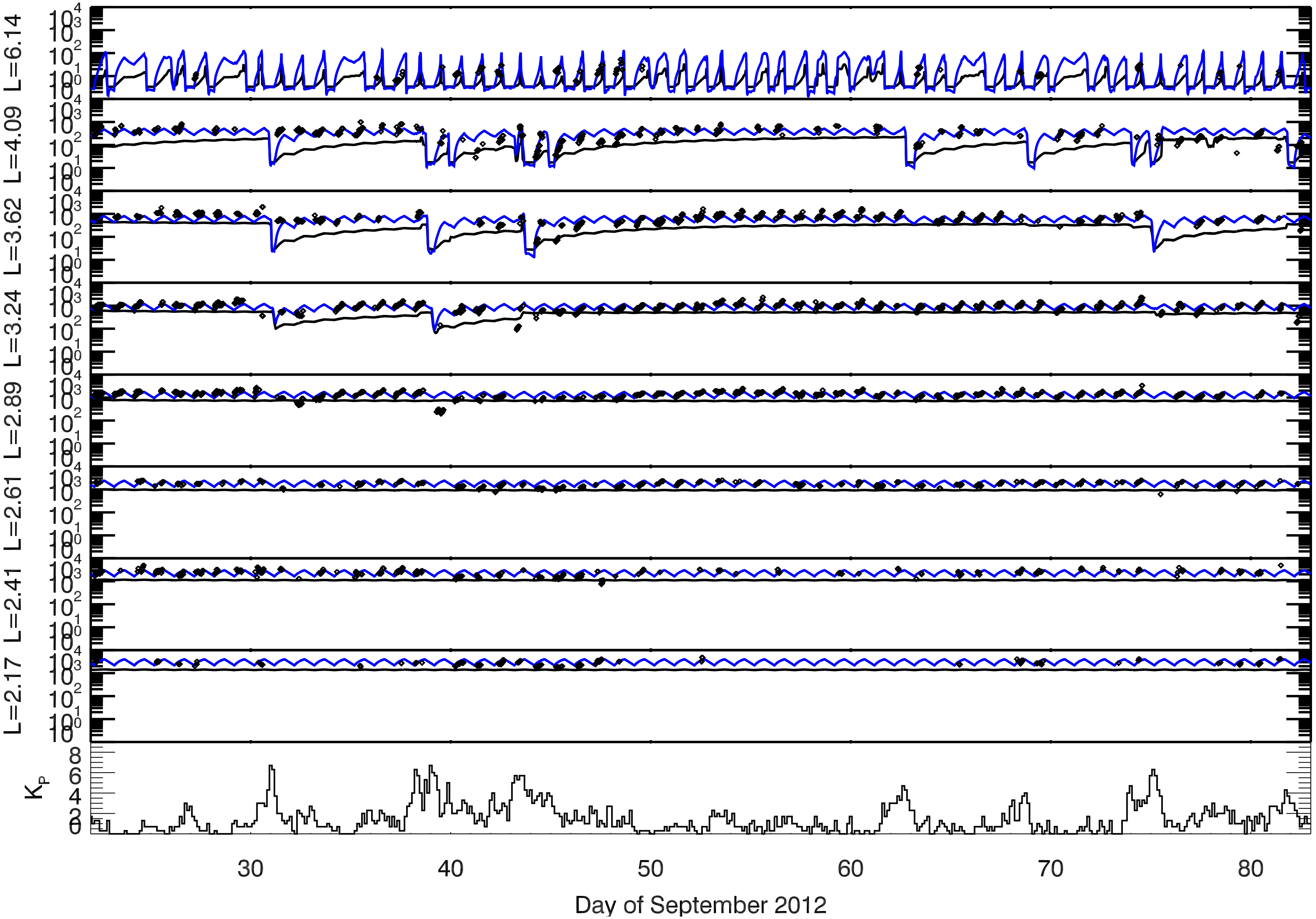


Model Data Comparisons (1)

Gallagher et al. (1995) Sojka et al. (1986)



Model Data Comparisons (2)

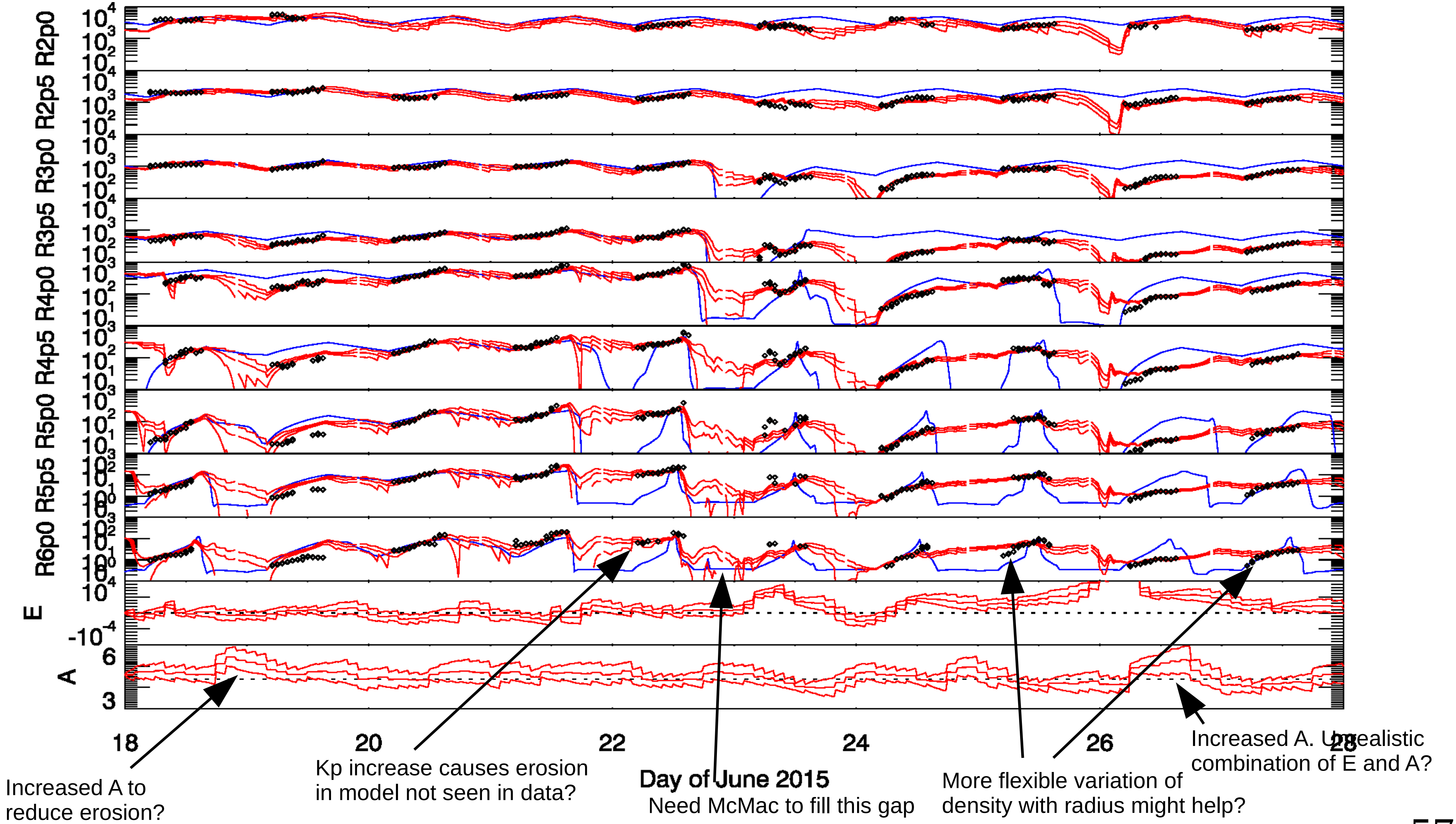


Black symbols: FLR mass density measurements

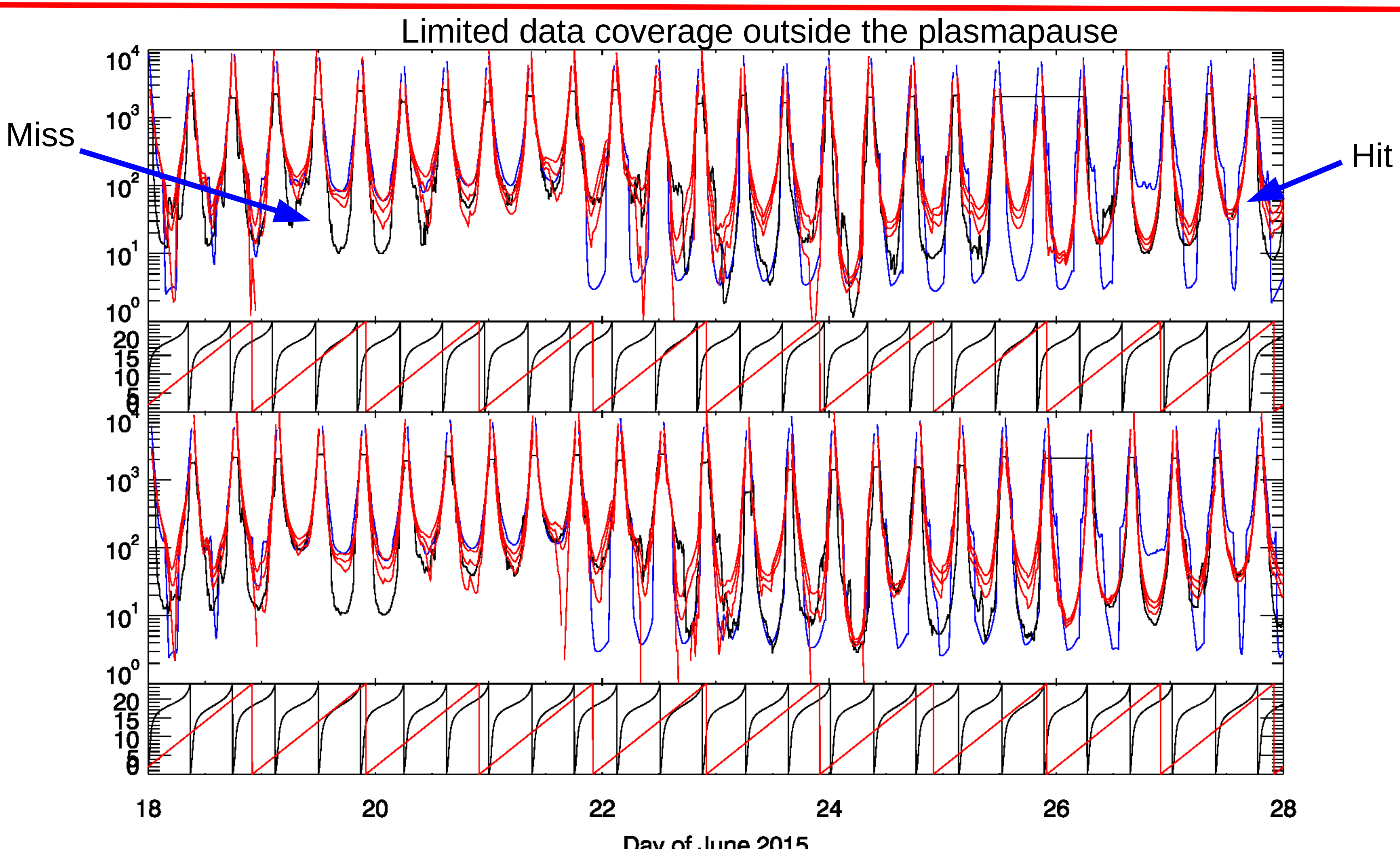
Black curves: Original DGCPM

Blue curves: Revised DGCPM to match observations

A Data Assimilation Run

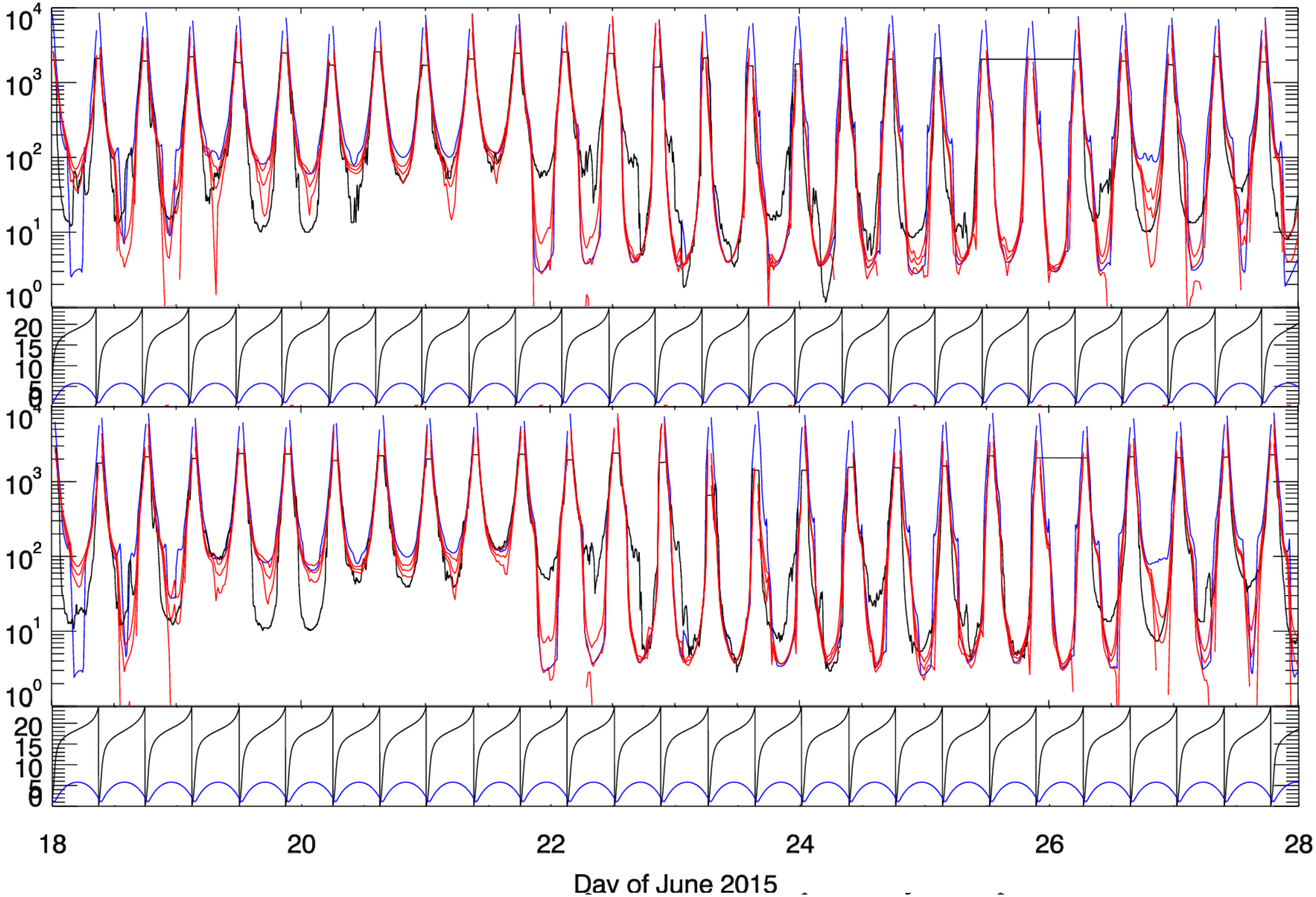


Comparison with Van Allen Probes



Another Data Assimilation Run: Data Coverage

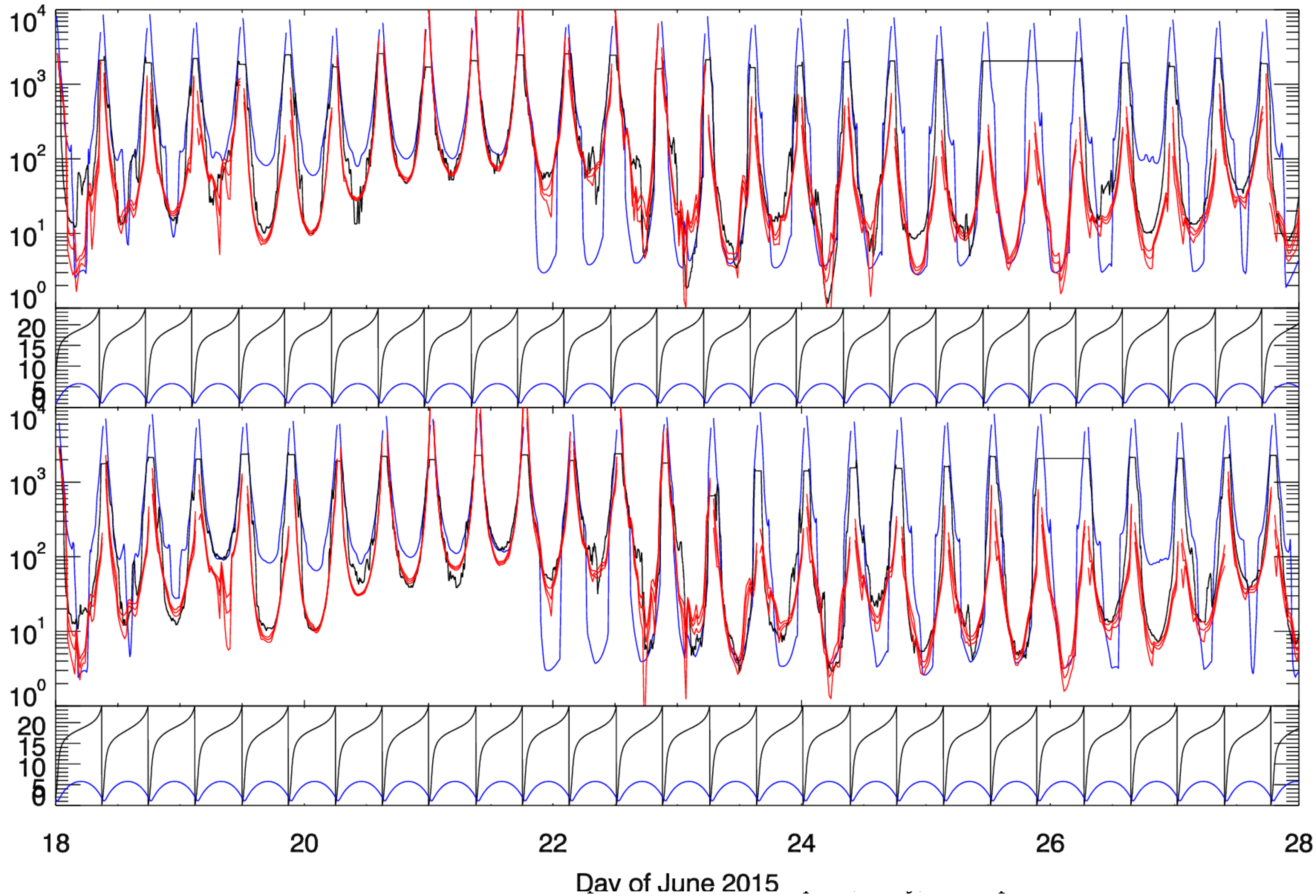
Using a neural net model which includes observations from outside the plasmapause



Using Samples from L=2-6

Another Data Assimilation Run: Data Coverage

Using a neural net model which includes observations from outside the plasmapause



Using Samples from L=5-8

Discussion

- We are using a simple plasmasphere model:
- Centered dipole magnetic field, no tilt, no stretching, no dynamics
- Simple electric field model
- Simple description of refilling and loss, no seasonal effects, no dependence on UV flux/F10.7
- It is amazing this works at all

Future

- Incorporate a dynamic magnetic field (e.g. Tsyganenko model)
  - This will probably significantly increase computation time
- Incorporate better electric field mode: Weimer model plus SAPS
- Incorporate seasonal effects
- Incorporate a better model for refilling and loss (TBD)

Summary and Conclusions

- We introduced the theory of data assimilation
- We discussed the linear Kalman Filter with an example
- We developed the Ensemble Kalman Filter
- We introduced a very simple model of the plasmasphere
- We incorporated the model into a data assimilation framework
- We demonstrated that believable results can be produced despite the simplicity of the model and the sparseness of the data