Data Assimilation Methods

Introduction

In our previous lecture, we developed two analogous statistical frameworks for data assimilation in one dimension: least-squares minimization and cost-function minimization. Both approaches involve identifying the optimal combination of background estimates with observations, in light of the estimated errors associated with each, to produce an updated analysis that is as close to the "true" atmospheric state as possible. We also presented the multidimensional analog to the least squares problem, showing that the same basic tenets underlie the multidimensional problem.

In this lecture, we wish to explore several of the most widely used data assimilation algorithms in greater detail. We begin with three- and four-dimensional variational data assimilation (3DVar and 4DVar), cost-function minimization methods that traditionally use flow-independent background error covariance matrix specifications. We close with the extended and ensemble adjustment Kalman filters, least-squares minimization methods that use flow-dependent specifications for the background error covariance matrix. The variational data assimilation material is provided mostly as a continued introduction to basic data assimilation concepts; our focus through the rest of the semester lies instead with ensemble Kalman filter approaches.

The data assimilation algorithms considered in this lecture are not the only such algorithms that exist. For instance, hybrid ensemble-variational schemes make use of an ensemble of background estimates to provide a flow-dependent estimate of the background error covariance matrix for use in a variational data assimilation scheme. Other non-hybrid schemes exist. Thus, the focus of this lecture is not to provide background regarding every possible algorithm but instead to demonstrate how the fundamental principles outlined in the previous lecture are employed within widely used data assimilation algorithms.

Three-Dimensional Variational Data Assimilation

Variational data assimilation algorithms use iterative methods to minimize a cost function that reflects the departure of the analysis from the background and observation given their respective errors. Recall that the one-dimensional formulation for cost-function minimization is given by:

$$J(T_a) = J(T_o) + J(T_b) = \frac{(T_a - T_o)^2}{\sigma_o^2} + \frac{(T_a - T_b)^2}{\sigma_b^2}$$

Each term of the cost function is equal to the squared error relative to the analysis divided by its respective error variance. With knowledge of the observation, background, and error variances of each, one can find the analysis temperature T_a that minimizes the cost function by taking the first partial derivative of J with respect to T_a, setting the result to zero, and solving for T_a.

The multidimensional formulation for the cost function takes the form:

$$J\left(\vec{x}_a\right) = J\left(\vec{x}_b\right) + J\left(\vec{y}\right) = \frac{\left(\vec{x}_a - \vec{x}_b\right)\left(\vec{x}_a - \vec{x}_b\right)^T}{\vec{B}} + \frac{\left(\vec{H}\left(\vec{x}_a\right) - \vec{y}\right)\left(\vec{H}\left(\vec{x}_a\right) - \vec{y}\right)^T}{\vec{R}}$$

Definitions of the terms listed above may be found in the previous lecture notes. The exponent *T* refers to the transpose matrix. It is hopefully apparent, however, that this formulation is identical to that for the one-dimensional problem apart from the added dimensionality: each term is equal to the squared error relative to the analysis weighted by its respective error covariance.

Similar to the one-dimensional problem, the cost function minimum can be obtained by taking the gradient (across model space) of the cost function and seeing it equal to zero. The analytic expression for the gradient of the cost function is given by:

$$\nabla J\left(\vec{x}_a\right) = \frac{\left(\vec{x}_a - \vec{x}_b\right)}{\vec{B}} + \frac{\vec{H}\left(\vec{x}_a\right)^T\left(\vec{H}\left(\vec{x}_a\right) - \vec{y}\right)}{\vec{R}} = 0$$

Note that this gradient is not taken over physical space but rather is taken across the model space defined by **x**. It is not computationally feasible to obtain an analytic solution for the analysis in the multidimensional problem in this way. Instead, an iterative procedure is typically used to minimize the cost function and obtain the updated analysis. One might start by assuming that the analysis equals the background to obtain an initial cost function estimate. In the next iteration, one might assume that the analysis equals the observations to obtain a second cost function estimate. The third and subsequent iterations would proceed to find the analysis that minimizes the cost function. Fig. 1 provides an example of cost-function minimization in a two-dimensional model space.

Practical implementations of an iterative procedure for cost-function minimization are designed such that the minimum is approached over a relatively small number of iterations. In this regard, the goal of the iterative procedure is to extract the greatest amount of minimization without passing a point where the added computational expense of further minimizing the cost function outweighs the benefit to the analysis of doing so. The Laplacian of the cost function, defining the slope of the cost function's gradient (e.g., how quickly you are approaching the cost function's minimum), can be used to inform this minimization procedure. In practice, on the order of 100 iterations may be required to minimize the cost function.

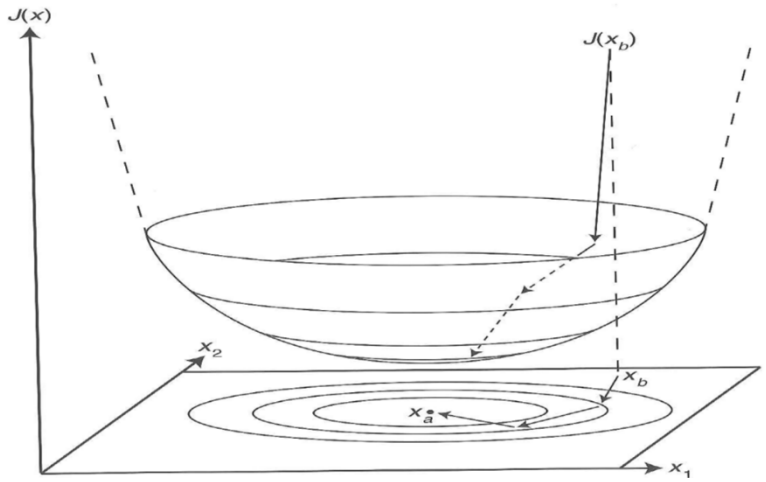
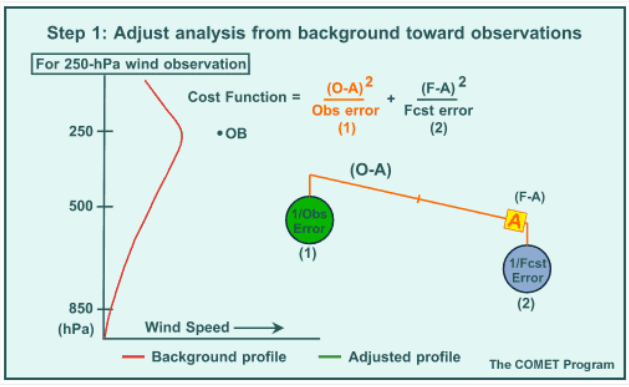


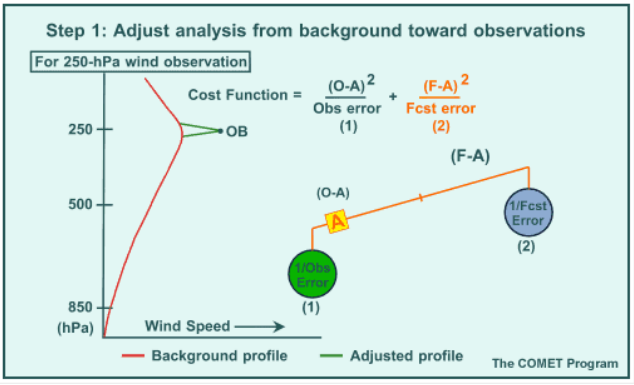
Figure 1. Idealized schematic showing the process of cost function minimization within a two variable (x_1, x_2) model space. In this example, the cost function takes the shape of a parabola. The background $\mathbf{x_b}$, where $J(\mathbf{x_b})$ is minimized, provides the initial guess for the analysis. Two iterations are used in this example to find the cost function minimum that defines the analysis $\mathbf{x_a}$. Figure reproduced from Warner (2011), their Fig. 6.12.

We can also consider a practical example of three-dimensional variational data assimilation. All figures in this example are reproduced from the UCAR MetEd tutorial, "Understanding Assimilation Systems: How Models Create Their Initial Conditions" (account and login required).

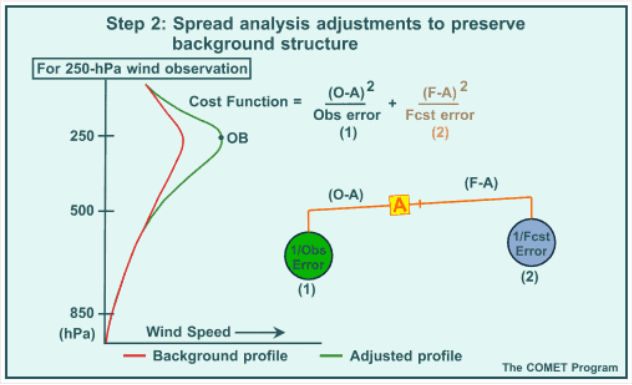
Let us consider a wind observation at 250 hPa (OB). The first guess for the analysis (A) is provided by the background (red line). From this, an initial cost function value equal to the cost of the observation (assuming it is imperfect), as the cost of the background is zero, is obtained.



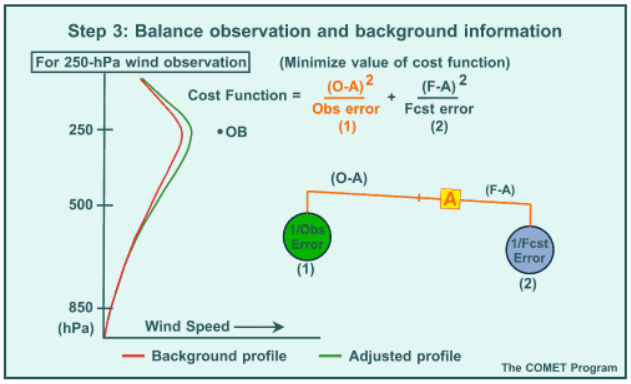
The first iteration occurs as the analysis is adjusted to match the observation. Another cost function value, equal to that of the background as the cost of the observation is zero, is obtained. Implicitly, both it and the initial cost function estimate are too large, with further iteration needed to determine the minimum cost function value.



Adjusting the analysis to match the observation has resulted in a profile shape that departs significantly from that of the background. The next iteration can involve updating the analysis of this variable at altitudes above and below that of the observation such that the profile shape more closely resembles that of the background. This enables for another cost function estimate to be obtained, one that is smaller than before because of a better overall fit (considering the full profile structure, not just the level of the observation) to the background.



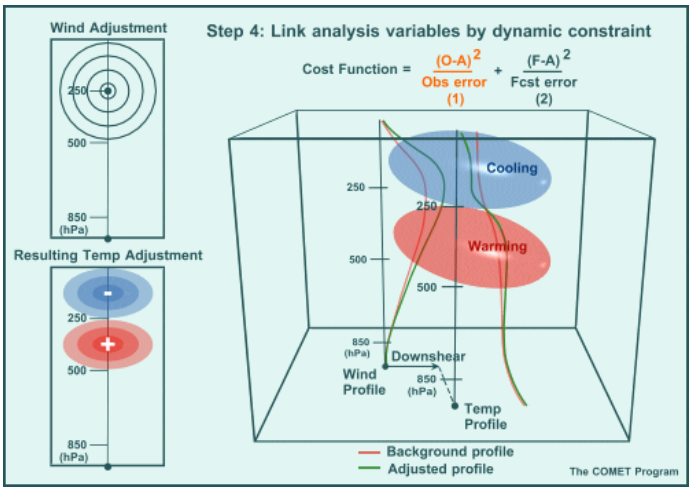
Subsequent iterations can involve updating the analysis to better match the background. The extent to which the analysis is updated to better match the background depends, as we discussed earlier, upon the error characteristics of the background and the observation. This process is repeated until the cost function minimum is found, defining the new wind speed profile.



As we have seen, the assimilated wind observation has produced a minor update to the wind speed at the observation location roughly between 500-150 hPa. Intuitively, we know that other fields such as height and temperature are related to the wind (e.g., through geostrophic and thermal wind balance). Thus, physically, assimilating this wind observation should also update these fields.

Assume that the wind profile depicted in the above figures is of the zonal wind. Assimilating the wind observation increased westerly vertical wind shear between ~500-250 hPa and increased easterly vertical wind shear between ~250-150 hPa. From thermal wind balance, which describes the relationship of the horizontal layer-mean temperature gradient to the vertical wind shear, the magnitude of the horizontal layer-mean temperature gradient should increase in both layers. To accomplish this, warming to the south and cooling to the north below 250 hPa, and cooling to the south and warming to the north above 250 hPa, are needed.

Note that this update is encapsulated within the cost function minimization procedure outlined above; though treated separately here, it actually occurs concurrently with the wind update. It should also be emphasized that this update does *not* necessarily occur from an explicit physical balance statement within the model but rather through the covariance between errors in each field, representing a statistical relationship between the fields. It is hoped that the physical relationship is conveyed through the statistical relationship, but in practice this is approximate at best because of issues like sampling error (estimating correlations with a subset of the true sample).



Let us now consider a 500 hPa temperature observation at the same location as the 250 hPa wind observation. The procedure outlined above is followed to produce an updated analysis for both temperature and its related fields (e.g., the horizontal winds at multiple vertical levels, also via thermal wind balance). As above, this also is encapsulated within the cost function minimization procedure outlined previously; the analysis is determined iteratively in light of all observations. This results in an analysis that is improved relative to that which would be obtained if only one observation were considered.

Most practical implementations of 3DVar use flow-independent specifications for **B** within their formulation. This, naturally, will have an impact to how the background influences the analysis relative to that if a flow-dependent specification for **B** were utilized. As noted above, however, it is possible to use a hybrid ensemble-variational method to obtain a flow-dependent **B** that may be used within the cost function minimization process.

Four-Dimensional Variational Data Assimilation

Four-dimensional variational data assimilation, or 4DVar, is a generalization of 3DVar to allow for the continuous assimilation of all available observations over some assimilation interval. Recall that the 3DVar formulation of the cost function is given by:

$$J\left(\vec{x}_a\right) = \frac{\left(\vec{x}_a - \vec{x}_b\right)\left(\vec{x}_a - \vec{x}_b\right)^T}{\vec{B}} + \frac{\left(\vec{H}\left(\vec{x}_a\right) - \vec{y}\right)\left(\vec{H}\left(\vec{x}_a\right) - \vec{y}\right)^T}{\vec{R}}$$

The 4DVar analog is similarly expressed as:

$$J\left(\vec{x}_a\left(t_0\right)\right) = \frac{\left(\vec{x}_a\left(t_0\right) - \vec{x}_b\left(t_0\right)\right)\left(\vec{x}_a\left(t_0\right) - \vec{x}_b\left(t_0\right)\right)^T}{\vec{B}_{t_0}} + \sum_{i=0}^{n} \frac{\left(\vec{H}\left(\vec{x}_a\right)_i - \vec{y}_i\right)\left(\vec{H}\left(\vec{x}_a\right)_i - \vec{y}_i\right)^T}{\vec{R}_i}$$

Here, t_0 is the initial/analysis time, t_i is an intermediate time at which one or more observations are assimilated, and t_n is the time at the end of the assimilation window. As compared to 3DVar, there is no change to the background portion of the cost function formulation. The only change is found with the observation portion of the cost function formulation, wherein observations over a series of times between t_0 ($i = 0$) and t_n ($i = n$) rather than just at t_0 are assimilated. Each set of observations are assimilated at the time at which they are taken. Consequently, the analysis state vector in the observation portion of the cost function is that valid at the time of the observation. In the case of observations all being valid at t_0, this formulation is identical to 3DVar.

In contrast to 3DVar, which requires no model integration, the 4DVar algorithm requires forward integration of the model from t_0 to t_n as observations are assimilated. To finalize the analysis at t_0, a backward integration of the model from t_n is required. Backward integration requires the use of an adjoint operator, typically reflecting a backward linear version of the model. As an adjoint operator is specific to a given forecast model, the adjoint must be updated each time that the forecast model is updated. This can be resource-intensive. Furthermore, forward and backward integration during assimilation results in added computational expense for 4DVar versus 3DVar. Historically, this has limited its operational use to the ECMWF model, which is not limited by the same timeliness constraints as (and has a more powerful supercomputer available to it than) NCEP.

In practice, 4DVar seeks to minimize the cost function throughout the assimilation window. It determines the analysis state vector $\mathbf{x_a}$ at t_0 that produces a model solution, given by the forecast state vector $\mathbf{x_f}$, that minimizes the cost function <u>*at all times*</u> between t_0 and t_n (Fig. 2). In the case of only assimilating observations at t_0, this is identical to 3DVar and reflects cost function minimization at only the analysis time t_0. Assimilating observations at the time they were taken over an assimilation window that often extends into the first few hours of the subsequent model forecast results in improved skill of model forecasts initialized using 4DVar relative to 3DVar, as is exhibited by the historical performance of the ECMWF model versus other models.

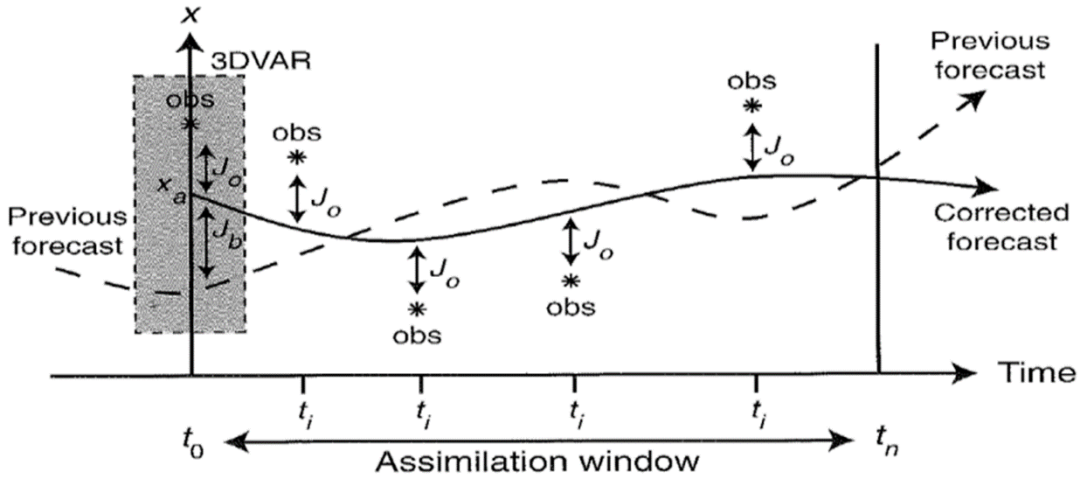


Figure 2. Schematic illustrating the 4DVar process. In this example, a previous model forecast is used to provide the background at t_0, as in 3DVar, but also at all subsequent intermediate times t_i. Observations are assimilated at t_0, all t_i, and t_n, seeking to produce the optimal combination of the background and observations throughout the assimilation window that is a valid solution to the model equations. Here, the model state vectors from the corrected and previous forecasts are identical at t_n, although it is more common for the corrected model state vector to be a better match to the available observations. Figure reproduced from Warner (2011), their Fig. 6.18.

Extended Kalman Filter

The extended Kalman filter is a sequential (i.e., individual observations are assimilated one at a time rather than all at once) implementation of least-squares minimization that allows for a flow-dependent, time-varying background error covariance matrix to the calculated to better determine the weighting matrix **K**.

First, the background estimate of the model state $\mathbf{x_b}$ at time t+1 is obtained by integrating the model $\boldsymbol{M}$ forward from a previous analysis $\mathbf{x_a}$ at time t, i.e.,

$$\overrightarrow{x_b}(t+1) = \overrightarrow{M}_{t \to t+1}\left(\overrightarrow{x_a}(t)\right)$$

Next, the background error covariance matrix **B** valid at time *t*+1 is obtained. This is a function of (a) the forward propagation of analysis errors from time *t* to time *t*+1 and (b) the accumulated model error over the interval between times *t* and *t*+1:

$$\vec{B}(t+1) = \vec{\mathrm{M}}_{t\to t+1}\vec{A}(t)\vec{\mathrm{M}}^{T}_{t\to t+1} + \vec{Q}(t)$$

Here, **A** is the analysis error covariance matrix at time *t*, **Q** is the forecast error covariance matrix over the interval between times *t* and *t*+1, **M** represents the linear forward model operator, and $\mathbf{M}^{\mathbf{T}}$ represents the adjoint of **M**. This equation requires that **A** be known. We will demonstrate how **A** can be obtained shortly. Of these, **Q** can be particularly challenging to estimate or obtain.

With knowledge of **B**, the weighting matrix **K** can be obtained. As before, **K** controls the weight and spread of innovations. In the extended Kalman filter, this matrix is referred to as the *Kalman gain matrix*. It takes an identical form to that considered previously except for time dependence:

$$\vec{K}(t+1) = \frac{\vec{B}(t+1)\mathrm{H}^{\mathrm{T}}(t+1)}{\mathrm{H}(t+1)\vec{B}(\mathrm{t}+1)\mathrm{H}^{\mathrm{T}}(t+1) + \vec{R}(t+1)}$$

Here, the Kalman gain matrix is equal to the background error covariance divided by the total error covariance (background plus observations). H represents the linear forward observation operator and H^{T} represents its adjoint.

Once **K** has been determined, the updated analysis may be obtained. As before, this is equal to the background plus an optimally weighted innovation, i.e.,

$$\vec{x_a}(t+1) = \vec{x_b}(t+1) + \vec{K}(t+1)\left[\vec{y}(t+1) - \vec{H}(t+1)\vec{x_b}(t+1)\right]$$

Here, **y** represents the observations to assimilate, **H** represents the forward operator to convert from model to observation space, and the innovation (in brackets) is given by the observations minus the transformed background estimates. It is implicitly assumed that the innovation is transformed back to model space during the assimilation process.

With an updated analysis, one can then repeat the process outlined above for the next analysis. Before doing so, however, we need to obtain the analysis error covariance matrix **A**. Recall that in the one-dimensional problem, we related the analysis error variance σ_a^2 to the background error variance σ_b^2 as a function of the weighting factor *k*, i.e.,

$$\sigma_a^2 = (1-k)\sigma_b^2$$

A similar formulation can be used to relate **A** to **B** in terms of **K**, i.e.,

$$\vec{A}(t+1) = \left(\vec{I} - \vec{K}(t+1)\vec{H}(t+1)\right)\vec{B}(t+1)$$

Here, **I** is the identity matrix, defined as 1 along the diagonals and 0 elsewhere. All other terms are as previously defined. This equation can be used after the first assimilation time; this means that alternative means of defining **A** during the first assimilation period are needed, however.

Ensemble Kalman Filter

The ensemble Kalman filter extends the extended Kalman filter to an ensemble framework. We start with an ensemble of atmospheric state analyses. Prior to the first assimilation time, this is typically obtained by randomly perturbing a single analysis state. This single analysis state may be drawn from another model's analysis, while the random perturbations are often obtained from randomly sampling a static, climatological background error covariance matrix **B**. The result is an ensemble of initial conditions $\mathbf{x_a}$ containing random variations related to climatological model background errors.

Once the initial ensemble has been generated, the model is integrated to the next analysis time. In so doing, note that the lateral boundary conditions (assuming a limited-area model domain) should be randomly perturbed so as to not constrain the ensemble forecasts toward information provided by an identical lateral boundary condition specification for all ensemble members. The ensemble forecasts valid at the next analysis time provide the first guess $\mathbf{x_b}$. Observations are then assimilated to correct each ensemble member's background so as to provide an updated analysis. No further perturbations to the analysis are applied. The cycle then repeats from here.

The ensemble Kalman filter differs from the extended Kalman filter in how the background error covariance matrix is specified. Recall the generic expressions for the background error variance (one dimension) and background error covariance matrix (multidimensional):

$$\sigma_b^2 = E\left(\varepsilon_b^2\right) = \overline{\left(x_b - x_t\right)^2}$$

$$\vec{B} = \overline{\left(\vec{x_b} - \vec{x_t}\right)\left(\vec{x_b} - \vec{x_t}\right)^T}$$

These represent the mean squared errors in the background estimates relative to the true state. In the ensemble Kalman filter, there exists an ensemble of background estimates. Assuming that these estimates are unbiased, the true state can be approximated relative to the ensemble mean background, i.e.,

$$\vec{x_t} = \vec{\bar{x}_b}$$

This enables the background error covariance matrix to be expressed as:

$$\vec{B} = \overline{\left(\vec{x_b} - \vec{\vec{x}_b}\right)\left(\vec{x_b} - \vec{\vec{x}_b}\right)^T}$$

Here, the exponent of T represents the matrix transpose operator. In this method, no adjoint is needed to obtain **B**. Instead, the ensemble forecasts themselves provide a direct estimate of how analysis errors propagate and how model errors accumulate in time. This is a clear advantage of the ensemble Kalman filter relative to the extended Kalman filter, although the ensemble approach does come with the disadvantage of being very computationally expensive. It should be noted that this **B** is also flow-dependent, a clear advantage relative to 3DVar and 4DVar.

The computation of **K** and $\mathbf{x_a}$ follow from that for the extended Kalman filter. Similar to the computation of **B**, the analysis error covariance matrix can be computed as:

$$\vec{A} = \overline{\left(\vec{x_a} - \vec{\vec{x}_a}\right)\left(\vec{x_a} - \vec{\vec{x}_a}\right)^T}$$

Here, the true state has been approximated by the ensemble mean analysis state.

We will spend much of the remainder of the semester discussing the ensemble Kalman filter, and thus further details are left for those discussions. More information about ensemble Kalman filters, particularly operational implementations thereof, is provided by Whitaker and Hamill (2002, *Mon. Wea. Rev.*), Hakim and Torn (2008, *Meteor. Monogr.*), and Houtekamer and Zhang (2016, *Mon. Wea. Rev.*).