PSU WRF EnKF/4DVar Hybrid Regional Data Assimilation System: Technical Notes

Development Team:

Yue Ying, Xingchao Chen, Yunji Zhang, Masashi Minamide, Robert Nystrom, Hans Chen,

Jonathan Poterjoy, Christopher Melhauser,

Yonghui Weng, Zhiyong Meng, Altug Aksoy,

Fuqing Zhang

Department of Meteorology and Atmospheric Science, and Center for Advanced Data Assimilation and Predictability Techniques, The Pennsylvania State University, University Park, Pennsylvania

June 2018

Table of Contents

1 System overview

The Pennsylvania State University (PSU) regional hybrid data assimilation system (hereafter referred to as the "PSU system") started as a simple proof-of-concept ensemble Kalman filter (EnKF) code for radar data assimilation (Snyder and Zhang 2003). It was later adopted to work with the Weather Research and Forecasting (WRF) model to perform regional data assimilation (Zhang et al. 2006; Meng and Zhang 2007; 2008 a, b). And it was further developed into an operational hurricane data assimilation and forecast system that assimilates both conventional and aircraft reconnaissance data (Zhang et al. 2009; 2011; Weng and Zhang 2012). Over the course of development in the past decade, the PSU system becomes more versatile. Currently, the supported data assimilation methods include EnKF, 3DVar and 4DVar (using WRFDA package), as well as hybrid methods such as E4DVar and 4DEnVar. More types of observations are supported recently, including the airborne Doppler radar radial velocity, and the satellite brightness temperature.

The WRF model is a fully compressible, nonhydrostatic mesoscale model (Skamarock et al. 2005). The vertical coordinate follows the terrain using hydrostatic pressure, and the model uses an Arakawa C grid. Prognostic variables are the column mass of dry air, velocities, potential temperature, and geopotential with optional variables including turbulent kinetic energy and any number of scalars such as water vapor mixing ratio, rain/snow mixing ratio, and cloud water/ice mixing ratio. The model domain is typically configured to perform a regional simulation of the weather system of interest at a convection-permitting resolution. For example, the hurricane operational forecast setting uses three nested domain with horizontal grid spacing of 27, 9, and 3 km. It is possible to configure the model to simulate systems across a wide range of scales.

The PSU system supports a variety of ensemble and variational data assimilation methods. The EnKF was first proposed in the geophysical literature by Evensen (1994) as an approximation to the Kalman filter (1960), which provides the optimal state estimation for a linear system with Gaussian errors. The Kalman filter was derived under the Bayesian estimation framework, it combines information from model forecast and available observations and their respective uncertainties. The EnKF uses an ensemble of model forecasts to characterize a flow-dependent background error covariance, which helps better propagate observed information in space and time to those variables that are unobserved. In geoscience applications, the EnKF typically faces the challenge of limited ensemble size comparing to dimension of the state, which will cause sampling noises in the estimated error covariance and requires ad hoc localization techniques to reduce the dimensionality of the analysis. Inflation of the ensemble spread is often needed to account for unrepresented error sources in the system. A comprehensive review of the development of EnKF is provided by Houtekamer and Zhang (2015).

More recently, hybrid data assimilation methods are proposed to combine the merit from both ensemble and variational methods. Using WRFDA package as variational component, the PSU system now fully support the state-of-the-art hybrid methods, such as the E4DVar and the 4DEnVar. The PSU system uses bash control scripts to coordinate the workflow and I/O of data among modules. This document provides a detailed technical description of the PSU system.

2 Basic EnKF algorithms

2.1 Ensemble Kalman Filter

Consider assimilating observations y^o to update a model state x with n variables. The Bayesian framework for data assimilation combines the information from a prior distribution for x and likelihood function of x given the observation y^o and arrives at an updated posterior distribution for x according to the Bayes' Theorem. The Kalman filter assumes the involved probability distributions are all Gaussian (normal distribution) characterized by their mean and covariance (**Fig. 2.1**).

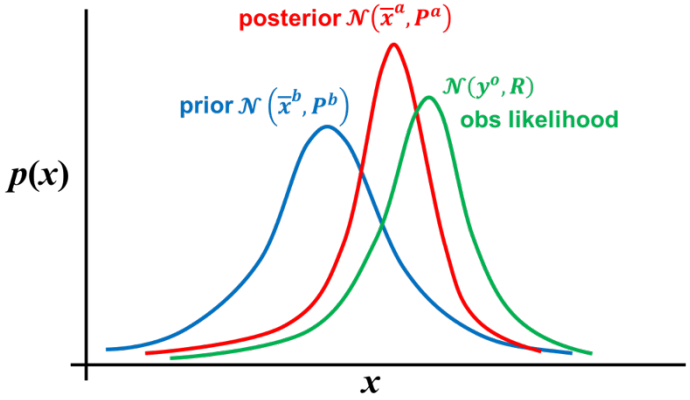

Fig. 2.1. A schematic of probability distributions during Bayesian data assimilation. The prior distribution is updated to posterior distribution by the observation likelihood function.

Ensemble Kalman filter (EnKF) uses a Monte Carlo sampling strategy to estimate the mean and covariance of prior distribution $N(\overline{x}^b, P^b)$ using an ensemble. Let x_k^b be the prior ensemble members for $k = 1, 2, \ldots, N$, where N is the ensemble size. The prior ensemble mean is found by

$$\overline{x}^b = \frac{1}{N}\sum_{k=1}^{N} x_k^b, \tag{2.1}$$

and the prior error covariance can be calculated as

$$P^b = XX^T, \tag{2.2}$$

where $X = \frac{1}{\sqrt{N-1}}\left(x_1'^b, x_{2,}'^b \ldots, x_N'^b\right)$ is the ensemble perturbation matrix. The ensemble perturbations are calculated as $x_k'^b = x_k^b - \overline{x}^b$, for $k = 1, 2, \ldots, N$.

The observation y^o is a vector containing p observations, whose error covariance is R. The nonlinear observation operator that converts model states to observations is denoted as $h(\cdot)$, and H is the corresponding tangent linear observation operator. The EnKF update equation for ensemble mean can be written as

$$\overline{x}^a = \overline{x}^b + K\left[y^o - h\left(\overline{x}^b\right)\right], \tag{2.3}$$

where K is the Kalman gain

$$K = P^b H^T (HP^b H^T + R)^{-1}. \tag{2.4}$$

The ensemble perturbations are updated as

$$x_k'^a = x_k'^b + K\left[y_k'^o - Hx_k'^b\right], \text{ for } k = 1, 2, \ldots, N, \tag{2.5}$$

so that the posterior error covariance satisfies

$$P^a = (I - KH)P^b. \tag{2.6}$$

This is the formulation of EnKF with perturbed observation (Evensen 1994; Houtekamer and Mitchell 1998), in which $y_k'^o$ is randomly drawn from the distribution $N(0, R)$ for each member k.

2.2 Serial EnKF with square root modification

The simultaneous assimilation of all observations requires the calculation of a matrix inversion $(HP^bH^T + R)^{-1}$ that can be costly. For the PSU system, we employ a serial algorithm instead. Instead of solving (2.3) and (2.5) in one step, observations (j = 1, 2, …, p) are assimilated serially, and the state x is updated iteratively one observation at a time. The following is a pseudo-code for the serial EnKF:

Table 2.1. Pseudo code of serial EnKF with perturbed observation.

$x_k^{(1)} = x_k^b$, for k = 1, 2, …, N

for j = 1, 2, …, p

$\overline{x}^{(j)} = \frac{1}{N}\sum_{k=1}^{N} x_k^{(j)}$ (2.7)

for k = 1, 2, …, N

$x'^{(j)}_k = x_k^{(j)} - \overline{x}^{(j)}$ (2.8)

$y_{j,k}^{(j)} = h_j\left(x_k^{(j)}\right)$ (2.9)

end

$\overline{y}_j^{(j)} = \frac{1}{N}\sum_{k=1}^{N} y_{j,k}^{(j)}$ (2.10)

for k = 1, 2, …, N

$y'^{(j)}_{j,k} = y_{j,k}^{(j)} - \overline{y}_j^{(j)}$ (2.11)

end

$\mathrm{var}\left(y_j^o\right) = R_{jj}$ (2.12)

$\mathrm{var}\left(y_j^{(j)}\right) = H_j P^{(j)} H_j^T = \frac{1}{N-1}\sum_{k=1}^{N}\left(y'^{(j)}_{j,k}\right)^2$ (2.13)

$\mathrm{cov}\left(x^{(j)}, y_j^{(j)}\right) = P^{(j)} H_j^T = \frac{1}{N-1}\sum_{k=1}^{N}\left(x'^{(j)}_k \cdot y'^{(j)}_{j,k}\right)$ (2.14)

$K_j = \frac{\mathrm{cov}\left(x^{(j)}, y_j^{(j)}\right)}{\mathrm{var}\left(y_j^o\right) + \mathrm{var}\left(y_j^{(j)}\right)}$ (2.15)

$\overline{x}^{(j+1)} = \overline{x}^{(j)} + K_j\left(y_j^o - \overline{y}_j^{(j)}\right)$ (2.16)

for k = 1, 2, …, N

$x'^{(j+1)}_k = x'^{(j)}_k + K_j\left(y'^o_{j,k} - y'^{(j)}_{j,k}\right)$ (2.17)

$x_k^{(j+1)} = \overline{x}^{(j+1)} + x'^{(j+1)}_k$

end

end

$x_k^a = x_k^{(p+1)}$.

Note that h_j is the nonlinear observation operator for the j-th observation; H_j denotes the j-th row of H; superscript (j) denotes the state of a variable at the j-th iteration, namely after the assimilation of the previous j-1 observations. By the end of observation loop the state x has been updated p times and is output as the analysis. When observation errors are uncorrelated (R is a diagonal matrix), the serial algorithm is equivalent to the original simultaneous algorithm. **Appendix A** provides a proof of such equivalence.

The perturbed observation EnKF is a stochastic formulation, in which random sampling of the observation error $y'^{o}_{j,k}$ may introduce sampling noises to the filter solution. For the PSU system, we use a deterministic formulation of EnKF. In (2.17), we do not draw random perturbation to characterize observation error, a zero is used instead of $y'^{o}_{j,k}$. Whitaker and Hamill (2002) showed that the lack of observation error perturbations in (2.17) will cause the analysis ensemble to be under-dispersive, and thus requires a modification term to compensate the missing error variance. (2.17) becomes

$$x'^{(j+1)}_{k} = x'^{(j)}_{k} + \phi K_j \left(-y'^{(j)}_{j,k}\right), \tag{2.18}$$

where

$$\phi = \left(1 + \sqrt{\frac{\mathrm{var}\left(y^{o}_{j}\right)}{\mathrm{var}\left(y^{(j)}_{j}\right)+\mathrm{var}\left(y^{o}_{j}\right)}}\right)^{-1} \tag{2.19}$$

is the modification term. The resulting algorithm is call an ensemble square root filter (EnSRF).

2.3 Covariance Localization and Inflation

The dimension of model state, n, is typically much larger than the ensemble size N. The limited ensemble size causes the sample estimated prior error covariance P^b to be rank deficient, and results in the covariance between observation and model state, $cov\left(x^{(j)}, y^{(j)}\right)$, to be contaminated with sampling noises when they are at a larger distance. Localizing the impact from distant observations will remedy the negative impact from these sampling noises (Hamill et al. 2001; Houtekamer and Mitchell 2001). In update equations (2.16) and (2.17), the Kalman gain is replace with $\rho_j \circ K_j$, where the circle denotes an element-wise (Schur) product, and ρ_j is a localization function that maximizes at the j-th observation location and tapers to zero at the cutoff distance. A typical choice of localization function is the Gaspari and Cohn (1999) fifth-order polynomial, which is similar to the shape of a Gaussian yet has compact support. The cutoff distance is also referred as the radius of influence (ROI).

When extra error sources (model error, representation error, etc.) in the system are not well accounted for in EnKF, the filter solution (ensemble mean) can deviate from the truth yet the ensemble spread keeps decreasing. This phenomenon is called “filter divergence”, where filter eventually ignores all observation because of the collapse of ensemble spread. To prevent this, one can inflate the ensemble spread every cycle either before or after the assimilation step. Multiplicative inflation (Anderson and Anderson 1999) inflate the ensemble spread by multiplying an inflation factor,

$$x'_k \leftarrow \lambda x'_k, \tag{2.20}$$

and additive inflation (Mitchell and Houtekamer 2000) draws a random noise from a given error distribution and add it to the ensemble perturbations. Covariance relaxation (Zhang et al. 2004) is another form of inflation by mixing the analysis ensemble perturbations with the prior ones that are larger,

$$x'^a_k \leftarrow (1-\alpha)x'^a_k + \alpha x'^b_k. \tag{2.21}$$

Another option for relaxation is the relax-to-prior-spread method proposed by Whitaker and Hamill (2012), in which the analysis ensemble is inflated using a multiplicative factor which is in between prior and posterior spread, $x'^a_k \leftarrow \left[\alpha \frac{\sigma^b - \sigma^a}{\sigma^a} + 1\right] x'^a_k$, where σ^b and σ^a are the prior and posterior ensemble spread, respectively.

2.4 Observation prior calculation

The calculation of observation priors in (2.9) can be very costly if the h operator is complicated, such as the Radiative Transfer Model (RTM) for satellite observations. To avoid this computational cost, we calculate the observation priors before the assimilation step, and update them during the assimilation loop.

Table 2.2 is a pseudo-code for the final algorithm with all the aforementioned modifications that is used in the EnKF component of the PSU system. (2.24) – (2.26) calculate the initial values of observation priors and store them as ensemble mean and perturbations similar to the way state variables are initialized. During the assimilation step, a group of update equations (2.33) – (2.36) are added for these observation values $\overline{y}_l$ and $y'_{l,k}$ corresponding to the updated state variables. These equations are derived by left multiplying the update equations for $\overline{x}$ and x'_k (2.27) – (2.32) with H_l.

Note that ϕ is the square root modification term as in (2.19), ρ_j is the localization function for the j-th observation, and ρ_{lj} is an element in ρ_j that depends on the distance of the l-th observation to the j-th observation.

Table 2.2. Pseudo code of the serial EnSRF algorithm used in the PSU system.

$x_k^{(1)} = x_k^b$, for k = 1, 2, …, N

$\overline{x}^{(1)} = \frac{1}{N}\sum_{k=1}^{N} x_k^{(1)}$ (2.22)

for k = 1, 2, …, N

$x'^{(1)}_k = x_k^{(1)} - \overline{x}^{(1)}$ (2.23)

end

for j = 1, 2, …, p

for k = 1, 2, …, N

$y_{j,k}^{(1)} = h_j\left(x_k^{(1)}\right)$ (2.24)

end

$\overline{y}_j^{(1)} = \frac{1}{N}\sum_{k=1}^{N} y_{j,k}^{(1)}$ (2.25)

for k = 1, 2, …, N

$y'^{(1)}_{j,k} = y_{j,k}^{(1)} - \overline{y}_j^{(1)}$ (2.26)

end

end

for j = 1, 2, …, p

$\mathrm{var}\left(y_j^o\right) = R_{jj}$ (2.27)

$\mathrm{var}\left(y_j^{(j)}\right) = H_j P^{(j)} H_j^T = \frac{1}{N-1}\sum_{k=1}^{N}\left(y'^{(j)}_{j,k}\right)^2$ (2.28)

$\mathrm{cov}\left(x^{(j)}, y_j^{(j)}\right) = P^{(j)} H_j^T = \frac{1}{N-1}\sum_{k=1}^{N}\left(x'^{(j)}_k \cdot y'^{(j)}_{j,k}\right)$ (2.29)

$K_j = \frac{\mathrm{cov}\left(x^{(j)}, y_j^{(j)}\right)}{\mathrm{var}\left(y_j^o\right)+\mathrm{var}\left(y_j^{(j)}\right)}$ (2.30)

$\overline{x}^{(j+1)} = \overline{x}^{(j)} + \rho_j \circ K_j\left(y_j^o - \overline{y}_j^{(j)}\right)$ (2.31)

for k = 1, 2, …, N

$x'^{(j+1)}_k = x'^{(j)}_k + \phi\rho_j \circ K_j\left(-y'^{(j)}_{j,k}\right)$ (2.32)

end

for l = 1, 2, …, p

$\mathrm{cov}\left(y_l^{(j)}, y_j^{(j)}\right) = H_l P^{(j)} H_j^T = \frac{1}{N-1}\sum_{k=1}^{N}\left(y'^{(j)}_{l,k} \cdot y'^{(j)}_{j,k}\right)$ (2.33)

$H_l K_j = \frac{\mathrm{cov}\left(y_l^{(j)}, y_j^{(j)}\right)}{\mathrm{var}\left(y_j^o\right)+\mathrm{var}\left(y_j^{(j)}\right)}$ (2.34)

$\overline{y}_l^{(j+1)} = \overline{y}_l^{(j)} + \rho_{lj} H_l K_j\left(y_j^o - \overline{y}_j^{(j)}\right)$ (2.35)

for k = 1, 2, …, N

$y'^{(j+1)}_{l,k} = y'^{(j)}_{l,k} + \phi\rho_{lj} H_l K_j\left(-y'^{(j)}_{j,k}\right)$ (2.36)

end

end

end

$x_k^a = \overline{x}^{(p+1)} + x'^{(p+1)}_k$, for k = 1, 2, ..., N

2.5 Parallelization of the EnKF algorithm

Here we outline the basic parallelization strategy employed in the PSU system. **Table 3.1** provides a list of variable names used in pseudo code (2.22) – (2.36) and in the actual Fortran code. For more details of the Fortran code, please refer to the next section.

The model state x is typically considered an $n \times 1$ vector in pseudo code formulation. However, when applying the algorithm to a real atmospheric model, it is necessary to work with the original model dimensions (zonal, meridional, vertical directions, and number of variables) instead of squeezing them into a long one-dimensional vector. The WRF model state space is defined as `x(ix+1,jx+1,kx+1,nv,nens+1)`, where `ix`, `jx`, and `kx` are the sizes of the model grid in zonal, meridional, and vertical directions, respectively. The staggered model grid requires one additional grid point in each direction to accommodate all variables. `nv` is the number of model variables. `nens` is the size of ensemble. The `1:nens` locations store the ensemble perturbations x'_k in (2.23), and the `nens+1` location stores the ensemble mean $\overline{x}$ in (2.22). To parallelize the EnKF algorithm, we decompose the work load in the `1:nens+1` dimension, as well as the model grid into slabs in the horizontal directions `1:ix` and `1:jx`. Such decomposition will reduce both the computation and storage load on each processor.

The Message Passing Interface (MPI) library is utilized to perform parallel computation. After initialization, the MPI provides each processor a unique rank from `0` to `nprocs-1`, called `proc_id`, where `nprocs` is the total number of processors. `comm` is a communicator that coordinates inter-processor data transfer among these processors. Let `nicpu`, `njcpu`, and `nmcpu` be the number of processors assigned to collectively store the data in `1:ix`, `1:jx`, and `1:nens+1` dimensions, respectively. They shall satisfy

```
nicpu*njcpu*nmcpu=nprocs
```

The global communicator `comm` is then split into `s_comm` that handles inter-processor operations among different slabs of the domain, and `g_comm` that handles operations among different members. The processors are also divided into subgroups according the particular domain slab and members they store as follows.

```
gid=int(proc_id/(nicpu*njcpu))
sid=mod(proc_id,nicpu*njcpu)
call MPI_Comm_split(comm,gid,sid,s_comm,ierr)
call MPI_Comm_split(comm,sid,gid,g_comm,ierr)
```

The processors that store the same domain slab will share the same `sid`, and those that store the same members will share the same `gid`. The data structure of model state x stored on each processor becomes `x(ni,nj,nk,nv,nm)`, where

```
ni=int((ix+1)/nicpu)+1
nj=int((jx+1)/njcpu)+1
nk=kx+1
nm=int((nens+1)/nmcpu)+1
```

And the start and end indices for each domain slab can be calculated as follows.

```
iid=mod(sid,nicpu)
jid=int(sid/nicpu)
```

```
istart=iid*ni+1
iend=(iid+1)*ni
jstart=jid*nj+1
jend=(jid+1)*nj
if(iid==(nicpu-1)) iend=ix+1
if(jid==(njcpu-1)) jend=jx+1
```

A schematic of decomposing along one dimension is shown in **Fig. 2.2** for the case of 3 processors. Note that the last processor will store fewer grid points if the total number of grid points cannot be exactly divided by the number of processors. For better parallel efficiency, one should design `nicpu`, `njcpu`, and `nmcpu` such that the last slab is as large as the previous ones as possible.

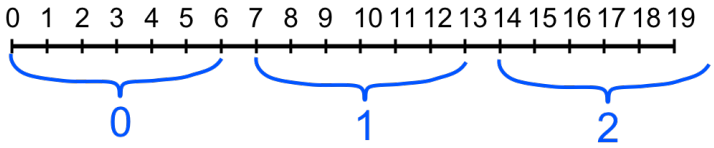

Fig. 2.2. A schematic of decomposing a mesh grid of size 20 to be stored on 3 processors. Each processor will store 7 grid points, except that the third processor (`id=2`) will store only 6 grid points.

Some inter-processor communication is required to complete the calculation along the ensemble members dimension. For example, the calculation of ensemble mean in (2.22) is done as follows

```
Call MPI_Allreduce(sum(x,5), xm, ni*nj*nk*nv, &
                   MPI_REAL, MPI_SUM, g_comm, ierr)
xm=xm/real(nens)
```

For processors with the same `sid`, the members stored locally on each processor is first summed as `sum(x,5)`, then summed together using `MPI_Allreduce` with `MPI_SUM` operation. The end result `xm` is the sum of all members, which is divided by `nens` to get ensemble mean. Similarly, the ensemble-estimated error variances and covariances in (2.28) and (2.29) can be calculated using `MPI_Allreduce` among the `g_comm` processors as well.

During the calculation of Kalman gain in (2.30), processors with different `sid` need to communicate to gather all required information. **Fig. 2.3** shows an example of domain decomposition when `nicpu` = 2 and `njcpu` = 2. The model domain is decomposed into 4 slabs with `sid` = 0, 1, 2, and 3. The blue box shows the update zone associated with an observation located in slab 3. The update zone is centered at the observation and its size is defined by the ROI. To speed up the calculation of Kalman gain, the update zone is also decomposed into 4 slabs for each `sid` to compute simultaneously. In this example, the update zone slab 0 would require small pieces from all 4 domain slabs to finish calculation; update zone slab 2 would require domain slabs 2 and 3; update zone slab 1 would require domain slabs 1 and 3; and update zone slab 3 is the only one that does not require any inter-processor communication.

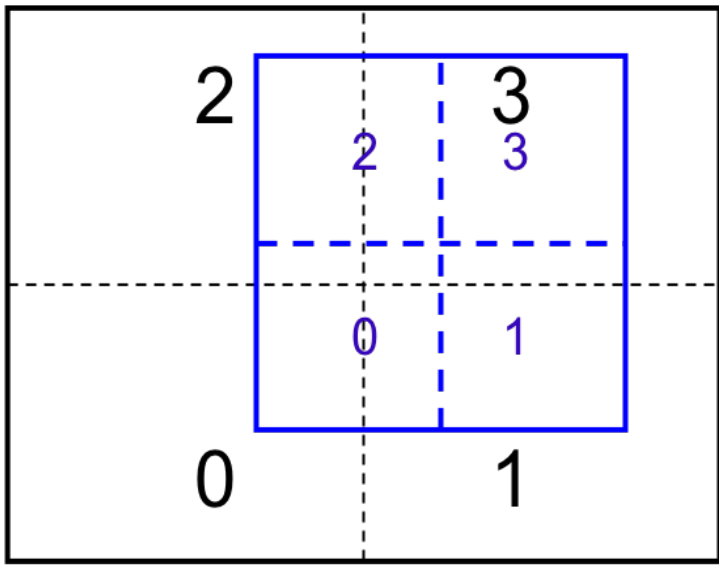

Fig. 2.3. A sample domain decomposing schematic for the case of `nicpu=2` and `njcpu=2`. Numbers indicate the `sid` of a processor. Black box indicates the domain slabs stored on each processor, and the blue box indicates the Kalman gain centered at the observation location with size 2*ROI+1 in both directions, which is also decomposed into 4 slabs.

Sending and receiving using `MPI_Send` and `MPI_Recv` need to be choreographed to prevent deadlocks, which are caused by a pair of processors trying to receive from each other and cannot proceed to the code that actually send out the message. **Table 2.3** shows the sending and receiving sequence that will avoid such deadlocks. A pseudo code shared by all `sid` to perform this sequence is as follows.

```
for is = 0,nicpu*njcpu-1
        find slab indices
        if (sid>is)
                MPI_Recv slab from processor is
        end
end
for i = 1,nicpu*njcpu
        is = mod(sid+i, nicpu*njcpu)
        find slab indices
        if (sid==i)
                get slab locally
        else
                MPI_Send slab to processor is
        end
end
```

```
for is = 0,nicpu*njcpu-1
        find slab indices
        if (sid<is)
                MPI_Recv slab from processor is
        end
end
```

The same sequence will take place again when the calculation of analysis increment in (2.31) and (2.32) is complete and needs to be written back to the `x` slabs.

Table 2.3. A sample sending (=>) and receiving (<=) sequence among 4 processors in `s_comm` during the gathering of domain slabs. From top to bottom is what happens as a result of running the pseudo code for each processor `sid`.

Processor 0	**Processor 1**	**Processor 2**	**Processor 3**
	<= 0	<= 0	<= 0
=> 1		<= 1	<= 1
=> 2	=> 2		<= 2
=> 3	=> 3	=> 3	
	=> 0	=> 0	=> 0
<= 1		=> 1	=> 1
<= 2	<= 2		=> 2
<= 3	<= 3	<= 3	

Note that the serial EnKF algorithm has an iterative assimilation loop over the number of observations (2.27) – (2.36). The iterative algorithm does not allow the assimilation of nearby observations to be parallelized, because the assimilation of the *j*-th observation would require the state $x^{(j-1)}$ which is a result from assimilating the previous (*j*-1)-th observation. The current parallelized algorithm scales well when number of observations is relatively small and ROIs are large. When a large amount of observations with relatively small ROIs are used, the current parallelization strategy is not optimal. However, one can potential improve the efficiency by decomposing the observation grid into slabs whose update zones do not overlap and thus can be assimilated simultaneously (such as the algorithm in Wang et al. 2013).

2.6 Prior and posterior innovation statistics

The observation space innovation statistics (Desroziers et al. 2005) is a useful tool to provide a consistency check for the filter. From (2.25), a copy of observation prior mean, $\overline{y}^b = \overline{y}^{(1)}$ can be saved before assimilation. After the observation loop is complete, the observations y^o and the observation posterior mean $\overline{y}^a = \overline{y}^{(p+1)}$ from (2.35) are also saved for the calculation of innovation statistics.

Consider innovation vectors $d^{o-b} = y^o - \overline{y}^b$, $d^{o-a} = y^o - \overline{y}^a$, and $d^{a-b} = \overline{y}^a - \overline{y}^b$ collected during a data assimilation cycle. The innovation statistics are written as follows.

$$\mathbb{E}[d^{o-b}(d^{o-b})^T] = HP^bH^T + R. \tag{2.37}$$

$$\mathbb{E}[d^{a-b}(d^{o-b})^T] = HP^bH^T. \tag{2.38}$$

$$\mathbb{E}[d^{o-a}(d^{o-b})^T] = R. \tag{2.39}$$

$$\mathbb{E}[d^{a-b}(d^{o-a})^T] = HP^aH^T. \tag{2.40}$$

One can calculate a consistency ratio,

$$\mathrm{CR} = \sqrt{\frac{\mathrm{tr}\left(\mathbb{E}\left[d^{o-b}\left(d^{o-b}\right)^T\right]\right) - \mathrm{tr}(R)}{\mathrm{tr}\left(HP^bH^T\right)}}, \tag{2.41}$$

to check the consistency of prior ensemble spread. If CR is larger than 1, the ensemble is under-dispersive and requires some inflation. Similar consistency ratios can also be defined for posterior ensemble spread as well as for the observation errors. Several studies have documented adaptive inflation methods that estimate the optimal amount of inflation utilizing the innovation statistics (Wang and Bishop 2003; Anderson 2007, 2009; Li et al. 2009; Miyoshi 2011; Ying and Zhang 2015).

2.7 Treatment of non-negative state variables

In EnKF, the background errors associated with state variables are assumed to be Gaussian random processes (white noise). While this assumption holds for some model variables, it is problematic to update non-negative state variables those errors do not follow a Gaussian distribution. For example, when updating the hydrometeor mixing ratio Q, say most of the members have $Q = 0$, and only a few members have $Q > 0$. The prior ensemble mean is apparently $\overline{Q} > 0$. When the observation is $Q = 0$, the ensemble mean will be reduced during data assimilation, and the members with $Q = 0$ will have to adjust toward negative values as a result of shifting of the ensemble mean. We apply an ad hoc adjustment to the posterior ensemble to correct the negative values. First, the negative members are set back to zero. Then, the positive members are shifted with an even amount so that the ensemble mean is unchanged. **Fig. 2.4** shows a schematic of this process.

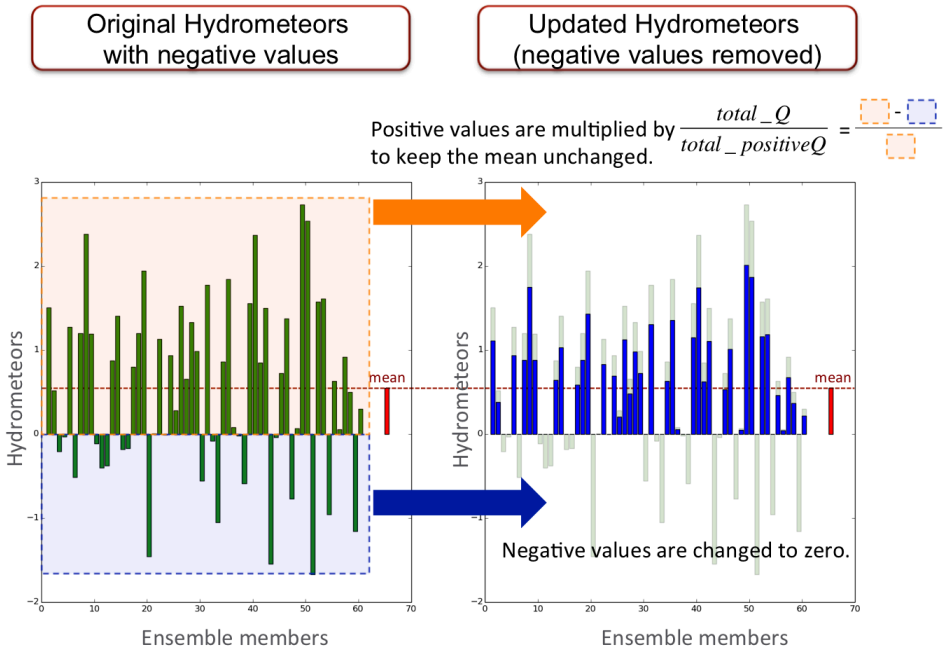

Fig. 2.4. Schematics showing adjustment of negative posterior state hydrometeors back to zero while preserving the posterior ensemble mean.

3 System workflow and components

The PSU system can perform hybrid ensemble and variational data assimilation with a combination of the EnKF and WRFDA code packages. **Fig. 3.1** shows the schematics of data assimilation workflows that the PSU system is capable of. Control scripts written in `bash` helps coordinate different modules and monitor job running status.

3.1 Code and working directories

Table 3.1. Contents of the PSU system code directory `$SCRIPT_DIR`.

config/	Stores the configuration files.
Irma_EnKF	A sample configuration file for the Irma (2017) test case using EnKF to assimilate conventional and satellite observations.
EnKF/src/	EnKF source code directory. See section 3.4 for more details of each code component.
WRF_BC_v2.1_alltime/src	Code package to update and/or perturb lateral boundary condition at the correct time slot in the `wrfbdy` file.
gen_be	A collection of control scripts to run `gen_be` code from the WRFDA package, in preparation for perturbing IC with the CV5 option.
module_*.sh	Control scripts to run a certain module in the system.
run_*.sh	Top-level scripts to run experiments.
util*	Utility scripts and functions.
namelist_*.sh	Scripts that generate `namelist` files for a certain component according to the input variables.
…	See section 3.2 for more details about other miscellaneous scripts and tools.

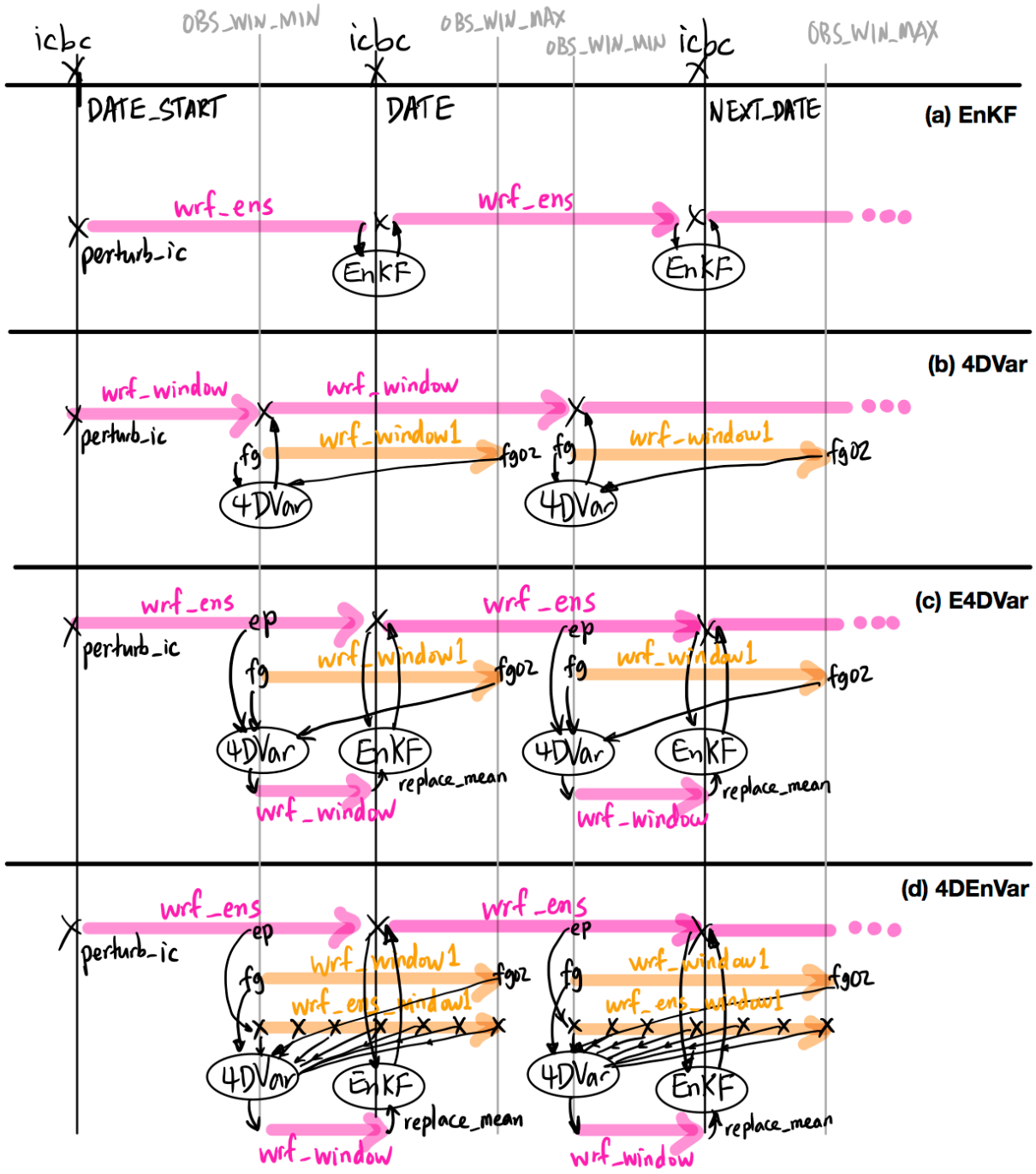

Fig. 3.1. PSU System workflows for (a) EnKF, (b) 4DVar, (c) E4DVar, and (d) 4DEnVar. The first spin up cycle from `DATE_START` to `DATE` and two data assimilation cycles at `DATE` and `NEXT_DATE` are shown. `OBS_WIN_MIN` and `OBS_WIN_MAX` are the time offsets for the window in which observations are assimilated for a cycle. Color marker arrows are associated with WRF forecast steps, and black thin arrows indicate file I/O for a certain module.

Table 3.2. Contents of the runtime working directory `$WORK_DIR`.

rc/	Output from real.exe at each cycle.
|-------- **$DATE/**	
|	
|---------- **wrfinput_$dm**	IC and LBC for domain `$dm`
|---------- **wrfbdy_$dm**	
|	
fc/	Input/output from the data assimilation steps at each cycle.
|-------- **$DATE/**	
|	
|---------- **wrfinput_$dm_$id**	IC files valid at `$DATE` for each member `$id` (the analyses)
|	
|	
|---------- **wrfinput_$dm_<$NEXTDATE>_$id**	IC files valid at `$NEXTDATE` for member `$id` (the forecast, prior ensemble for next cycle).
|	
|	
|-------- **wrfbdy_$dm_$id**	BC files if they are perturbed for each member `$id`
|-------- **wrflowinp_$dm_$id**	
|	
|-------- **wrfbdy_$dm**	BC files if they are shared by the ensemble.
|-------- **wrflowinp_$dm**	
|	
run/	
|-------- **$DATE/**	
|	Runtime directory for module at each cycle
|---------- **<module>/**	
|	
|---------- **stat**	Contains status of the module, "running/complete/error"
|	
|	
|---------- **run_<module>.sh**	Job submission script.[1]
|	
|---------- **…**	Other runtime files.
|	
output/	
|-------- **$DATE/**	
|---------- **wrfout_$dm_<$date>_$id**	Output from WRF forecast steps for member `$id` and at forecast time `$date`.

[1]A separate submission script for each module is generated if `$JOB_SUBMIT_MODE == 2`.

Table 3.3. A list of the external packages required by the PSU system.

WRF, WPS	WRF forecast model and the preprocessing WPS package.
WRFDA	WRF data assimilation package. The 3DVar with RANDOMCV mode is required by `module_perturb_ic` to generate initial ensemble for EnKF; 3/4Dvar is required when running hybrid data assimilation. Note that 4DVar will require WRFPLUS and WRFDA compiled with 4DVar options.
MULTI_INC	Some Fortran programs written for manipulating WRF output files (decimation, interpolation, etc.) during the incremental 4DVar step.
NETCDF	Software package to handle NETCDF file I/O.
MPI	Message Passing Interface software package that provides parallel computation subroutines. Compatible packages include mpich2, mvapich2, and intel impi.
CRTM	Community Radiative Transfer Model library, required by EnKF and WRFDA when assimilating satellite radiance observations.

3.2 Control scripts

run_cycle.sh

Top-level run script for cycling data assimilation.

The batch queue system headers, such as `#PBS -l walltime=1:00:00`, are required if submitting this script to the queue as one job (`$JOB_SUBMIT_MODE==1`). See the instructions from the HPC system for how to write these headers.

`cd $WORK/PSU_WRF_EnKF` changes directory to the `$SCRIPT_DIR`. Then, the configuration file `$CONFIG_FILE` is defined and loaded. `$total_ntasks` is the total number of processors allocated for this batch job.

What follows is the time loop for performing cycling data assimilation. The experiment period is defined as `$DATE_START` to `$DATE_END`. `$DATE` denotes the time of the current cycle, while `$PREVDATE` and `$NEXTDATE` denote the time of previous and next cycle, respectively. The first cycle is from `$DATE_START` to `$DATE_CYCLE_START`, which is typically longer to allow the initial ensemble to spin up. The cycling data assimilation will take place between `$DATE_CYCLE_START` and `$DATE_CYCLE_END`. The time interval between cycles is `$CYCLE_PERIOD`, and the lateral boundary condition is available every `$LBC_INTERVAL`

minutes. Observations are available within the window from `$OBS_WIN_MIN` to `$OBS_WIN_MAX`. Typically, `$DATE+$OBS_WIN_MAX` will equal `$NEXTDATE+$OBS_WIN_MIN`.

At each cycle, the control script for each module is called and run in the background (with `&`). In the control script of each module, the completion status is obtained from a `stat` file in its runtime directory. If its status is "completed", the current module will be skipped. Otherwise, it will check dependency, wait for prerequisite modules to complete, and then run the following scripts. These `stat` files allow easy restart of an experiment if a batch job doesn't complete successfully, just resubmit `run_cycle.sh` and it will scan over the completed modules and pick up where it left off.

module_icbc.sh

Prepares the initial and boundary conditions by running `geogrid.exe`, `ungrib.exe`, `metgrid.exe`, and `real.exe` from the WPS code package.

Typically, at `$DATE_START`, the analyses from global data assimilation system (e.g., from the GFS) are used as input files to create IC and BC files. However, it is sometimes not necessary to run `module_icbc.sh` for every cycle, for example, when the BCs are not perturbed for the ensemble members and a single `wrfbdy` file that contain all the time steps is stored in `fc/` to be shared among all members. For realtime application, `module_icbc.sh` is typically run every cycle as the global analyses become available.

module_perturb_ic.sh

Perturbs the initial condition to generate the initial ensemble.

The initial ensemble is generated by running 3DVAR in RANDOMCV mode. One can either use the CV3 or the CV5 background error covariance options (NMC method; Barker et al. 2004). If CV3 option is chosen, the `be.dat` calculated from global model climatology will be used. If CV5 option is chosen, one needs to prepare the `be.dat` file offline using the `gen_be` package. The perturbed variables include horizontal wind components, potential temperature, geopotential, and mixing ratio for water vapor, and their error statistics are defined by the climatological background error covariance. Other prognostic variables such as vertical velocity and hydrometeor mixing ratios are not perturbed.

If the boundary condition `wrfbdy` will be perturbed as well for the ensemble, a total of 100 perturbations will be generated for IC, and random draws of these perturbations will later be used by `update_wrf_bc.exe` to perturb the BC in a consistent fashion.

module_obsproc.sh

Prepares observation data files for EnKF using `obsproc.exe` from WRFDA.

The `obsproc.exe` program takes input data in LITTLE_R format. Currently, the LITTLE_R formatted GTS/MADIS observations prepared by NCAR are included. BUFR formatted data can also be included when their decoders are installed. For special data sources such as from field campaigns (PREDICT, DYNAMO, etc.), one can convert their data into LITTLE_R

format, and concatenate the text data file with other data sources all together in the `obs.raw` file, which is the input for `obsproc.exe`. Observations are logged in files every `$obs_interval` hours. To prevent occasional missing of observations that are logged with time lag (files with slightly later time stamp), a wider observation time window is applied when gathering the files to form `obs.raw`.

The output observation file is called `obs_gts_<time string>.3DVAR`. These files will later be linked in EnKF and 4DVar modules for assimilation.

run_obsproc.sh

Top-level control script that runs all the observation pre-processing jobs.

module_enkf.sh

Runs the EnKF module.

Prior ensemble members are linked as `fort.$((80010+mid))`, where `mid` is the member id from 1, 2, …, to `nens`. The prior ensemble mean will be output as `fort.$((80010+nens+1))`. Similarly, the posterior ensemble members will be output as `fort.$((90010+mid))`, and the posterior ensemble mean (the analysis) will be output as `fort.$((90010+nens+1))`.

GTS observations prepared by obsproc.exe will be linked as `obs_3dvar_<time string>`. The format for time string is "ccyymmddHHMMSS". Radar and radiance observation files are linked separately, see Appendix B and Section 4 for more details.

Runtime diagnostic output is written to `enkf.log`. Information of assimilated and rejected observations are written to `fort.10000` and `fort.10001`, respectively.

namelist_*.sh

Scripts that generate `namelist` files for a certain module. The variables are defined using `bash export` command before call these scripts, and the output is write to a `namelist` file that reflect the variables defined.

module_wrf_ens.sh

Runs ensemble WRF forecast across the cycle period (from `$DATE` to `$NEXTDATE`).

Before running the ensemble forecasts, the lateral boundary conditions (LBC) are updated and/or perturbed, because after data assimilation the values along the boundary may have changed and thus become inconsistent with the current boundary tendencies. For ensemble forecasts, the LBC for each ensemble member needs to be perturbed to allow uncertainties to enter the domain from boundaries, which will help prevent the LBC to over-constrain the solution and maintain ensemble spread.

There are two approaches to update a BC file:

1). Using `da_update_bc.exe` from the WRFDA package. This requires LBC files `wrfbdy` to be prepared separately for each cycle, since `da_update_bc.exe` only udpates the first time step in `wrfbdy` files.

2). Using `update_wrf_bc.exe` from the WRFBC_v2.1_alltime package. This is the default option. LBC files can be created at `DATE_START`, and each member maintain a copy of LBC file as `fc/wrfbdy_$dm_$mid`, which contains all the necessary time steps for this experiment. A `param.in` configuration file will be generated automatically when running `update_wrf_bc.exe` with the following contents.

```
&control_param
 wrf_3dvar_output_file = 'wrfinput_d01_update'
 wrf_bdy_file          = 'wrfbdy_d01_update'
 wrf_bdy_file_real     = 'wrfbdy_d01_real'
 wrf_input_from_si     = 'wrfinput_d01_real'
 wrf_input_from_si_randmean = 'random_mean'
 wrf_3dvar_random_draw = 'random_draw'
 cycling = .true.
 low_bdy_only = .false.
 perturb_bdy = .true.
 n_1 = 1
/
```

`wrfinput_d01_real` and `wrfbdy_d01_real` are the original output from real.exe at the current cycle, which serve as baseline unperturbed values. `wrfinput_d01_update` and `wrfbdy_d01_update` are the perturbed/updated files. `wrfinput_d01_update` is typically linked from the analysis at this cycle from data assimilation. `random_mean` and `random_draw` defines the perturbation, which is drawn from the pool of 100 perturbed ICs generated at `DATE_START`. `n_1` is the number of time step in wrfbdy that the program will update/perturb, which is calculated automatically by the script.

module_wrf.sh

Runs deterministic forecast from analysis. This module can either be called during cycling data assimilation (in `run_cycle.sh`), or after all cycling is complete (call `run_forecast.sh` after `run_cycle.sh` finishes).

run_forecast.sh

Top-level control script that runs all the deterministic forecasts.

util.sh

A collection of utility functions used by all control scripts.

`advance_time $DATE $inc`

Handles time calculation. It advances a time string `$DATE` forward `$inc` minutes. Time string is formatted as “ccyymmddHHMM”.

`wrf_time_string $DATE`

Converts `$DATE` time string into format used in WRF input/output file names, "ccyy-mm-dd_HH:MM:SS".

`wait_for_module $rundir`

Scans the `stat` files in the runtime directory of a module, and `sleep` until it's content becomes "complete".

`watch_log $logfile $keyword $timeout $rundir`

Scans the runtime `$logfile` of a module, and `sleep` until the log file contains a `$keyword` indicating success in completion (e.g. "SUCCESS" for WRF runs). If the function has slept for longer than `$timeout` minutes, the function will return an error message and exit.

`watch_file $filename $timeout $rundir`

Sleeps until the `$filename` output file exists. If the function has slept for longer than `$timeout` minutes, the function will return an error message and exit.

job_submit.sh

Script that handles job submission and monitoring for a certain module. This script is where specific parameters and commands should be defined according to the batch scheduler system being used. Currently, we provide examples for the NOAA Jet, TACC Stampede2, and NCAR Cheyenne HPC systems.

There are two job submit modes. For `$JOB_SUBMIT_MODE==1`, the top-level `run_cycle.sh` itself is submitted to the batch queue with allocation of a relatively large amount of resources, and each module is executed using `mpiexec` (or something similar) directly. This mode is suitable for HPC systems that are crowded and have long queuing time. Submitting the whole experiment as one batch job will reduce the amount of waiting in the queue, but the allocated resources should accommodate the most demanding module (usually EnKF and 4DVar).

For `$JOB_SUBMIT_MODE==2`, the top-level `run_cycle.sh` is executed directly in the command line. A separate run script is created by job_submit.sh for each individual module and submitted separately to the queue. The control script sleeps while the job is still waiting/running in the queue. This mode will have better cost efficiency since the resources are spent in an on-demand fashion, but this is not suitable for a slow queue.

jstat

Running "`$SCRIPT_DIR/jstat $WORK_DIR`" in the command line will provide a formatted view of the current job status.

multi_physics*.sh

Scripts that generates WRF model physics options that are randomly specified for each member. This is useful in creating a mulit-physics ensemble.

run_gen_be.sh and **gen_be/*ksh**

Prepares background error covariance file `be.dat` using `gen_be` from the package. An ensemble that samples the background error, either from 12- and 24-h forecast differences (NMC method) or from an ensemble with different model configuration.

calc_domain_moves.sh and **calc_ij_parent_start.sh**

Defines preset moves for storm-following nested domains. The storm motion is given by best track observations, such as the TC Vitals data from NHC.

3.3 Hybrid data assimilation components

module_4dvar.sh

Runs the 4DVar module. The WRFDA code `da_wrfvar.exe` will be called. Observations from `DATE+OBS_WIN_MIN` to `DATE+OBS_WIN_MAX` will be included as several time slots `ob1.ascii`, `ob2,ascii`, etc. Note that when preparing for these observations, the `obsproc.exe` should be running in 4DVar mode and generate the corresponding time slots.

module_*window.sh and **module_*window1.sh**

4DVar cycling (**Fig. 3.1b**) differs from EnKF in that the analysis time is valid at the beginning of the observation window instead of at the center of this window. The `module_wrf_window.sh` script handles the WRF forecast step that runs from `DATE+OBS_WIN_MIN` to `NEXTDATE+OBS_WIN_MIN`.

When running WRF in storm following mode, an extra run with domains fixed in space is necessary so that observation locations will be correctly calculated relative to the domains. This extra run will be handled by `module_wrf_window1.sh`, which runs the WRF model across the observation window to prepare a background trajectory with fixed domain.

The workflows for E4DVar and 4DEnVar are compared in **Fig. 3.1c** and **3.1d**. When coupling 4DVar to EnKF in E4DVar, the analysis from 4DVar at `DATE+OBS_WIN_MIN` is run forward to `DATE` using `module_wrf_window.sh`, so that the EnKF analysis ensemble can be recentered about this new analysis. For 4DEnVar, since the ensemble perturbations (ep) across the whole observation window are required to replace the functionality of tangent-linear and adjoint model, an ensemble forecast across the observation window is performed by `module_wrf_ens_window1.sh` with fixed domain. Note that `module_wrf_ens.sh` ensemble forecasts are run with storm-following domains.

For more details of the formulation of hybrid methods, please refer to Poterjoy and Zhang (2014a,b).

3.4 EnKF code components

Makefile

Configuration file for compiling the code using `make`.

main.f

Top-level source code for the EnKF parallel program `enkf.mpi`. It contains the following steps.

1. Initialize MPI environment: `parallel_start`.
2. Load configuration: `get_wrf_info` and `read_namelist.`
3. Read observation file: `get_all_obs`.
4. Figure out parallel strategy: calculate `nmcpu`, `nicpu`, `njcpu`, and related ids (`sid`, `gid`) and indices (`iid`, `jid`).
5. Allocate data structure `x` and `xm`, and read in the prior ensemble: `read_ensemble`.
6. Calculate and output prior ensemble mean using `MPI_Allreduce` and `output`.
7. Figure out observation assimilation sequence `ind(obs%num)`.
8. Run EnKF algorithm: `enkf`.
9. Output posterior ensemble and mean.
10. Finish and clean up.

obs_io.f

Subroutines for processing observations.

`get_all_obs`

Top-level observation processing subroutine, read in all observation files and calls sorting algorithms.

`get_gtsobs_3dvar`

Reads the `obs_3dvar` file output from `obsproc.exe`, and store data in `raw%gts`.

`get_wsr88d_radar`

Reads radar data files, and store data in `raw%radar`.

`get_airborne`

Reads airborne radar data files, and store data in `raw%airborne`.

`get_radiance`

Reads satellite radiance data file, and store date in `raw%radiance`.

sort_{sounding,upperair,surface,radarRV}_data

Scans `raw%gts`, and sort each observation type according to their platform number (FM).

Observation quality control and thinning are performed. Final observations are stored in `obs` data structure.

Note that `obs%type(iob)` is a 10-character string, which begins with 'S', 'P', or 'H' (surface, pressure level, or height level); followed by an `instrument` string that is one of the `obstype` listed in **Table B1**, and end with a character indicating the variable type ('U', 'V', 'T', 'Q', etc.). For special observation types, the string only contains its description, such as 'Radiance '.

`obs%position(iob,1:4)` stores the observation location in x, y, and z direction (model grid points), and on pressure level.

`obs%roi(iob,1:2)` stores the horizontal and vertical localization cutoff distances.

`obs%dat(iob)` stores the observation value.

xb.f

Subroutines `xb_to_*` that calculate corresponding observations values from the model state.

enkf.f

The main EnKF algorithm. Contains the following steps. See **Table 3.4** for a list of variable names in actual code and in pseudo code.

1. Calculate domain slab start and end indices: `istart`, `iend`, `jstart`, `jend`.

2. Calculate observation priors (Hx).

First, loop over all observations and figure out their location in the vertical, run `xb_to_*` subroutines in finding-k-location mode (`kkflag==0`), and store the location (in terms of model vertical level) in `obs%position(iob,3)`.

Then, loop over the observations again and calculate the corresponding prior values as converted from model prior ensemble states. For horizontal and vertical interpolation, gather a $3 \times 3 \times$ nk chunck of the prior state centered at the observation, `xob`, and convert to observation value `yasend`. Note that each processor in `s_comm` only compute part of the `ya` values in `yasend`, and at the end the `ya` values are gathered together.

3. A copy of `ya` is saved as `yf` (observation priors). A copy of `x` is saved as `xf` (prior ensemble).

4. Ensemble perturbations are calculated and multiplicative inflation is applied.

5. Start assimilation loop

   5.1. Kick off the observation if innovation `y_hxm` is 5 times larger than observation error.

   5.2. Calculate innovation variances: `d`, and the square root modification term: `alpha`.

   5.3. Update zone (also the Kalman gain `km`) of the observation is `ist:ied`, `jst:jed`, and `kst:ked`, defined by localization distance `ngx` and `ngz`.

5.4. Gather slabs in `x` to form the update zone slab via choreographed `MPI_Send` and `MPI_Recv`. The indices of the `km` slabs are `uist:uied,ujst:ujed`, and the indices of `x` slabs gathered to `sid` are calculated (`sistart`, `siend`, `sjstart`, and `sjend`).

5.5. Calculate Kalman gain `km`.

5.6. Scatter `km` slabs back to `sid`.

5.7. Update `x` and `xm`.

5.8. Loop over all `ya` and update those within the localization distance.

6. Run diagnostics for filter performance. Calculate consistency ratio.
7. Apply covariance relaxation.
8. Add the ensemble mean `xm` back to perturbations `x` to form the final analysis ensemble.

sub_enkf_util.f

A collection of utility functions shared by other EnKF code modules.

`read_namelist`

Reads `namelist.enkf` and initialize default values.

`read_ensemble`

Reads the prior ensemble files `fort.800??` and store then in local data structure `x(ni,nj,nk,nv,nm)`, each processor stores part of the `ni`, `nj`, and `nm` dimensions.

`output`

Output the posterior ensemble and write to `fort.900??` files.

`wrf_var_dimension`

Handles staggered WRF grid for different variables.

`gaussdev`

Generate random draw from Guassian distribution.

`quicksort`

Sorts an array using Quick Sort algorithm.

cal_roi.f

Calculates localization factor `comp_cov_factor` using the Gaspari and Cohn (1999) polynomial. `ngx` and `ngz` are the cutoff distances (ROI) as number of grid points in horizontal and vertical directions, respectively. Subroutine `cal_hroi` specifies different ROIs to batches of observations as in SCL method (Weng and Zhang 2012).

module_structure.f

Defines data structure, constants, and namelist entries.

module_wrf_tools.f and **module_map_utils.f**

A collection of utility functions from the WRF package.

module_netcdf.f

Subroutines that handle the I/O of NETCDF formatted files.

mpi_module.f

Provides interface with MPI subroutines for parallelization. The initialized MPI environment provides each processor an id `proc_id`, and a communicator `comm` that coordinates inter-processor communication.

module_radar.f

Subroutines that processes radar observations.

hurricane_center.f

Subroutines that handles hurricane position and intensity (HPI) observations.

ensemble_mean.f

Source code for `ensemble_mean.exe` that calculates ensemble mean.

replace_mean*.f

Source code for `replace_mean.exe` that recenters the ensemble at a given mean file `fort.70010`.

Table 3.4. A list of variable names in pseudo code (**Table 2.2**) and in the actual Fortran code along with their descriptions.

Pseudo code	**Actual code**	**Description**
N	`nens`	Ensemble size
k	`ie`	Index for ensemble members
p	`obs%num`	Number of observations
j	`iob`	Index for observations
n	`(ix+1)*(jx+1)*(kx+1)*nv`	Number of state variables

$x_k^{\prime(j)}$	`x`	Prior ensemble perturbations
$\overline{x}^{(j)}$	`xm`	Prior ensemble mean
y_k^b	`yf`	Observation prior ensemble.
$y_k^{(j)} = \overline{y}_j^{(j)} + y_{j,k}^{\prime(j)}$	`ya`	Observation posterior ensemble.
y^o	`y`	Observation value
$y_j^o - \overline{y}_j^{(j)}$	`y_hxm`	Innovation
$y_{j,k}^{\prime(j)}$	`hxa`	Observation prior ensemble perturbations
$\mathrm{var}(y^o)$	`error**2`	Error variance of observation
$\mathrm{var}\left(y_j^{(j)}\right)$	`fac*var`	Background error variance of observation priors
$\mathrm{var}(y_j^o) + \mathrm{var}\left(y_j^{(j)}\right)$	`d`	Innovation variance
K_j	`km*fac/d`	Kalman gain
ϕ	`alpha`	Square root modification term
$\mathrm{cov}\left(y_l^{(j)}, y_j^{(j)}\right)$	`fac*cov`	Covariance between observation *l* and *j*.
ρ_j	`corr_coef`	Localization function
HROI	`ngx`	Horizontal localization cutoff distance (grid points)
VROI	`ngz`	Vertical localization cutoff distance (levels)

4 Observation data file formats

4.1 Conventional observations

GTS data prepared routinely by NCAR can be found recorded in LITTLE_R text data format.

The following is a typical sounding record:

```
            13.48000             2.1600061052                                NIAMEY-
AERO / NIGER                       FM-35 TEMP                               GTS (ROHK)
USNR20 DRRN 302300                      227.00000         1   -888888   -888888       374
-888888         T         F         F   -888888   -888888      20110930230000-888888.00000
0-888888.00000        0-888888.00000        0-888888.00000        0-888888.00000        0-
888888.00000         0-888888.00000         0-888888.00000         0-888888.00000         0-
888888.00000       0-888888.00000       0-888888.00000       0-888888.00000       0


  98600.00000        0     227.00000        0     305.35001        0     291.35001        0
2.05778       0    280.00000       0-888888.00000       0-888888.00000       0-888888.00000
0-888888.00000       0


  92500.00000        0     790.00000        0     303.14999        0     282.14999        0-
888888.00000         0-888888.00000         0-888888.00000         0-888888.00000         0-
888888.00000       0-888888.00000       0
...


  10000.00000        0   16550.00000        0     192.44998        0-888888.00000        0-
888888.00000         0-888888.00000         0-888888.00000         0-888888.00000         0-
888888.00000       0-888888.00000       0


-888888.00000         0-888888.00000         0-888888.00000         0-888888.00000         0-
888888.00000         0-888888.00000         0-888888.00000         0-888888.00000         0-
888888.00000       0-888888.00000       0


-777777.00000         0-777777.00000         0      13.00000         0-888888.00000         0-
888888.00000         0-888888.00000         0-888888.00000         0-888888.00000         0-
888888.00000       0-888888.00000       0


     38      0      0
```

LITTLE_R formatted data can be easily concatenated and processed with `obsproc.exe` to produce the final data input to EnKF and 4DVar. The `obsproc.exe` output data format, `obs_3dvar` file, contains a header that looks like the following.

```
TOTAL =   1153, MISS. =-888888.,
SYNOP =     24, METAR =      4, SHIP  =      9, BUOY  =      0, BOGUS =      0,
TEMP  =      8,
AMDAR =      4, AIREP =      0, TAMDAR=      0, PILOT =      0, SATEM =      0,
SATOB =   1104,
GPSPW =      0, GPSZD =      0, GPSRF =      0, GPSEP =      0, SSMT1 =      0,
SSMT2 =      0,
```

```
TOVS   =      0, QSCAT =      0, PROFL =      0, AIRSR =      0, OTHER =      0,
PHIC   =   0.00, XLONC =  75.00, TRUE1 =   0.00, TRUE2 = -30.00, XIM11 =   1.00,
XJM11 =   1.00,
base_temp= 290.00, base_lapse=  50.00, PTOP  =   100., base_pres=100000.,
base_tropo_pres= 20000., base_strat_temp=   215.,
IXC    =    223, JXC    =    334, IPROJ =      3, IDD    =      1, MAXNES=      1,
NESTIX=    223,
NESTJX=    334,
NUMC   =      1,
DIS    =   9.00,
NESTI  =      1,
NESTJ  =      1,
INFO   = PLATFORM, DATE, NAME, LEVELS, LATITUDE, LONGITUDE, ELEVATION, ID.
SRFC   = SLP, PW (DATA,QC,ERROR).
EACH   = PRES, SPEED, DIR, HEIGHT, TEMP, DEW PT, HUMID (DATA,QC,ERROR)*LEVELS.
INFO_FMT = (A12,1X,A19,1X,A40,1X,I6,3(F12.3,11X),6X,A40)
SRFC_FMT = (F12.3,I4,F7.2,F12.3,I4,F7.3)
EACH_FMT = (3(F12.3,I4,F7.2),11X,3(F12.3,I4,F7.2),11X,3(F12.3,I4,F7.2))
```

Followed by the actual records of observations. Each record contains three parts whose format are described by `INFO_FMT`, `SRFC_FMT`, and `EACH_FMT` in the header. A typical record looks like the following.

```
FM-35 TEMP   2011-10-19_00:00:00 UpperAir OBS from MADIS                  126
-7.300                 72.400                  0.000                 61967

 -888888.000 -88 200.00 -888888.000 -88  0.200

  100900.000    0 100.00         2.600    0    1.10       100.000    0    1.10
1.000     0     7.00        300.750     0     1.00        295.950     0     1.00
74.663    0   15.00

  100000.000    0 100.00         5.100    0    1.10       130.000    0    1.10
79.000     0     7.00        300.550     0     1.00        296.150     0     1.00
76.503    0   15.00

   99000.000    0 100.00        10.800    0    1.10       125.000    0    1.10
168.000     0     7.06        299.750     0     1.00        295.850     0     1.00
78.798    0   14.69

...
```

The INFO part contains, from left to right, the platform number, observation type, a time string, a description, number of vertical levels, latitude, longitude, elevation, station id.

Each observation contains three numbers (observed value `"%12.3f"`, quality control flag `"%4i"`, and observation error `"%7.2f"`). The SRFC part contains the surface observations: sea level pressure (slp) and precipitable water (pw). The EACH part contains the upper-air observations: pressure, wind speed, wind direction, height, temperature, dewpoint temperature, and relative humidity.

4.2 Radar observations

Land-based WSR-88D radar

Content of the `radar_data.info` file is:

```
station idn="000386" id="SPOL" name="Gan/SPOLS-bandradar" st="  " co="
" lat="-0.63" lon="73.10" elev="10"
observation attribute hgt="0" scan="8" elevation="0.5 1.5 2.5 3.5 5.0
7.0  9.0  11.0"  mindis="4.0"  maxdis="150."  ddis="4.0"  dazm="5.0"
rf_err="5.0" rv_err="3.0"
```

`station:`

`idn`: the radar station id number.

`id`: the 4-character station name.

`name`: a description of the station.

`lat`: the latitude of the station.

`lon`: the longitude of the station.

`elev`: the terrain elevation of the station.

`observation attribute:`

`hgt`: the height of the station.

`scan`: the number of scans.

`elevation`: a list of elevation angles for different scans.

`mindis`: the minimum range distance.

`maxdis`: the maximum range distance.

`ddis`: the increment of distance.

`dazm`: the increment of azimuthal angles.

`rf_err`: observation error for reflectivity.

`rv_err`: observation error for radial velocity.

The radar superob data file contains records with the following format:

```
        350.000       1.500       87.700       -1.795       12.110
```

From left to right, azimuthal angle, elevation angle, radial distance, radial velocity (rv), and reflectivity (rf) are recorded as `"%12.3f"`.

Airborne Doppler radar

The airborne Doppler radar data file contains records with the following format:

```
201110010000   10.000   -80.000    12000   350.000    1.500   87.700
-1.795    12.110
```

From left to right:

the time of observation point is a 12-character string

latitude, longitude, and height of the basepoint (radar location),

azimuthal angle, elevation angle, radial distance, radial velocity (rv), and reflectivity (rf) of the data point are recorded as `"%10.3f"`.

4.3 Satellite observations

The satellite brightness temperature data file contains records with the following format:

```
201110080100   mviriNOM_m07          3     -10.273      59.483     250.534
```

From left to right:

the time of observation point is a 12-character string,

the satellite instrument id is a 15-character string,

the channel number is recorded as a `"%12i"`,

the latitude, longitude, and brightness temperature are recorded as `"%12.3f"`.

5 Tunable parameters

To achieve the best performance of a data assimilation system, a tuning process is usually required. For EnKF, some ad hoc modification of the filter, such as localization and inflation, need to be tuned for a particular application scenario, considering the observation type, network density, and accuracy, and the particular spatial and temporal scales of the system of interest. Houtekamer and Zhang (2015) provide a comprehensive review of such consideration when applying an EnKF to study a variety of systems.

5.1 Covariance Inflation

The PSU system uses covariance relaxation, also known as the relax-to-prior-perturbation (RTPP) method proposed by Zhang et al. (2004), to inflate the background error covariance after each assimilation cycle. Unlike the standard inflation method (Anderson 2001), in which all points in the prior field are inflated, this relaxation method only inflates the covariance at grid points affected by the data assimilation via a weighted average between the prior perturbations and the posterior perturbations. The mixing coefficient α is set from 0.7 to 0.8 for imperfect-model experiments (Meng and Zhang 2007) or real-data applications (Whitaker et al. 2008; Meng and Zhang 2008a,b; Torn and Hakim 2008) due to un-avoidable imperfections in the forecast model and nonlinear error growth between assimilation cycles. For perfect-model application, the mixing coefficient α is typically tuned around 0.5. Adaptive methods that estimate this α parameter online with innovation statistics are also available (e.g. Ying and Zhang 2015; Kotsuki et al. 2016).

5.2 Localization

The localization cutoff distance, also known as the radius of influence (ROI), is another important tuning parameter. For a given observation network, studies showed that the optimal localization distance is dependent on the underlying dynamical correlation length scale, the observation density, as well as the ensemble size (e.g. Kirchgessner et al. 2014; Perianez et al. 2014; Ying et al. 2018). Typically, the ROI is set to ~5 times the observation intervals. As ensemble size decreases, the ROI needs to decrease as well. The smaller scale a system is, the smaller ROI needs to be as well. If the ROI is too large, too much sampling error will contaminate the analysis increments. If the ROI is too small, the available information from observations are not fully utilized and the solution can suffer from imbalances induced by the imposed length scale (Greybush et al. 2011).

For dense observing network, the PSU system has an additional option to utilize a successive covariance localization (SCL) technique (Zhang et al. 2009), which assimilates observations that contain information about the state of the atmosphere at a wide range of scales and is designed to reduce computational costs and sampling errors. This technique uses the Gaspari and Cohn (1999) fifth-order correlation function for covariance localization, but a different localization radius of influence (ROI) is used for different groups of observations by random sampling. SCL assumes that both large- and small-scale errors are simultaneously present. First, one tries to remove dynamically important aspects of the large-scale error by assimilating a relatively small subset of observations with a large ROI. Next, the ROI is made smaller, and higher-density observations are used to constrain both smaller-scale errors and what remains of the large-scale error. The process is repeated until all scales resolved by the observational network have been adequately dealt with.

5.3 Hybrid-related parameters

In hybrid data assimilation method (E4DVar and 4DEnVar), the mixing coefficient β control the relative contribution of climatological (static) and ensemble-estimated background error covariance.

$$P^{\text{hyb}} = (1-\beta)B + \beta\ \rho_L \circ P^b \tag{5.1}$$

The ensemble-estimated covariance has better flow dependency but is subject to sampling noises, while the climatological covariance is full-rank and therefore allows new directions to occur in the analysis increment, which effectively serves as an additive inflation that maintains the filter stability.

Typically, the mixing coefficient is set to 0.8. See Poterjoy and Zhang (2014a, b; 2015) for more analysis of how this parameter influence the performance of a hybrid method.

Appendix A: Proof of equivalence between simultaneous and serial algorithms

The simultaneous assimilation algorithm (2.3) – (2.6) can be written as

$$x^a = x^b + P^b H^T (H P^b H^T + R)^{-1} [y^o - h(x^b)], \tag{A1}$$

and

$$P^a = P^b - P^b H^T (H P^b H^T + R)^{-1} H P^b. \tag{A2}$$

We can simplify (A1) using the following assumption,

$$\begin{aligned} y^o - h(x^b) &\approx h(x^t) + H(x^o - x^t) - h(x^t) - H(x^b - x^t) \\ &= H(x^o - x^b). \end{aligned} \tag{A3}$$

And (A1) becomes

$$x^a = x^b + P^b H^T (H P^b H^T + R)^{-1} H (x^o - x^b). \tag{A4}$$

Since R is diagonal, there exists $S = \text{diag}\left(\sqrt{R_{11}}, \sqrt{R_{22}}, \ldots, \sqrt{R_{pp}}\right)$, so that $R = S^T S$. Let $\widetilde{H} = S^{-T} H$, we have

$$P^b H^T (H P^b H^T + R)^{-1} H \tag{A5}$$

$$= P^b H^T [S^T (S^{-T} H P^b H^T S^{-1} + I) S]^{-1} H \tag{A6}$$

$$= P^b H^T S^{-1} (S^{-T} H P^b H^T S^{-1} + I)^{-1} S^{-T} H \tag{A7}$$

$$= P^b \widetilde{H}^T \left(\widetilde{H} P^b \widetilde{H}^T + I\right)^{-1} \widetilde{H} \tag{A8}$$

Using matrix identity $A(I + BA)^{-1} = (I + AB)^{-1} A$, and let $Q = P^b \widetilde{H}^T \widetilde{H}$, we can further simplify (A8) as

$$= P^b \widetilde{H}^T \widetilde{H} \left(P^b \widetilde{H}^T \widetilde{H} + I\right)^{-1} \tag{A9}$$

$$= Q(Q + I)^{-1} \tag{A10}$$

Finally, (A1) and (A2) can be rewritten as

$$\begin{aligned} x^a &= x^b + Q(Q + I)^{-1} (x^o - x^b) \\ &= Q(Q + I)^{-1} x^o + (Q + I)^{-1} x^b, \end{aligned} \tag{A11}$$

and

$$\begin{aligned} P^a &= P^b - Q(Q + I)^{-1} P^b \\ &= (Q + I)^{-1} P^b. \end{aligned} \tag{A12}$$

Similarly, the update equations in the serial algorithm (2.7) – (2.17) can be written as

$$\begin{aligned} x^{(j+1)} &= x^{(j)} + P^{(j)} H_j^T \left(H_j P^{(j)} H_j^T + R_{jj}\right)^{-1} H_j \left(x^o - x^{(j)}\right) \\ &= P^{(j)} \widetilde{H}_j^T \widetilde{H}_j \left(P^{(j)} \widetilde{H}_j^T \widetilde{H}_j + I\right)^{-1} x^o + \left(P^{(j)} \widetilde{H}_j^T \widetilde{H}_j + I\right)^{-1} x^{(j)}, \end{aligned} \tag{A13}$$

and

$$P^{(j+1)} = P^{(j)} + P^{(j)}H_j^T\left(H_jP^{(j)}H_j^T + R_{jj}\right)^{-1}H_jP^{(j)}$$
$$= \left(P^{(j)}\tilde{H}_j^T\tilde{H}_j + I\right)^{-1}P^{(j)}. \quad \text{(A14)}$$

Here $\tilde{H}_j$ is the j-th row of $\tilde{H}$, so that $Q = P^b\tilde{H}^T\tilde{H} = \sum_{j=1}^{p} P^b\tilde{H}_j^T\tilde{H}_j$.

Now, we prove the equivalence between serial and simultaneous assimilation algorithms by performing a mathematical induction on the number of observations p.

If p = 1, the equivalence apparently holds.

Assume that the equivalence is true for the first p-1 observation. Namely, $x^{(p)}$ from the serial algorithm is the same as the analysis from the simultaneous algorithm assimilating the first p-1 observations. We will show that the assimilation of the final p-th observation that updates $x^{(p)}$ to $x^{(p+1)}$ will yield the same result as the analysis from the simultaneous algorithm x^a.

First, we rewrite $\tilde{H} = \begin{pmatrix}\tilde{H}_{1\to p-1}\\ \tilde{H}_p\end{pmatrix}$, where $\tilde{H}_{1\to p-1}$ contains the first p-1 rows of $\tilde{H}$ and $\tilde{H}_p$ is the p-th row. According to (A11) and (A12), we can write the analysis from simultaneous algorithm assimilating the first p-1 observation as

$$Q_{1\to p-1}\left(Q_{1\to p-1} + I\right)^{-1}x^o + \left(Q_{1\to p-1} + I\right)^{-1}x^b, \quad \text{(A15)}$$

where $Q_{1\to p-1} = P^b\tilde{H}_{1\to p-1}^T\tilde{H}_{1\to p-1}$.

Since $x^b = x^{(1)}$, $P^b = P^{(1)}$, and we assumed that this analysis is equivalent to $x^{(p)}$ from the serial algorithm, we have

$$x^{(p)} = Q_{1\to p-1}\left(Q_{1\to p-1} + I\right)^{-1}x^o + \left(Q_{1\to p-1} + I\right)^{-1}x^{(1)}, \quad \text{(A16)}$$

$$P^{(p)} = \left(Q_{1\to p-1} + I\right)^{-1}P^{(1)}, \quad \text{(A17)}$$

and

$$Q_{1\to p-1} = P^{(1)}\tilde{H}_{1\to p-1}^T\tilde{H}_{1\to p-1}. \quad \text{(A18)}$$

According to (A13), the assimilation of the p-th observation yields

$$x^{(p+1)} = P^{(p)}\tilde{H}_p^T\tilde{H}_p\left(P^{(p)}\tilde{H}_p^T\tilde{H}_p + I\right)^{-1}x^o + \left(P^{(p)}\tilde{H}_p^T\tilde{H}_p + I\right)^{-1}x^{(p)}. \quad \text{(A19)}$$

Using (A16) and (A17),

$$x^{(p+1)} = \left(Q_{1\to p-1} + I\right)^{-1}P^{(1)}\tilde{H}_p^T\tilde{H}_p\left[\left(Q_{1\to p-1} + I\right)^{-1}P^{(1)}\tilde{H}_p^T\tilde{H}_p + I\right]^{-1}x^o +$$
$$\left[\left(Q_{1\to p-1} + I\right)^{-1}P^{(1)}\tilde{H}_p^T\tilde{H}_p + I\right]^{-1} \times$$
$$\left[Q_{1\to p-1}\left(Q_{1\to p-1} + I\right)^{-1}x^o + \left(Q_{1\to p-1} + I\right)^{-1}x^{(1)}\right]. \quad \text{(A20)}$$

Let $Q_p = P^{(1)}\widetilde{H}_p^T\widetilde{H}_p$. We can show that

$$Q = P^{(1)}\widetilde{H}^T\widetilde{H} = P^{(1)}\widetilde{H}_{1\to p-1}^T\widetilde{H}_{1\to p-1} + P^{(1)}\widetilde{H}_p^T\widetilde{H}_p = Q_{1\to p-1} + Q_p, \qquad \text{(A21)}$$

and

$$\begin{aligned}
&\left(Q_{1\to p-1} + I\right)^{-1}Q_p + I \\
&= \left(Q_{1\to p-1} + I\right)^{-1}\left(Q_p + Q_{1\to p-1} + I\right) \\
&= \left(Q_{1\to p-1} + I\right)^{-1}(Q + I). \qquad \text{(A22)}
\end{aligned}$$

Using (A21) and (A22), (A20) becomes

$$\begin{aligned}
x^{(p+1)} = \left(Q_{1\to p-1} + I\right)^{-1}Q_p &\left[\left(Q_{1\to p-1} + I\right)^{-1}(Q + I)\right]^{-1}x^o + \\
&\left[\left(Q_{1\to p-1} + I\right)^{-1}(Q + I)\right]^{-1} \times \\
&\left[Q_{1\to p-1}\left(Q_{1\to p-1} + I\right)^{-1}x^o + \left(Q_{1\to p-1} + I\right)^{-1}x^{(1)}\right].
\end{aligned}$$

Note that Q and I are symmetric matrices,

$$\begin{aligned}
x^{(p+1)} &= \left[\left(Q_{1\to p-1} + I\right)^{-1}(Q + I)\right]^{-1}\left(Q_{1\to p-1} + I\right)^{-1}\left(Q_p + Q_{1\to p-1}\right)x^o + \\
&\quad \left[\left(Q_{1\to p-1} + I\right)^{-1}(Q + I)\right]^{-1}\left(Q_{1\to p-1} + I\right)^{-1}x^{(1)} \\
&= Q(Q + I)^{-1}x^o + (Q + I)^{-1}x^{(1)}, \qquad \text{(A23)}
\end{aligned}$$

which is equivalent to x^a as in (A11).

Similarly, we can show for covariance

$$\begin{aligned}
P^{(p+1)} &= \left(P^{(p)}\widetilde{H}_p^T\widetilde{H}_p + I\right)^{-1}P^{(p)} \\
&= \left[\left(Q_{1\to p-1} + I\right)^{-1}P^{(1)}\widetilde{H}_p^T\widetilde{H}_p + I\right]^{-1}\left(Q_{1\to p-1} + I\right)^{-1}P^{(1)} \\
&= \left[\left(Q_{1\to p-1} + I\right)^{-1}Q_p + I\right]^{-1}\left(Q_{1\to p-1} + I\right)^{-1}P^{(1)} \\
&= \left[\left(Q_{1\to p-1} + I\right)^{-1}(Q + I)\right]^{-1}\left(Q_{1\to p-1} + I\right)^{-1}P^{(1)} \\
&= (Q + I)^{-1}P^{(1)}, \qquad \text{(A24)}
\end{aligned}$$

which is equivalent to P^a as in (A12).

Q.E.D.

Appendix B: EnKF namelist entries

EnKF basic configuration, definition of update domain, and some tunable parameters (inflation, localization, etc.)

&enkf_parameter

nens = 10,	Ensemble size.
enkfvar = 'T ', 'W ', 'U ', 'V ', 'QVAPOR ', 'QCLOUD ', 'QRAIN ', 'PH ', 'MU ', 'PSFC ', 'P ', 'PHB ', 'PB ', 'MUB ', 'U10 ', 'V10 ',	State variables participating the calculation of Kalman gain. Variable names are 10-character strings. The number of variables is nv.
updatevar = 'T ', 'W ', 'U ', 'V ', 'QVAPOR ', 'QCLOUD ', 'QRAIN ', 'PH ', 'MU ', 'PSFC ', 'P ', 'U10 ', 'V10 ',	State variables actually being updated by the filter.
update_is = 2, **update_ie** = 301, **update_js** = 2, **update_je** = 301, **update_ks** = 1, **update_ke** = 42,	Start/end indices of region inside the model domain that will be updated by EnKF. s = start index, e = end index. i = zonal direction, j = meridional direction, k = vertical direction. Typically the whole domain is updated, the BC will be updated according to the updated values as well. If one does not want to change the BC (e.g. in a perfect-model experiment), then set this update region to skip the boundary buffer zone.
inflate = 1.0,	Multiplicative inflation factor applied before assimilating observations, λ as in (2.20).
relax_opt = 0,	Covariance relaxation methods: 0 = relax-to-prior-perturbation (Zhang et al. 2004) 1 = relax-to-prior-spread (Whitaker and Hamill 2012)

relax_adaptive = .false.,	If true, turn on adaptive algorithm to estimate relaxation coefficient online using innovation statistics (Ying and Zhang 2015).
mixing = 0.8,	Relaxation coefficient, α as in (2.21).
random_order = .false.,	If true, assimilate observations in random sequence.
print_detail = 1,	Debug level.

/

Parallel algorithm configuration. See section 2.5 for more details on the parallelization strategy and a description of the algorithm.

&parallel

manual_parallel = .true.,	If true, manually specify the decomposition of work load using nicpu, njcpu, and nmcpu. If false, the algorithm will try to figure out a parallel scheme automatically.
nmcpu = 16,	Number of processors used in nens+1 dimension (how many groups of members are there).
nicpu = 2,	Number of processors used in zonal direction when decomposing the domain.
njcpu = 2,	Number of processors used in meridional direction when decomposing the domain.

/

Assimilation switches and configuration for each observation type **<obstype>** from the GTS data records. See **Table B1** for a list of supported types. The data file that contains all routine observations is named obs_3dvar_"ccyymmddHHMMSS" (for more details, see **Section 4.1**).

&<obstype>_obs

use_<obstype> = .true.,	Whether to include this type of observation in EnKF assimilation.
datathin_<obstype> = 0,	Factor used to thin the observations, the larger the fewer observations are assimilated.
hroi_<obstype> = 10,	Localization radius of influence (ROI) in the horizontal direction, in units of number of model grid points.
vroi_<obstype> = 5,	Localization ROI in the vertical direction, in units of number of vertical levels.

/

Assimilation of hurricane position and intensity (HPI) observations from best-track data (e.g. from JTWC, or NHC TC Vitals). The best track data file is named `hurricane_best_track`.

&hurricane_PI

use_hurricane_PI = .false.,	Whether to include hurricane position and intensity (HPI) observations.
hroi_<obstype> = 10,	Localization radius of influence (ROI) in the horizontal direction, in units of number of model grid points.
vroi_<obstype> = 5,	Localization ROI in the vertical direction, in units of number of vertical levels.

/

Assimilation of land-based WSR-88D radar data, which requires a `radar_data.info` file that describes the radar site and a text file `"RadarID"_"ccyymmddHHMMSS"_so` for each site that contains the superobs (see **Section 4.2** for more details).

&radar_obs

use_radar_rf = .false.,	Whether to include radar reflectivity.
use_radar_rv = .true.,	Whether to include radar radial velocity.
datathin_radar = 0,	Factor used to thin the observations, the larger the fewer observations are assimilated.
hroi_radar = 10,	Localization radius of influence (ROI) in the horizontal direction, in units of number of model grid points.
vroi_radar = 5,	Localization ROI in the vertical direction, in units of number of vertical levels.

/

Assimilation of land-based radar data, which requires a `airborne_"ccyymmddHHMM` file that contains the superobs (see **Section 4.2** for more details).

&airborne_radar

use_airborne_rf = .false.,	Whether to include radar reflectivity.
use_airborne_rv = .true.,	Whether to include radar radial velocity.
datathin_airborne = 0,	Factor used to thin the observations, the larger the fewer observations are assimilated.
hroi_airborne = 10,	Localization radius of influence (ROI) in the horizontal direction, in units of number of model grid points.
vroi_airborne = 5,	Localization ROI in the vertical direction, in units of number of vertical levels.

/

Table B1. A list of supported GTS observation types.

obstype	**WMO FM (platform) number**	**Description (BUFRLIB class)**
surface	12, 14	Land synoptic reports (SYNOPTIC)
sounding	32, 33, 34, 35, 36, 37, 38	Upper air wind, temperature, and humidity profiles (RAOB, PIBAL, RECCO, DROPS)
profiler	132	Wind profiler reports
aircft	42, 96, 97	Aircraft reports (AIREP/PIREP, AMDAR, E-ADAS)
metar	15, 16	Surface land reports (METAR)
sfcshp	13, 18	Surface marine reports (SHIP, BUOY, C-MAN)
spssmi	125, 126	DMSP SSM/I retrieval products, reprocessed wind speed, TPW
atovs	131	TIROS operational vertical sounder retrieval products, temperature and humidity profiles.
satwnd	88	Satellite-derived wind reports Atmospheric Motion Vectors
seawind	281	Sea-surface wind speed retrievals from satellite scatterometer, such as ASCAT
gpspw	111	GPS retrieval of Precipitable Water

Appendix C: Configuration file entries

System workflow control parameters.

`RUN_ENKF=true`	If `true`, run EnKF component during cycling.
`RUN_4DVAR=false`	If `true`, run 4DVar component; If both `RUN_ENKF` and `RUN_4DVAR` are `true`, run hybrid data assimilation.
`RUN_ENVAR=false`	If `true`, run 4DEnVar instead of E4DVar.
`FORECAST_TO_END=false`	If `true`, run deterministic forecast (`module_wrf.sh`) from analysis at each data assimilation cycle towards `DATE_END`.
`MULTI_PHYS_ENS=false`	If `true`, use multi-physics ensemble.
`MULTI_INC=false`	If `true`, run incremental 4DVar.
`DECIMATION_FACTOR=3`	Factor of decimation of high-resolution model grid to create lower-resolution model state for incremental 4DVar.
`FOLLOW_STORM=false`	If `true`, use hurricane best track data (e.g. TC Vital) to define preset moves for nested domains so that inner domains follow the storm.
`STORM_ID="al112017"`	Hurricane ID as defined in TC Vital data.
`CLEAN=false`	Whether to clean up runtime directory to save disk space. If true, this will purge the redundant `wrfout` files after the WRF forecasts are successfully complete and output are saved to `fc/` and `output/`.

Experiment time control. Time unit is in minutes.

`EXP_NAME="IRMA_EnKF"`	Name of the cycling data assimilation experiment.
`DATE_START=201808300000`	Start time of the experiment, the initial ensemble is generated at this time.

DATE_END =201809130000	End time of the experiment. This is typically the end of simulation of the system of interest.
DATE_CYCLE_START=201808310000	The first assimilation cycle starts at this time. The ensemble spin-up cycle period (form DATE_START to DATE_CYCLE_START) can be longer than assimilation cycles, controlled by this start time.
DATE_CYCLE_END =201809121800	The last assimilation step is at this time.
CYCLE_PERIOD=180	Time period between assimilation cycles.
WRFOUT_INTERVAL=(180 180 180)	Output interval for WRF forecasts (MAX_DOM).
LBC_INTERVAL=360	Time interval at which LBC tendencies are available in wrfbdy files.
OBS_WIN_MIN=-90	Observation window left offset relative to DATE.
OBS_WIN_MAX=90	Observation window right offset relative to DATE.
MINUTES_PER_SLOT=30	Interval of each time slot in 4DVar, number of time slots is (OBS_WIN_MAX-OBS_WIN_MIN)/MINUTES_PER_SLOT
FORECAST_MINUTES=360	Default WRF deterministic forecast run period. If FORECAST_TO_END=true, this will be overwritten to the actual forecast period that goes all the way to DATE_END.

Define directories, where code and data are stored.

WORK_DIR=$SCRATCH/$EXP_NAME	Runtime directory for the experiment. Usually located on scratch system so that large amount of data can be output temporarily.
SCRIPT_DIR=$WORK/PSU_WRF_EnKF	Top-level PSU system code directory.
CODE_DIR=$WORK/code	Top-level code directory storing external packages.
DATE_DIR=$WORK/data	Top-level data directory.

WPS_DIR=$CODE_DIR/WPSV3	Location of WPS code.
WRF_DIR=$CODE_DIR/WRFV3	Location of WRF model code.
WRF_BC_DIR=$CODE_DIR/WRF_BC_v2.1_alltime	Location of WRF_BC code.
WRFDA_DIR=$CODE_DIR/WRFDAV3	Location of WRFDA code package.
MULTI_INC_DIR=$CODE_DIR/MULTI_INC	Location of Multi-incremental utility functions.
ENKF_DIR=$CODE_DIR/EnKF/src	Location of EnKF code.
FG_DIR=$DATA_DIR/fnl	First guess from global forecast or reanalysis that provides IC and BC for the experiment.
GEOG_DIR=$WORK/data/geog	Location of geog data for WPS geogrid.exe
BE_DIR=$DATA_DIR	Location of background error covariance file be.dat for module_perturb_ic.sh.
TCVITALS_DIR=$WORK/data/nhc/tcvitals	Location of TC Vitals best track data.

Domain configuration for WRF. After the description, (MAX_DOM) indicates that entry is an array with one for each nested domain.

TWO_WAY_NESTING=true	If true, use two-way nesting (feedback=1) for WRF.
MAX_DOM=3	Total number of nested domains.
E_WE=(379 298 222)	Number of grid points in west-east direction (MAX_DOM).
E_SN=(244 298 222)	Number of grid points in south-north direction (MAX_DOM).
E_VERT=(35 35 35)	Number of vertical levels (MAX_DOM).
DX=(27000 9000 3000) DY=(27000 9000 3000)	Grid spacing in meters (MAX_DOM).
TIME_STEP=(90 30 10)	Model time step in seconds (MAX_DOM).

PARENT_ID=(0 1 2)	Domain id (MAX_DOM).
GRID_RATIO=(1 3 3)	Ratio of grid spacing between parent and child domains (MAX_DOM).
TIME_STEP_RATIO=(1 3 3)	Ratio of time steps between parent and child domains (MAX_DOM).
I_PARENT_START=(1 100 100)	Start index in west-east direction in the parent domain for the lower left corner of the child domain (MAX_DOM).
J_PARENT_START=(1 70 100)	Start index in south-north direction in the parent domain for the lower left corner of the chile domain (MAX_DOM).
MAP_PROJ="mercator"	Map projection method.
REF_LAT=25.0	Reference latitude of the largest domain.
REF_LON=-65.0	Reference longitude of the largest domain.
STAND_LON=-65.0	Standard longitude of the largest domain.
TRUELAT1=30.0 TRUELAT2=60.0	True latitude 1 and 2 (in lambert projection).
P_TOP=1000	Model top level pressure.

WRF physics parameterization configuration. Extra options can be added by modifying the namelist_wrf.sh file. For example, if you want to make sst_skin in namelist.input a configuration file entry, change the "sst_skin=1" line in namelist_enkf.sh to "sst_skin=${SST_SKIN:-1}", where 1 is the default value if SST_SKIN is not set in configuration file. See WRF manual for more details of available options.

MP_PHYSICS=(6 6 6)	Microphysics parametrization option (MAX_DOM).
RA_LW_PHYSICS=(1 1 1)	Longwave radiation scheme (MAX_DOM).
RA_SW_PHYSICS=(1 1 1)	Shortwave radiation scheme (MAX_DOM).
RADT=(27 9 3)	Time interval of radiation scheme calculation (MAX_DOM).
SF_SFCLAY_PHYSICS=(1 1 1)	Surface layer scheme (MAX_DOM).

SF_SURFACE_PHYSICS=(1 1 1)	Land surface model (MAX_DOM).
BL_PBL_PHYSICS=(1 1 1)	Boundary layer model (MAX_DOM).
BLDT=(0 0 0)	Time interval of boundary layer calculation (MAX_DOM).
CU_PHYSICS=(0 0 0)	Cumulus parameterization scheme (MAX_DOM).
CUDT=(5 5 5)	Time interval of cumulus scheme calculation (MAX_DOM).
SST_UPDATE=0	0 = do not update SST 1 = update SST, will require SST input from wrflowinp file.

EnKF options. Those options are connected to those in namelist_enkf.sh.

NUM_ENS=60	Ensemble size.
NMCPU=16	Number of processors used in nens+1 dimension (how many groups of members are there).
NICPU=4	Number of processors used in zonal direction when decomposing the domain.
NJCPU=4	Number of processors used in meridional direction when decomposing the domain.
INFLATION_COEF=1.0	Multiplicative inflation factor applied before assimilating observations, λ as in (2.20).
RELAX_OPT=0	Covariance relaxation methods: 0 = relax-to-prior-perturbation (Zhang et al. 2004) 1 = relax-to-prior-spread (Whitaker and Hamill 2012)
RELAX_ADAPTIVE=false	If true, turn on adaptive algorithm to estimate relaxation coefficient online using innovation statistics (Ying and Zhang 2015).
RELAXATION_COEF=0.8	Relaxation coefficient, α as in (2.21).

REPLACE_MEAN=false	After EnKF analysis step, if true, replace ensemble mean with fields defined by REPLACE_MEAN_WITH.
REPLACE_MEAN_WITH="forecast"	"forecast": replace ensemble mean with a deterministic forecast from previous cycle. "gfs": replace ensemble mean with GFS forecast from global model.
INCLUDE_LITTLE_R=true	Include LITTLE_R formatted data as input to obsproc.exe.
INCLUDE_BUFR=false	Include BUFR formatted data as input to obsproc.exe. This will require BUFR decoders.
INCLUDE_MADIS=false	Include MADIS formatted data as input to obsproc.exe.

Switches of observation types that are shared in namelist_enkf.sh and namelist_wrfvar.sh to include/exclude certain observations.

	namelist_enkf.sh	**namelist.wrfvar.sh**
USE_SYNOPOBS=true	N/A	use_synopobs
USE_SURFOBS=true	use_surface	N/A
USE_SOUNDOBS=true	use_sounding	use_soundobs
USE_PILOTOBS=true	N/A	use_pilotobs
USE_PROFILEROBS=true	use_profiler	use_profilerobs
USE_AIREPOBS=true	use_aircft	use_airepobs
USE_METAROBS=true	use_metar	use_metarobs
USE_SHIPSOBS=true	use_sfcshp	use_shipsobs
USE_SSMIOBS=true	use_spssmi	N/A
USE_SATEMOBS=true	N/A	use_satemobs
USE_GPSPWOBS=true	use_gpspw	use_gpspwobs
USE_GPSREFOBS=true	N/A	use_gpsrefobs
USE_ATOVS=true	use_atovs	N/A
USE_GEOAMVOBS=true	use_satwnd	use_geoamvobs
USE_POLARAMVOBS=true	N/A	use_polaramvobs
USE_QSCATOBS=true	N/A	use_qscatobs
USE_RADAROBS=true	N/A	use_radarobs
USE_RADAR_RF=true	use_radar_rf	N/A

USE_RADAR_RV=true	use_radar_rv	N/A
USE_AIRBORNE_RF=true	use_airborne_rf	N/A
USE_AIRBORNE_RV=true	use_airborne_rv	N/A
USE_BOGUSOBS=true	N/A	use_bogusobs
USE_BUOYOBS=true	N/A	use_buoyobs

Data thinning in EnKF namelist

THIN_SURFACE=0	Data thinning parameter for surface observations.
THIN_SOUNDING=0	Data thinning parameter for sounding observations.
THIN_PROFILER=0	Data thinning parameter for profiler observations.
THIN_AIRCFT=0	Data thinning parameter for aircft observations.
THIN_METAR=0	Data thinning parameter for metar observations.
THIN_SFCSHP=0	Data thinning parameter for sfcshp observations.
THIN_SPSSMI=0	Data thinning parameter for spssmi observations.
THIN_ATOVS=0	Data thinning parameter for atovs observations.
THIN_SATWND=0	Data thinning parameter for satwnd observations.
THIN_GPSPW=0	Data thinning parameter for gpspw observations.
THIN_RADAR=0	Data thinning parameter for radar/airborne observations.
THIN_RADIANCE=0	Data thinning parameter for radiance observations.

Localization distance, which is the radius of influence ROI, also known as the cutoff distance (where the covariance is tapered to zero).

HROI_SFC=900	Horizontal ROI for surface observations (in km).
HROI_UPPER=2000	Horizontal ROI for upper-air observations (in km).
HROI_RADAR=45	Horizontal ROI for radar observations (in km).
HROI_RADIANCE=30	Horizontal ROI for satellite radiance (in km).
VROI=5	Vertical ROI (number of vertical levels).

VROI_RADAR=15	Vertical ROI for radar observations (number of vertical levels).

WRFDA options.

CV_OPTIONS=3	3 = CV3 option, error covariance from global model climatology (uses be.dat.cv3 from the WRFDA package). 5 = CV5 option, error covariance calculated from ensemble forecasts using the NMC method (generating customized be.dat file using gen_be from the WRFDA package).
VAR_SCALING1=1.0 VAR_SCALING2=1.0 VAR_SCALING3=1.0 VAR_SCALING4=1.0 VAR_SCALING5=1.0	Variance scaling factors for: 1: streamfunction; 2: unbalanced velocity potential; 3 – unbalanced temperature; 4 – pseudo relative humidity; 5 – unbalanced surface pressure.
LEN_SCALING1=1.0 LEN_SCALING2=1.0 LEN_SCALING3=1.0 LEN_SCALING4=1.0 LEN_SCALING5=1.0	Length scaling factor.
MAX_EXT_ITS=3	Maximum number of outer loops (evaluating nonlinear trajectory and observation priors).
NTMAX=80	Maximum number of inner loops (minimizing cost function).
VAR4D_LBC=false	If true, include boundary condition in the cost function.
ALPHACV_METHOD=2	α control variable method for hybrid data assimilation: 1 = ensemble perturbations in control variable space. 2 = ensemble perturbations in model space.
JE_FACTOR=1.25	Ensemble covariance weighting factor. This is equivalent to $1/\beta$, where β is from (5.1).

Parallel algorithm configuration.

`HOSTPPN=12`	Number of processors per node (for a particular HPC the experiment is running on).
`HOSTTYPE="jet"`	Type of HPC. Used in job_submit.sh
`JOB_SUBMIT_MODE=2`	Job submission mode 1 = submit top-level `run_cycle.sh` to the queue as one batch job and components run by `mpiexec` calls. 2 = execute `run_cycle.sh` in the command line, and submit each component as separate batch jobs to the queue.
`real_ntasks=16`	Number of processors used for `real.exe` jobs.
`wrf_ntasks=64`	Number of processors used for WRF deterministic forecast, or for each member of the WRF ensemble forecasts.
`var3d_ntasks=64`	Number of processors used for 3DVar jobs using WRFDA.
`var4d_ntasks=256`	Number of processors used for 4DVar jobs using WRFDA.
`var4d_ppn=16`	Number of processors per node (ppn) used for 4DVar jobs.
`enkf_ntasks=256`	Number of processors used for EnKF jobs.
`enkf_ppn=16`	Number of processors per node (ppn) used for EnKF jobs. If the memory required by each processor is too large, one can reduce ppn so that each processor get allocated larger memory.

References

Barker, D.M., W. Huang, Y.-R. Guo, A. J. Bourgeois, and Q. N. Xiao, 2004: A three-dimensional variational data assimilation system for MM5: Implementation and initial results. Mon. Wea. Rev., 132, 897–914.

Evensen, G., 1994: Sequential data assimilation with a nonlinear qua- si-geostrophic model using Monte Carlo methods to forecast error statistics. J. Geophys. Res., 99, 10 143–10 162.

Gaspari, G., and S. E. Cohn, 1999: Construction of correlation func- tions in two and three dimensions. Quart. J. Roy. Meteor. Soc., 125, 723–757.

Houtekamer, P., H. Mitchell, G. Pellerin, M. Buehner, M. Charron, L. Speak, and B. Hansen, 2005: Atmospheric data assimilation with an en- semble Kalman filter: Results with real observations. Mon. Wea. Rev., 133, 604–620.

Houtekamer, P. L., and F. Zhang, 2016: Review of the Ensemble Kalman Filter for Atmospheric Data Assimilation. Mon. Wea. Rev., 144, 4489-4532.

Kalman, R., 1960: A new approach to linear filtering and prediction theory. Trans. ASME: J. Basic Eng., 83, 95–108.

Skamarock, W. C., J. B. Klemp, J. Dudhia, D. O. Gill, D. M. Barker, W. Wang, and J. G. Powers, 2005: A description of the advanced research WRF version 2. NCAR Technical Note, NCAR/TN-468+STR.

Snyder, C., and F. Zhang, 2003: Assimilation of simulated Doppler radar observations with an ensemble Kalman filter. Monthly Weather Review, 131, 1663-1677.

Wang, Y., Y. Jung, T.A. Supinie, and M. Xue, 2014: A hybrid MPI-OpenMP parallel algorithm and performance analysis for an ensemble square root filter designed for multiscale observations. *J. Atmos. Oceanic Tech.*, 30, 1382-1397.

Weng, Y. and F. Zhang, 2012: Assimilating Airborne Doppler Radar Observations with an Ensemble Kalman Filter for Cloud-resolving Hurricane Initialization and Prediction: Katrina (2005). Monthly Weather Review, 140, 841-859.

Whitaker, J. S., and T. M. Hamill, 2002: Ensemble data assimilation without perturbed observations. Mon. Wea. Rev., 130, 1913– 1924.

Zhang, F., C. Snyder, and J. Sun, 2004: Impacts of initial estimate and observation availability on convective-scale data assimilation with an ensemble Kalman filter. Monthly Weather Review, 132, 1238-1253.

Zhang, F., Weng, Y., Sippel, J. A., Meng, Z., and Bishop, C. H., 2009: Convection-permitting Hurricane Initialization and Prediction through Assimilation of Doppler Radar Observations with an Ensemble Kalman Filter: Humberto (2007). Mon. Wea. Rev., 137, 2105-2125.