EnKF Ensemble Kalman Filter

COMMUNITY VERSION 1.0

compatible with GSI community release v3.4

User's Guide

July 2015

Developmental Testbed Center

National Center for Atmospheric Research
National Centers for Environmental Prediction, NOAA
Earth System Research Laboratory, NOAA

Foreword

This User's Guide for the community Ensemble Kalman Filter data analysis system (EnKF) is geared particularly towards beginners. It describes the fundamentals of using EnKF, including basic skills of installing, running, and diagnosing and tuning EnKF.

EnKF version 1.0 was released on July 31, 2015. Its code is based on a revision of the EnKF repository from March 2015, with its supplemental libraries and joined files with Gridpoint Statistical Interpolation (GSI) based on community GSI version 3.4.

This User's Guide includes six chapters and one appendix:

This document is updated annually. For the latest version of this document, please visit the EnKF User's Website at:

http://www.dtcenter.org/EnKF/users/index.php.

Citation:

```
Developmental Testbed Center, 2015: Ensemble Kalman Filter (EnKF)
User's Guide for Version 1.0. Available at
http://www.dtcenter.org/EnKF/users/docs/index.php, 55 pp.
```

Please send questions and comments to enkf_help@ucar.edu.

Contributors to this guide:

Hui Liu, Ming Hu, Donald Stark, Jeff Whitaker, Hui Shao, Kathryn Newman

Acknowledgements:
We thank the National Oceanic and Atmospheric Administration Hurricane Forecast Improvement Program (HFIP) and Office of Oceanic and Atmospheric Research (OAR) for their support of this work. This work is also facilitated by National Center for Atmospheric Research (NCAR). NCAR is supported by the National Science Foundation (NSF).

Table of Contents

Chapter 1: Overview

EnKF history and background:

The ensemble Kalman Filter (EnKF) is a Monte-Carlo algorithm for data assimilation that uses an ensemble of short-term forecasts to estimate the background-error covariance in the Kalman Filter. Each ensemble member is cycled through the data assimilation system and updated by the EnKF.

The NOAA operational EnKF system provides the ensemble that is used within the operational GSI Ensemble-Variational Hybrid data assimilation system. The EnKF code was developed by the National Oceanic and Atmospheric Administration (NOAA) Earth System Research Lab (ESRL) in collaboration with the research community. It contains two separate algorithms for calculating an analysis increment, a serial Ensemble Square Root (EnSRF) described by article Whitaker and Hamill (DOI: http://dx.doi.org/10.1175/1520-0493(2002)130<1913:EDAWPO>2.0.CO;2) and a Local Ensemble Kalman Filter (LETKF) algorithm described by the article by Hunt et al. (DOI: 10.1016/j.physd.2006.11.008). The parallelization scheme used by the EnSRF algorithm is based on that used in the Data Assimlation Research Testbed (DART) toolkit developed at NCAR and described by Collins and Anderson (DOI: http://dx.doi.org/10.1175/JTECH2049.1). The LETKF code was contributed by Yoichiro Ota of the Japanese Meteorological Agency (JMA) while he was a visitor at the NOAA National Centers for Environmental Prediction (NCEP).

The EnKF code became operational in May 2012 as part of the GSI Hybrid 3D-Ensemble Variational Data Assimilation system upgrade to the global forecast system (GFS). It was initially managed as a separate project, but was merged with the GSI project in December 2014. It has the capability to work not only with the GFS global forecast model, but also the HWRF, NAM and WRF-ARW regional models.

EnKF becomes community code:

In 2014, the Developmental Testbed Center (DTC) started to collaborate with major development groups to transform the EnKF operational system into a community system following the same rules as the GSI community system (http://www.dtcenter.org/com-GSI/users/). The DTC has complemented the development groups in providing EnKF documentation, porting EnKF to multiple platforms that have worked with GSI, and testing EnKF in an independent and objective environment, while still maintaining functionally equivalent to operational centers. Working with code developers, the DTC is maintaining a community EnKF repository that is equivalent to the operational developmental repository and facilitates community users to contribute EnKF development. Based on the repository, the DTC is planning to release EnKF code annually with intermediate bug fixes. The beta v1.0 of the EnKF system was released on January 31, 2015, with the formal v1.0 release on July 31, 2015. The DTC began providing full community support following the formal release of EnKF.

GSI code management and Review Committee:

The EnKF code is managed through the administration of the Data Assimilation Review Committee (DRC). The DRC originated from the GSI Review Committee, formed in 2010. The committee was reformed in 2014 to include members representing EnKF aspects. Such a combination will further enhance collaboration of development teams in both variational and ensemble data assimilation areas. Currently, DRC is composed of members from NCEP/EMC, NASA/GMAO, NOAA/ESRL, NCAR/MMM, NOAA's Satellite and Information Service (NESDIS), the United States Air Force, University of Maryland, and the DTC.

The DRC primarily steers distributed GSI and EnKF development and community code management and support. The responsibilities of the Review Committee are divided into two major aspects: coordination and code review. The purpose and guiding principles of the Review Committee are as follows:

Coordination and Advisory

- Propose and shepherd new development
- Coordinate on-going and new development
- Process management
- Community support recommendation

Code Review

- Establish and manage a unified GSI/EnKF coding standard followed by all developers.
- Establish and manage a process for proposal and commitment of new developments to the GSI/EnKF repository.
- Review proposed modifications to the code trunk.
- Make decisions on whether code change proposals are accepted or denied for inclusion in the repository and manage the repository.
- Oversee the timely testing and inclusion of code into the repository.

The review committee is committed to facilitating the transition from research to operations (R2O). The DTC is the only committee member, with responsibilities to provide central support of GSI/EnKF to the general research community.

About this EnKF release:

This users guide is composed for the version 1.0 release of EnKF. It is the first users guide for this system produced by the DTC, with assistance from the code developers. This version of EnKF is based on a revision of the community EnKF repository from March 2015. Please note that currently the primary focus of the DTC is on testing and evaluation of EnKF for regional Numerical Weather Prediction (NWP) applications, though intructions and cases for running global EnKF applications are available with this release. Although the EnKF is released with its own version number, it is closely tied to the GSI v3.4 release. The GSI v3.4 and EnKF v1.0 are contained in the same code package due to

the EnKF's dependency on the GSI observation operator. Therefore, throughout this docuementation, the GSI User's Guide is referenced. This document may be obtained at http://www.dtcenter.org/com-GSI/users/docs/index.php.

Observations used by this version EnKF:

EnKF is using the GSI system as the observation operator to generate observation innovations. Therefore, the types of observations used in EnKF are similar to the GSI. This version of EnKF has been tested to work with the community GSI release version 3.4. It can assimilate, but is not limited to, the following types of observations:

Conventional observations: (including satellite retrievals)

- Radiosondes
- Pibal winds
- Synthetic tropical cyclone winds
- Wind profilers: US, JMA
- Conventional aircraft reports
- ASDAR aircraft reports
- MDCARS aircraft reports
- Dropsondes
- MODIS IR and water vapor winds
- GMS, JMA, and METEOSAT cloud drift IR and visible winds
- EUMETSAT and GOES water vapor cloud top winds
- GEOS hourly IR and coud top wind
- Surface land observations
- Surface ship and buoy observation
- SSM/I wind speeds
- QuikScat, ASCAT and OSCAT wind speed and direction
- SSM/I and TRMM TMI precipitation estimates
- Doppler radial velocities
- VAD (NEXRAD) winds
- GPS precipitable water estimates
- GPS Radio occultation (RO) refractivity and bending angle profiles
- SBUV ozone profiles, MLS (including NRT) ozone, and OMI total ozone
- SST
- Tropical storm VITAL (TCVital)
- PM2.5
- MODIS AOD (when using GSI-chem package)
- Doppler wind Lidar data
- Radar radial wind and reflectivity Mosaic
- METAR cloud observations
- Tail Dopppler Radar (TDR) radial velocity and super-observation
- Flight level and Stepped Frequency Miscrowave Radiometer (SFMR) High Density Observation (HDOB) from reconnaissance aircaraft
- Tall tower wind

Satellite radiance/brightness temperature observations (instrument/satellite ID)

- SBUV: n17, n18, n19
- HIRS: metop-a, metop-b, n17, n19
- GOES_IMG: g11, g12

- AIRS:aqua
- AMSU-A: metop-a, metop-b, n15, n18, n19, aqua
- AMSU-B: metop-b, n17
- MHS: metop-a, metop-b, n18, n19
- SSMI: f14, f15
- SSMIS: f16
- AMSRE: aqua
- SNDR: g11, g12, g13
- IASI: metop-a, metop-b
- GOME: metop-a, metop-b,
- OMI: aura
- SEVIRI: m08, m09, m10
- ATMS: NPP
- CRIS: NPP

The structure of this User's Guide:

The User's Guide is organized as follows:

Chapter 1 provides a background introduction of the EnKF operational and community system, EnKF review committee, and data types that can be used in this version.

Chapter 2 contains basic information about how to get started with EnKF– including system requirements; required software (and how to obtain it); how to download EnKF; and information about compilers, libraries, and how to build the code.

Chapter 3 focuses on the input files needed to run EnKF and how to configure and run GSI and EnKF through a sample run script. This chapter also provides an example of a successful EnKF run.

Chapter 4 includes information about diagnostics and tuning of the EnKF system through EnKF standard output and namelist variables.

Chapter 5 illustrates how to setup and run the GSI observer and EnKF for a regional configuration containing conventional, gpsro, and radiance data types and a global configuration using conventional observations, as well as how to diagnose the results.

Chapter 6 introduces EnKF theory and the main structure of the code.

Appendix A describes the contents of the EnKF namelist.

Chapter 2: Software Installation

2.1 Introduction

The DTC community EnKF is a community distribution of NOAA's operational ensemble Kalman filter. The community EnKF expands the portability of the operational code by adding a flexible build system and providing example run scripts that allow the system to run on many common platforms. The current version of the community EnKF is fully supported (e.g. builds and runs) on most Linux platforms using the Intel and PGI compilers.

This chapter describes how to build and install the DTC community EnKF on your computing resources. These instructions apply only to the DTC community EnKF. The source code for the community EnKF release is identical to the particular revision of NCEP's trunk code frozen for community release. The only difference from the NCEP trunk code, is the addition of the more general community build system in order to support a wider variety of computing platforms.

The EnKF build process consists of four general steps:

- Obtain the source code: combined GSI/EnKF, and WRF.
- Build the WRF model (see the WRF users guide).
- Set the EnKF code defaults (see the EnKF users guide)
- Build the GSI model (see the GSI users guide).
- Build the EnKF

Section 2.2 describes how to obtain the source code. Section 2.3 presents an outline of the build process. Sections 2.4 and 2.5 cover the system requirements (tools, libraries, and environment variable settings) and currently supported platforms in detail. Section 2.6 discusses what to do if you have problems with the build and where to get help.

2.2 Obtaining the source code

The community EnKF code and the GSI code are released as a combined source code package. The current EnKF release is v1.0 and is paired with the community GSI release version 3.4.

The community EnKF release is available from the DTC community EnKF users website;

```
http://www.dtcenter.org/EnKF/users/index.php
```

The community GSI/EnKF release includes the source code for both the EnKF v1.0 and the GSI v3.4 models, as well as an integrated build system, utilities, and documentation

necessary to build and run the EnKF. To assist the users, an ARW based practice case is provided as a download from the community website.

To download the source code from the either the GSI or the EnKF website, select the ***Download*** tab along with the ***GSI/EnKF System*** subtab on the vertical menu located on the left side of the main page. New users must first register before downloading the source code. Returning users only need to enter their registration email address to log in. After accessing the download page, select the link to the ***comGSIv3.4-EnKFv1.0*** tarball. Please only use the source code provided with the ***comGSIv3.4-EnKFv1.0*** tarball. Do not mix and match this tarball with other versions of the community GSI code or supplemental libraries, as this will lead to unpredictable results.

The community EnKF version 1.0 comes in a tar file named `comGSIv3.4-EnKFv1.0.tar`. The tar file may be unpacked by using the standard UNIX commands:

```
gunzip comGSIv3.4-EnKFv1.0.tar.gz
tar -xvf comGSIv3.4-EnKFv1.0.tar
```

This creates the top level GSI directory `comGSIv3.4-EnKFv1.0/.`

After downloading the source code, and prior to building, the user should check the *known issues* link on the download page of DTC website to determine if any bug fixes or platform specific customizations are needed.

2.3 Compiling EnKF

This section starts with a quick outline of how to build EnKF, followed by a discussion of the general build requirements. Typically, EnKF will build "straight out of the box" on any system that successfully builds GSI. Should the user experience any difficulties with the default build, check the build environment against the requirements described at the end of section 2.4.

Currently the EnKF code requires access to both a WRF and GSI build, so start by building both. The default EnKF build settings produce a regional version of EnKF, called *wrf_enkf*, which assumes that WRF style I/O is being used. EnKF is then built by running "*make*" in the *src/main/enkf* directory. This build makes use of the *configure.gsi* file produced for the GSI build.

Other I/O configurations for EnKF are available, and must be selected at build time. The current choices are regional, global or NMMB. The choice of I/O configuration is specified in the file src/main/enkf/Makefile.conf. Section 2.3.2 provides a full explanation on specifying the I/O configuration.

2.3.1 Build Overview

This section provides a quick outline of the steps necessary to build the EnKF code from the release distribution.

1. **Set the environment for the compiler:** If not already done so, set the necessary paths for using your selected compiler, such as loading the appropriate modules or modifying the path.
2. **If not already done, build and install a recent version of the WRF model**. The WRF build is currently needed for the WRF I/O libraries and should use the same compiler as used for the EnKF and GSI builds.
3. **Build GSI** (see chapter 2 of the GSI users guide for more details)
    a. **Set the environment variables** (see chapter 2.4.2 of the GSI users guide)
    b. **Run the configure script** located at the main GSI system directory.
    c. **Run the compile script**
    d. **Confirm that GSI has successfully built.**
4. **Configure and Build EnKF**
    a. **Change into the directory src/main/enkf**
    b. **Select the enkf configuration** (the default is regional, see section 2.3.2)
    c. **Run "make"**
    d. **Confirm that the EnKF executable resides in the directory src/main/enkf**

2.3.2 Versions of EnKF

The EnKF code has three build time configurations for the I/O; regional, global, and NMMB. The EnKF analysis is identical in each case, only the capability to digest model input differs. The regional version can only digest WRF formatted I/O files. The global version can only digest spectral input from the NCEP global model. Lastly, the NMMB version can only digest NMMB files.

Before initiating the compile command, the user must select which EnKF configuration is to be built by manually editing the file src/main/enkf/Makefile.conf. Examining lines 45 to 50 of Makefile.conf shows three pairs of build flags.

```
45  #  FFLAGS_F90    = -DGFS
46  #  EXE_FILE = global_enkf
47     FFLAGS_F90    = -DWRF
48     EXE_FILE = wrf_enkf
49  #  FFLAGS_F90    = -DNMMB
50  #  EXE_FILE = nmmb_enkf
```

By default the build system is set to build the regional configuration. This sets the Fortran preprocessor flag to "-DWRF", and the executable name to "*wrf_enkf*." Other EnKF configurations are selected by commenting out lines by adding "#" symbols and activating lines by removing "#" symbols. Once the desired configuration is selected by editing the file "Makefile.conf", the executable may be built by running the the top level GSI configure/compile commands.

2.4 System Requirements and External Libraries

The EnKF source code is written in FORTRAN 90 and requires some flavor of MPI and OpenMP for the distributed memory parallelism. Lastly the I/O relies on the NetCDF I/O libraries. The build system relies on standard make.

The basic requirements for building are:

- FORTRAN 95+ compiler
- MPI v1.2+
- OpenMP
- NetCDF V3.6.3 or V4.2+
- LAPACK and BLAS mathematics libraries, or equivalent
- WRF V3.4.1+

Because all but the last of these tools and libraries are typically the purview of system administrators to install and maintain, they are lumped together here as part of the basic system requirements.

2.5 Compilers tested for release

Version 1.0 of the DTC community EnKF system has been successfully tested on a variety of Linux platforms with many versions of the Intel and PGI Fortran compilers.

The following Linux compiler combinations have been fully tested:

	Fortran Compiler version	C Compiler version
Intel	ifort 12.1.5,13.0.1,13.1.2,14.0.2,15.0.0,15.0.1	icc or gcc 4.4.5
PGI	pgf90 12.10,13.2,13.9,13.10,14.3,14.7,14.9,15.1	pgcc or gcc 4.4.5

Unforeseen build issues may occur when using older compiler and library versions. As always, the best results come from using the most recent version of compilers.

2.6 Getting Help and Reporting Problems

Should the user experience any difficulty building EnKF on their system, please first confirm that both the WRF model and the GSI have successfully built. Should the EnKF build fail, but the other two succeed, feel free to contact the community EnKF support at enkf_help@ucar.edu for assistance.

At a minimum, when reporting issues building the code, please include a copy of the EnKF build log.

Chapter 3: Running EnKF

This chapter discusses the process of running EnKF cases. It includes:

1. Discussions of the input data required to run EnKF.
2. A detailed explanation of how to run both a regional and global EnKF with the released run scripts.
3. Introduction to the files in a successful regional and global EnKF run directory

3.1 Input Data Required to Run EnKF

In most cases, three types of input data, ensemble mean and members, observations, and fixed files, must be available before running EnKF.

- **Ensemble mean and members**

The ensemble members and ensemble mean of certain regional and global ensemble forecast systems are used as the background for the EnKF analysis. When this EnKF system is used for regional analysis with WRF ensembles, the ensemble members and ensemble mean follow the naming convention:

firstguess.mem001
firstguess.mem002
.....
firstguess.ensmean

Please note that the number of allocated computer cores to run EnKF must be larger than the ensemble size.

The ensemble members can be generated using various methods, such as:

- Using global/regional ensemble forecasts.
- Ensemble forecasts generated using multi-physics, multi-models, or adding random perturbations drawn from climatology.
- In cycling assimilation, using ensemble forecasts initialized from previous ensemble analyses generated by EnKF.

This version EnKF can use any of the following ensemble files as the background:

- ARW NetCDF forecast
- NMM NetCDF forecast
- GFS forecast files

- **Prepare observation ensemble priors (observation innovation)**

In addition to the ensemble backgrounds on model grids, the ensemble priors of all observations (observation innovation for all ensemble members) are also needed to run EnKF. The observation ensemble priors are generated by running GSI observation forward operators with the ensemble members as backgrounds (without doing actual GSI analyses). In this release, the GSI v3.4 run script includes options for generating the observation ensemble priors (details in section 3.2). The observation ensemble priors files should follow the following naming conventions:

- For conventional observations:
  diag_conv_ges.mem001
  diag_conv_ges.mem002
  ••••••••
  diag_conv_ges.ensmean

- For radiance observations:
  diag_instrument_Satellite.mem001 (e.g. diag_hirs4_n19.mem001)
  diag_ instrument_Satellite.mem002 (e.g. diag_hirs4_n19.mem002)
  ••••••••
  diag_ instrument_Satellite.ensmean (e.g. diag_hirs4_n19.ensmean)

These diag files contain a lot of information about each observation. For more details on the content of diag files, please refer to the GSI User's Guide Appendix A.2.

The preparation of observations for EnKF assimilation is done within GSI, including quality control of observations, selection of observation types for assimilation, and observation error tuning. In the default namelist situation, NO additional online quality control of observations is performed in the EnKF analysis step (although there is an option to do the similar quality control of observations as the GSI variational scheme.)

- **Fixed files**

EnKF uses the the same fixed files as GSI to setup the analysis configurations. Detailed explanation of all fixed files provided in the community GSI system is in the GSI user's Guide, Chapter 3. The following is a list of fixed files needed for EnKF analyses:

- for observation control:
  convinfo - conventional data (prepufr) info file
  ozinfo - ozone retrieval info file
  satinfo - satellite channel info file

- when satellite radiance data are assimilated, the following files are needed to do the adaptive radiance bias correction:

satbias_angle - satellite angle dependent file
satbias_in - satellite bias correction coefficient file

Note that a single file may be used when adaptive bias correction is configured to use the new combined satellite angle dependent and mass bias correction coeffients. See GSI User's Guide for more detail.

When the namelist parameter *readin_localization* is set to "*true*", file "*hybens_locinfo*" is needed, in which customized localization values varying by model level are contained.

3.2 EnKF and GSI Observer Run Scripts

In this release version, three sample run scripts for EnKF applications are under directory *comGSIv3.4-EnKFv1.0/run*:

- ***run_gsi_regional.ksh*** for running regional GSI to generate the observation ensemble priors. Referred to as the GSI observer run scripts.
- ***run_gsi_global.ksh*** GSI observer run script for global applications (to generate the observation ensemble priors).
- ***run_enkf_wrf.ksh*** for running regional EnKF
- ***run_enkf_global.ksh*** for running global EnKF

These run scripts are introduced in detail in the following sections. Also provided are two scripts for generating the GSI and EnKF namelist:

- ***comgsi_namelist.sh*** generates GSI namelist on the fly (called by GSI observer run scripts).
- ***comgsi_namelist_gfs.sh*** generates GSI namelist on the fly (called by GSI observer run scripts) for global GSI applications.
- ***enkf_wrf_namelist.sh*** generates EnKF namelist on the fly (called by EnKF run script) for regional EnKF applications.

3.2.1 General introduction to the run scripts

These run scripts provide the run time environment necessary for running the GSI and EnKF executables. They all have similar steps, as follows:

1. Request computer resources to run GSI/EnKF.
2. Set environmental variables for the machine architecture.
3. Set experimental variables (such as experiment name, analysis time, background, and observation).
4. Check the definitions of required variables.

5. Generate a run directory for GSI/EnKF
6. Copy the GSI/EnKF executable to the run directory.
7. Copy/Link the background file/ensemble files to the run directory.
8. Link observations to the run directory.
9. Link fixed files (statistic, control, and coefficient files) to the run directory.
10. Generate namelist for GSI/EnKF.
11. Run the GSI/EnKF executable.
12. Post-process: save analysis results, generate diagnostic files, clean run directory.

In the GSI User's Guide, three sections explain the first three steps in detail:

- Section 3.2.2.1: Setting up the machine environment (step 1)
- Section 3.2.2.2: Setting up the running environment (step 2)
- Section 3.2.2.3: Setting up an analysis case (step 3)

For this documentation, the first 2 steps will be skipped and the 3rd step in the GSI observer and EnKF run scripts will be discussed. The community GSI analysis run script and the GSI observer run script are one in the same, with the GSI observer capabilitiy controlled by flags that turn off the minimization, select appropriate namelist options and enable looping through all the ensemble members to generate the ensemble observation priors for each member, including the ensemble mean.

3.2.2 GSI observer run scripts

3.2.2.1 Setting up an case

This section discusses variables specific to the user's case, such as analysis time, working directory, background and observation files, location of fixed files and CRTM coefficients, and the GSI executable file. The script looks like:

```
####################################################
# case set up (users should change this part)
####################################################
#
# ANAL_TIME= analysis time  (YYYYMMDDHH)
# WORK_ROOT= working directory, where GSI runs
# PREPBURF = path of PreBUFR conventional obs
# BK_FILE  = path and name of background file
# OBS_ROOT = path of observations files
# FIX_ROOT = path of fix files
# GSI_EXE  = path and name of the gsi executable
  ANAL_TIME=2012022506
  WORK_ROOT=…/enkf/regional/gsidiag_nmm
  OBS_ROOT=…/enkf/enkfdata/obs
  PREPBUFR=${OBS_ROOT}/gdas1.t06z.prepbufr.nr
  BK_ROOT=…/enkf/enkfdata/nmm/bk
  BK_FILE=${BK_ROOT}/firstguess.ensmean
  CRTM_ROOT=…/gsi/CRTM_REL-2.1.3
  GSI_ROOT=…/enkf/code/comGSIv3.4-EnKFv1.0
  FIX_ROOT=${GSI_ROOT}/fix
  GSI_EXE=${GSI_ROOT}/run/gsi.exe
```

```
GSI_NAMELIST=${GSI_ROOT}/run/comgsi_namelist.sh
```

The options `ANAL_TIME`, `WORK_ROOT`, `PREPBURF`, `BK_FILE`, `OBS_ROOT`, `FIX_ROOT`, `GSI_EXE` are all the same settings as the GSI analysis configuration. Two options: `BK_ROOT` , `GSI_ROOT`, are the root directories for ensemble members and the GSI system. These exist to make links to the background and GSI system easy and shorter. The new option: `GSI_NAMELIST` is needed because the namelist section was taken out of the run scripts in this release as a separate file to improve the structure and readability of the run scripts. Users can find the namelist files for both GSI and EnKF in the same directory as the run scripts. Please note the option `BK_FILE` is pointing to the ensemble mean.

The next part of this block has additional options to specify other important aspects of the GSI observer.

```
#-------------------------------------------------
# bk_core= which WRF core is used as background (NMM or ARW or NMMB)
# bkcv_option= which background error covariance and parameter will be used
#              (GLOBAL or NAM)
# if_clean = clean  : delete temperal files in working directory (default)
#            no     : leave running directory as is (this is for debug only)
  bk_core=NMM
  bkcv_option=NAM
  if_clean=clean
# if_observer = Yes  : only used as observation operater for enkf
# no_member     number of ensemble members
# BK_FILE_mem   path and base for ensemble members
  if_observer=Yes
  no_member=20
  BK_FILE_mem=${BK_ROOT}/firstguess.mem
```

The options `bk_core`, `bkcv_option`, and, `if_clean` are the same as in the GSI analysis run scripts. The new option `if_observer` indicates if this GSI run is for the generation of the observation ensemble priors or for a regular GSI run. The new option `no_member` specifies the number of ensemble members that are need to be calculated for the observation ensemble priors. This should also be the ensemble number in the EnKF analysis. Option `BK_FILE_mem` is the path and the name of the ensembles without the ensemble member ID appended. The scripts will add the ensemble member ID as a three digital number, such as 000, 001,… .

3.2.2.2 Loop through ensemble members

As mentioned previously, the GSI ensemble observer run scripts are the same as the GSI analysis run scripts released with the community GSI. Since the observer only generates diag files, which includes useful information on the observation innovation, the GSI outer loop number for the observer should be set to 0 to skip all minimization iterations.

The contents of the run scripts can be divided into two parts, those before the following comments and those after:

```
#
```

```
################################################
# start to calculate diag files for each member
################################################
#
```

Before this comment, the scripts have the same functionality as when running a GSI analysis, except that the background options in the scripts for the observer funcationality are set for the ensemble mean. Additionally, the namelist is built with the two following options set, which skips the minimization and saves all observation processing from the ensemble mean:

```
if [ ${if_observer} = Yes ] ; then
nummiter=0
if_read_obs_save='.true.'
if_read_obs_skip='.false.'
```

Please refer to the GSI user's guide for detailed explanation of the remainder of this portion of the run scripts. The second portion of the script loops through each member to calculate the observation ensemble priors based on the GSI run environments setup by the first portion.

Listed below is an annotated version of the 2nd part of the GSI observer run script (`Courier New`) with explanations on each function block.

```
if [ ${if_observer} = Yes ] ; then
```

This 2nd part of the script only runs if option `if_observer` is set to "Yes". The diag files from the ensemble mean need to be saved first with the following commands:

```
string=ges
for type in $listall; do
  count=0
  if [[ -f diag_${type}_${string}.${ANAL_TIME} ]]; then
     mv diag_${type}_${string}.${ANAL_TIME} diag_${type}_${string}.ensmean
  fi
done
mv wrf_inout wrf_inout_ensmean
```

The following section builds the namelist for ensemble members. Please note two options need to be set different between mean and members:

```
# Build the GSI namelist on-the-fly for each member
  nummiter=0
  if_read_obs_save='.false.'
  if_read_obs_skip='.true.'
. $GSI_NAMELIST
cat << EOF > gsiparm.anl

 $comgsi_namelist

EOF
```

The `if_read_obs_save` and `if_read_obs_skip` switch from "True" and "False", respectively for the mean to "False" and "True", respectively for the ensemble members.

This saves all observation processing (including bias correction, thinning, etc) from the ensemble mean and skips the observation processing step for the ensemble members to keep observations constant.

The script loops through each ensemble member (from member *001* to *no_member*) to create the diag files for each member:

```
# Loop through each member
  loop="01"
  ensmem=1
  while [[ $ensmem -le $no_member ]];do

     rm pe0*

     print "\$ensmem is $ensmem"
     ensmemid=`printf %3.3i $ensmem`
```

After a member is processed, the script removes the old ensemble member and links to the new member before rerunning the calculation:

```
# get new background for each member
     if [[ -f wrf_inout ]]; then
       rm wrf_inout
     fi

     BK_FILE=${BK_FILE_mem}${ensmemid}
     echo $BK_FILE
     ln -s $BK_FILE wrf_inout
```

Run the GSI observer for this member:

```
#  run  GSI
     echo ' Run GSI with' ${bk_core} 'for member ', ${ensmemid}

     case $ARCH in
        'IBM_LSF')
          ${RUN_COMMAND} ./gsi.exe < gsiparm.anl > stdout_mem${ensmemid} 2>&1  ;;

        * )
           ${RUN_COMMAND} ./gsi.exe > stdout_mem${ensmemid} 2>&1 ;;
     esac

#  run time error check and save run time file status
     error=$?

     if [ ${error} -ne 0 ]; then
       echo "ERROR: ${GSI} crashed for member ${ensmemid} Exit status=${error}"
       exit ${error}
     fi

     ls -l * > list_run_directory_mem${ensmemid}
```

Generate diag files for this member:

```
# generate diag files

     for type in $listall; do
           count=`ls pe*${type}_${loop}* | wc -l`
```

```
         if [[ $count -gt 0 ]]; then
            cat pe*${type}_${loop}* > diag_${type}_${string}.mem${ensmemid}
         fi
       done

# next member
      (( ensmem += 1 ))

  done

fi
```

Since all members are using the same run directory, the run status of each member is overwriten by the following member. Any useful information should be preserved in this run directory before the next member starts.

3.2.3 Sample regional EnKF run scripts

As described in section 3.2.1, the regional EnKF run scripts have been designed to have a similar structure to the GSI analysis and observer run scripts. Again, please refer to the GSI User's Guide section 3.2.2.1 and 3.2.2.2 for the first two steps.

The 3rd step is to setup the variables specific to the user's case, such as analysis time, working directory, background and observation files, location of fixed files and CRTM coefficients, and the EnKF executable. Most of the options in this portion are the same as the 3rd step in the GSI observer run scripts, which is discussed in section 3.2.2.1 of this User's Guide. Users should setup most of the variables in this portion based on the options in the GSI observer run scripts as they are the variables to setup the same things for the GSI and EnKF. The following is a sample script with explanations:

```
##################################################
# case set up (users should change this part)
##################################################
#
# ANAL_TIME= analysis time  (YYYYMMDDHH)
# WORK_ROOT= working directory, where GSI runs
# PREPBURF = path of PreBUFR conventional obs
# BK_FILE  = path and name of background file
# OBS_ROOT = path of observations files
# FIX_ROOT = path of fix files
# GSI_EXE  = path and name of the gsi executable

  ANAL_TIME=2013091218
  WORK_ROOT=…/enkf/regional/enkf_nmm
  diag_ROOT=…/enkf/regional/gsidiag_nmm
  BK_ROOT=…/enkf/enkfdata/nmm/bk
  BK_FILE=${BK_ROOT}/firstguess.ensmean
  GSI_ROOT=…/enkf/code/comGSIv3.4-EnKFv3.3
  FIX_ROOT=${GSI_ROOT}/fix
  ENKF_EXE=${GSI_ROOT}/src/main/enkf/wrf_enkf
  CRTM_ROOT=…/CRTM_REL-2.1.3
  ENKF_NAMELIST=${GSI_ROOT}/run/enkf_wrf_namelist.sh
```

Options ANAL_TIME, BK_ROOT, BK_FILE, GSI_ROOT, FIX_ROOT, CRTM_ROOT have the same meanings as the ensemble GSI observer run scripts and should be set to the same values as the ensemble GSI observer run scripts. The option WORK_ROOT is the working directory which should have enough space to hold the ensemble members and EnKF analysis results. The option diag_ROOT is pointing to the run directory of the GSI observer, where the diag files are generated as data input for the EnKF. The option ENKF_EXE points to the EnKF executable, which is under the GSI source code directory in this release. The option ENKF_NAMELIST is the path and the EnKF namelist file, which sits outside of the run script as a separate file like the GSI namelist. Users can find the namelist files for both GSI and the EnKF in the same directory as the run scripts.

The next part of this block includes several additional options that specify several aspects of the ensemble members.

```
# ensemble parameters
#
  NMEM_ENKF=20
  BK_FILE_mem=${BK_ROOT}/firstguess
  NLONS=375
  NLATS=375
  NLEVS=60
  IF_ARW=.false.
  IF_NMM=.true.
  list="conv hirs4_n19 mhs_n19"
#  list="conv amsua_n18 mhs_n19 hirs4_n19"
#
```

Options NMEM_ENKF, BK_FILE_mem are also in the GSI observer run script and should be set to the same values as the GSI observer run script. Options NLONS, NLATS, and NLEVS specify 3 dimensions (XYZ) of the ensemble grid. Options IF_ARW and IF_NMM indicates which background, ARW NetCDF ensemble or the NMM NetCDF ensemble, is used in this EnKF run. Option list is a list of the observation types that the EnKF will use in the analysis. This list should be based on the diag files generated by the ensemble GSI observer.

At this point, users should be able to run the EnKF for simple cases without changing the rest of the script. However, some advanced users may need to change some of the following blocks for special applications.

```
#
####################################################
# Users should NOT change script after this point
####################################################
#
```

The next block sets the run command to run EnKF on multiple platforms. The ARCH is set at the beginning of the script.

```
case $ARCH in
   'IBM_LSF')
      ###### IBM LSF (Load Sharing Facility)
      RUN_COMMAND="mpirun.lsf " ;;

   'LINUX')
```

```
      if [ $GSIPROC = 1 ]; then
         #### Linux workstation - single processor
         RUN_COMMAND=""
      else
         ###### Linux workstation -  mpi run
        RUN_COMMAND="mpirun -np ${GSIPROC} -machinefile ~/mach "
      fi ;;

   'LINUX_LSF')
      ###### LINUX LSF (Load Sharing Facility)
      RUN_COMMAND="mpirun.lsf " ;;

   'LINUX_PBS')
      #### Linux cluster PBS (Portable Batch System)
#      RUN_COMMAND="mpirun -np ${GSIPROC} " ;;
      RUN_COMMAND="mpiexec_mpt -n ${GSIPROC} " ;;


   'DARWIN_PGI')
      ### Mac - mpi run
      if [ $GSIPROC = 1 ]; then
         #### Mac workstation - single processor
         RUN_COMMAND=""
      else
         ###### Mac workstation -  mpi run
         RUN_COMMAND="mpirun -np ${GSIPROC} -machinefile ~/mach "
      fi ;;

   * )
     print "error: $ARCH is not a supported platform configuration."
     exit 1 ;;
esac
```

The following block sets up fixed files and some analysis-time related values:

```
# Given the analysis date, compute the date from which the
# first guess comes.  Extract cycle and set prefix and suffix
# for guess and observation data files
# gdate=`$ndate -06 $adate`
gdate=$ANAL_TIME
YYYYMMDD=`echo $adate | cut -c1-8`
HH=`echo $adate | cut -c9-10`

# Fixed files
# CONVINFO=${FIX_ROOT}/global_convinfo.txt
# SATINFO=${FIX_ROOT}/global_satinfo.txt
# SCANINFO=${FIX_ROOT}/global_scaninfo.txt
# OZINFO=${FIX_ROOT}/global_ozinfo.txt
CONVINFO=${diag_ROOT}/convinfo
SATINFO=${diag_ROOT}/satinfo
SCANINFO=${diag_ROOT}/scaninfo
OZINFO=${diag_ROOT}/ozinfo
# LOCINFO=${FIX_ROOT}/global_hybens_locinfo.l64.txt
```

The next block creates a working directory (`workdir`) in which EnKF will run. The directory should have enough disk space to hold all the files needed for this run. This directory is cleaned before each run, therefore, save all the files needed from the previous run before rerunning EnKF.

```
# Set up workdir
```

```
rm -rf $WORK_ROOT
mkdir -p $WORK_ROOT
cd $WORK_ROOT
```

After creating a working directory, copy or link the EnKF executable, ensembles, diag files (observations), bias correction coefficients, and fixed files into the working directory.

```
cp $ENKF_EXE        ./enkf.x

cp $CONVINFO        ./convinfo
cp $SATINFO         ./satinfo
cp $SCANINFO        ./scaninfo
cp $OZINFO          ./ozinfo
# cp $LOCINFO         ./hybens_locinfo

cp $diag_ROOT/satbias_in ./satbias_in
cp $diag_ROOT/satbias_angle ./satbias_angle

# get mean
ln -s ${BK_FILE_mem}.ensmean ./firstguess.ensmean
for type in $list; do
   ln -s $diag_ROOT/diag_${type}_ges.ensmean .
done

# get each member
imem=1
while [[ $imem -le $NMEM_ENKF ]]; do
   member="mem"`printf %03i $imem`
   ln -s ${BK_FILE_mem}.${member} ./firstguess.${member}
   for type in $list; do
      ln -s $diag_ROOT/diag_${type}_ges.${member} .
   done
   (( imem = $imem + 1 ))
done
```

The following script is used to generate the EnKF namelist called ***enkf.nml*** in the working directory. Some namelist variables are explained in detail in Section 4.3. Appendix A gives a full list of namelist options.

```
# Build the GSI namelist on-the-fly
. $ENKF_NAMELIST
cat << EOF > enkf.nml

 $enkf_namelist

EOF
```

Copy the ensemble background files to the working directory and rename them as "*analysis.${member}*". The EnKF will update those files as the analysis results.

```
# make analysis files
cp firstguess.ensmean analysis.ensmean
# get each member
imem=1
while [[ $imem -le $NMEM_ENKF ]]; do
   member="mem"`printf %03i $imem`
   cp firstguess.${member} analysis.${member}
   (( imem = $imem + 1 ))
done
```

The following block runs EnKF and checks if the EnKF has successfully completed.

```
#
###################################################
#  run  EnKF
###################################################
echo ' Run EnKF'

${RUN_COMMAND} ./enkf.x < enkf.nml > stdout 2>&1

#####################################################################
#  run time error check
#####################################################################
error=$?

if [ ${error} -ne 0 ]; then
  echo "ERROR: ${ENKF_EXE} crashed  Exit status=${error}"
  exit ${error}
fi
```

If this point is reached, the EnKF successfully finishes and exits with 0:

```
exit
```

3.3 Understanding Resulting Files in GSI observer and EnKF run directory

To check if the GSI observer and EnKF runs have been successfully finished, it is important to understand the meaning of each file in the run directory.

3.3.1 The GSI Observer Run Directory

After customizing the GSI observer run script to your personal environment, it may be submitted to the batch system just as any other job.

Following a successful run, the majority of the files in the GSI observer run directory will be the same as those in a sucessful GSI analysis run directory. The difference for the observer run is that the GSI observer generates more diag and stdout files related to each ensemble member. Below is an example of the files generated in the run directory from a GSI observer run:

```
amsuabufr                   fit_p1.2013091218    l2rwbufr
amsubbufr                   fit_q1.2013091218    list_run_directory
anavinfo                    fit_rad1.2013091218  list_run_directory_mem001
berror_stats                fit_t1.2013091218    list_run_directory_mem002
convinfo                    fit_w1.2013091218    list_run_directory_mem003
diag_amsua_n15_ges.ensmean  fort.201             list_run_directory_mem004
diag_amsua_n15_ges.mem001   fort.202             list_run_directory_mem005
diag_amsua_n15_ges.mem002   fort.203             mhsbufr
diag_amsua_n15_ges.mem003   fort.204             ozinfo
diag_amsua_n15_ges.mem004   fort.205             pcpbias_out
diag_amsua_n15_ges.mem005   fort.206             pcpinfo
diag_conv_ges.ensmean       fort.207             prepbufr
diag_conv_ges.mem001        fort.208             prepobs_prep.bufrtable
diag_conv_ges.mem002        fort.209             satbias_angle
diag_conv_ges.mem003        fort.210             satbias_in
diag_conv_ges.mem004        fort.211             satbias_out
```

```
diag_conv_ges.mem005        fort.212            satinfo
diag_hirs4_n19_ges.ensmean  fort.213            sigf03
diag_hirs4_n19_ges.mem001   fort.214            stdout
diag_hirs4_n19_ges.mem002   fort.215            stdout.anl.2013091218
diag_hirs4_n19_ges.mem003   fort.217            stdout_mem001
diag_hirs4_n19_ges.mem004   fort.218            stdout_mem002
diag_hirs4_n19_ges.mem005   fort.219            stdout_mem003
diag_mhs_n19_ges.ensmean    fort.220            stdout_mem004
diag_mhs_n19_ges.mem001     fort.221            stdout_mem005
diag_mhs_n19_ges.mem002     gpsrobufr           wrfanl.2013091218
diag_mhs_n19_ges.mem003     gsi.exe             wrf_inout
diag_mhs_n19_ges.mem004     gsiparm.anl         wrf_inout_ensmean
diag_mhs_n19_ges.mem005     hirs3bufr
errtable                    hirs4bufr
```

This case was a regional analysis with WRF/NMM NetCDF backgrounds. In this case, 5 ensemble members are used to generate the diag files. A brief introduction of the additional files in the GSI observer runs is given below:

diag_conv_ges.mem??? *diag_conv_ges.ensmean*	Diag files of conventional observations for ensemble member ??? and ensemble mean
diag_{instrument}_{satellite}_ges.mem??? *diag_{instrument}_{satellite}_ges. ensmean*	Diag files of satellite radiance observation for ensemble member ??? and ensemble mean.
stdout_mem???	Standard output from GSI observer run for ensemble member ???
list_run_directory_mem???	The list of the files in the run directory after the GSI observer is finished for ensemble member ???

3.3.2 Files in the EnKF Run Directory

After customizing the EnKF run script to your personal environment, it may be submitted to the batch system just as any other job.

Upon successful assimilation, the ensemble analyses (both members and ensemble mean), covariance inflation factor to the ensemble analyses, and updated satellite bias correction coefficients (if requested) are output in the run directory. Below is an example of the files generated in the run directory from one of the EnKF regional test cases:

```
analysis.ensmean            diag_amsua_n18_ges.mem018   diag_hirs4_n19_ges.mem015
analysis.mem001             diag_amsua_n18_ges.mem019   diag_hirs4_n19_ges.mem016
analysis.mem002             diag_amsua_n18_ges.mem020   diag_hirs4_n19_ges.mem017
analysis.mem003             diag_conv_ges.ensmean       diag_hirs4_n19_ges.mem018
analysis.mem004             diag_conv_ges.mem001        diag_hirs4_n19_ges.mem019
analysis.mem005             diag_conv_ges.mem002        diag_hirs4_n19_ges.mem020
analysis.mem006             diag_conv_ges.mem003        diag_omi_aura_ges.ensmean
analysis.mem007             diag_conv_ges.mem004        diag_sbuv2_n16_ges.ensmean
analysis.mem008             diag_conv_ges.mem005        diag_sbuv2_n17_ges.ensmean
analysis.mem009             diag_conv_ges.mem006        diag_sbuv2_n18_ges.ensmean
analysis.mem010             diag_conv_ges.mem007        diag_sbuv2_n19_ges.ensmean
analysis.mem011             diag_conv_ges.mem008        enkf.log
analysis.mem012             diag_conv_ges.mem009        enkf.nml
analysis.mem013             diag_conv_ges.mem010        enkf.x
analysis.mem014             diag_conv_ges.mem011        firstguess.ensmean
analysis.mem015             diag_conv_ges.mem012        firstguess.mem001
```

```
analysis.mem016              diag_conv_ges.mem013           firstguess.mem002
analysis.mem017              diag_conv_ges.mem014           firstguess.mem003
analysis.mem018              diag_conv_ges.mem015           firstguess.mem004
analysis.mem019              diag_conv_ges.mem016           firstguess.mem005
analysis.mem020              diag_conv_ges.mem017           firstguess.mem006
convinfo                     diag_conv_ges.mem018           firstguess.mem007
covinflate.dat               diag_conv_ges.mem019           firstguess.mem008
diag_amsua_n18_ges.ensmean   diag_conv_ges.mem020           firstguess.mem009
diag_amsua_n18_ges.mem001    diag_gome_metop-a_ges.ensmean  firstguess.mem010
diag_amsua_n18_ges.mem002    diag_gome_metop-b_ges.ensmean  firstguess.mem011
diag_amsua_n18_ges.mem003    diag_hirs4_n19_ges.ensmean     firstguess.mem012
diag_amsua_n18_ges.mem004    diag_hirs4_n19_ges.mem001      firstguess.mem013
diag_amsua_n18_ges.mem005    diag_hirs4_n19_ges.mem002      firstguess.mem014
diag_amsua_n18_ges.mem006    diag_hirs4_n19_ges.mem003      firstguess.mem015
diag_amsua_n18_ges.mem007    diag_hirs4_n19_ges.mem004      firstguess.mem016
diag_amsua_n18_ges.mem008    diag_hirs4_n19_ges.mem005      firstguess.mem017
diag_amsua_n18_ges.mem009    diag_hirs4_n19_ges.mem006      firstguess.mem018
diag_amsua_n18_ges.mem010    diag_hirs4_n19_ges.mem007      firstguess.mem019
diag_amsua_n18_ges.mem011    diag_hirs4_n19_ges.mem008      firstguess.mem020
diag_amsua_n18_ges.mem012    diag_hirs4_n19_ges.mem009      ozinfo
diag_amsua_n18_ges.mem013    diag_hirs4_n19_ges.mem010      satbias_angle
diag_amsua_n18_ges.mem014    diag_hirs4_n19_ges.mem011      satbias_in
diag_amsua_n18_ges.mem015    diag_hirs4_n19_ges.mem012      satinfo
diag_amsua_n18_ges.mem016    diag_hirs4_n19_ges.mem013      stdout
diag_amsua_n18_ges.mem017    diag_hirs4_n19_ges.mem014
```

This case was a regional analysis with WRF/ARW NetCDF backgrounds. In this case, 20 ensemble members are used to estimate ensemble covariance, and both conventional observations (prepbufr) and radiance observations (AMSU-A and HIRS4) are assimilated.

A brief introduction of the files is given below:

stdout	A text output file. This is the most commonly used file to check the analysis processes as well as basic and important information about the analyses. The contents of stdout are explained in detail in Chapter 4 and users are encouraged to read this file to become familiar with the order of EnKF analysis processing.
firstguess. mem001-0??	ensemble members of WRF background, in NetCDF format. This is in the same format as the WRF forecast. The ensemble background of the analysis variables are extracted from the files.
firstguess.ensmean	ensemble mean of WRF background, in NetCDF format.
analysis. mem001-0??:	ensemble analysis if EnKF completes successfully. The format is the same as the background file.
analysis.ensmean	ensemble mean of analyses, in NetCDF format.
diag_conv_ges. mem001-0??:	diag files in binary for conventional and GPS RO observations, which contain the observations and their priors.
covinflate.dat	Three-dimensional multiplicative inflation factor fields (from

	function inflate_ens in *module inflation*). These inflation fields can be visualized using plotting software.
**info (convinfo, satinfo,…):*	info files that control data usage. Please see GSI User guide Chapter 4 for details.
satbias_in	the input coefficients of mass bias correction for satellite radiance observations.
satbias_angle	the input coefficients of scan angle bias correction for satellite radiance observations.

Chapter 4: EnKF Diagnostics and Tuning

This chapter will discuss how to assess whether an EnKF was successful based on the standard output. Properly checking the EnKF output will also provide useful information to diagnose potential errors in the system. The chapter begins with an introduction to the content and structure of the EnKF standard output (**stdout**). Followed by detailed discussion of tuning options in the namelist.

This chapter follows the online exercise for case 2012102506. This case uses a WRF-ARW NetCDF ensemble file as the background and analyzes several observations typical for operations, including most conventional observation data, and several radiance data (AMSU-A and HIRS4). The case was run on a Linux cluster supercomputer, using 32 cores. Users can follow this test to reproduce the following results by visiting:

http://www.dtcenter.org/com-GSI/users/tutorial/index.php

4.1 Understand the standard output from EnKF (*stdout*)

Upon a completion of an EnKF run, it is always useful to do a quick check of the standard output (*stdout),* to assess the performance of EnKF. The standard output file has information about whether the EnKF analysis has successfully completed, if the ensemble spread inflation looks good, and if the background and analysis fields are reasonable. Understanding the content of this file can also be very helpful for users to find clues in the event of a crash.

The EnKF **stdout** has following information:

a. namelist configuration
b. background ensemble members of observations and analysis variables
c. statistics of the ensemble prior
d. domain and observation partition
e. statistics of the ensemble analysis
f. spread inflation of the analysis ensemble

The following section contains a detailed description of the content of **stdout**, explained using the WRF/ARW case: 2012102506. The analysis domain consists of 111 x 111 x 56 grid points. To keep the output concise and make it more readable, redundant content are omitted.

The following indicates the start of the EnKF analysis. It shows how many processors are used to run the analysis and the beginning time of this run:

```
Execute poe command line: poe  ./enkf.x
 running on          32  processors ...
```

```
* . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * . *
PROGRAM ENKF_ANL HAS BEGUN. COMPILED 2011319.55      ORG: NP25
STARTING DATE-TIME  JAN 16,2015  12:45:58.893   16  FRI   2457039
```

The following lines show the analysis date, which is input from namelist variable "*datein*":

```
The analysis date is 2012102506
```

The following lines show how many satellite observation types are set to be read in and if the radiance bias correction is updated before the analysis process:

```
 number of satellite radiance files used          59
 number of satellite ozone files used           7
 PARAMS: NOT UPDATING BIAS CORRECTION COEFFS, SET NUMBER OF ITERATIONS TO 1
 LUPD_SATBIASC, NUMITER =  F           1
```

The following lines show the namelist used in the analysis:

```
namelist parameters:
 --------------------
 &NAM_ENKF
 DATEIN  =  2012102506,
 DATAPATH        = ./                                ,
 IASSIM_ORDER    =           0,
 COVINFLATEMAX   =   100.0000    ,
 COVINFLATEMIN   =   1.000000    ,
 DETERMINISTIC   = T,
 SORTINC = T,
 CORRLENGTHNH    =   500.0000    ,

    ••••••••

IAU     = F,
 NHR_ANAL        =           6,
 LETKF_FLAG      = F,
 BOXSIZE =   90.00000    ,
 MASSBAL_ADJUST  = F,
 USE_EDGES       = F,
 EMISS_BC        = T
 /
 --------------------
```

The following lines show analysis time, ensemble size, the number of 3D analysis variables, and total number of 2D fields of the 3D variables, plus surface pressure (Ps).

```
analysis time 2012102506
          20  members
 first-guess forecast hour for analysis = 06
           5 3d vars to update
 total of         281  2d grids will be updated (including ps)
 using multiplicative inflation based on Pa/Pb
 Vars in Rad-Jacobian (dims)
 --------------------------
 tv                           0
 q                            0
 oz                           0
 u                            0
 v                            1
 sst                          2
```

The following lines show the actual analysis variables and the background type. Also, the maximum and minimum of the surface pressure are printed for a check:

```
 Updating U, V, T, QVAPOR, PH, and MU for WRF-ARW...
 Surface pressure (spressmn) min/max range:   660.558898925781
   1017.63061523438
```

The next lines display the content of convinfo:

```
READ_CONVINFO: tcp     112    0  1   3.00000       0   0   0   75.0000      5.00000
1.000000      75.0000      0.00000        0  0.00000      0.00000        0
READ_CONVINFO: ps      120    0  1   3.00000       0   0   0   4.00000      3.00000
1.000000      4.00000     0.300000E-03    0  0.00000      0.00000        0
 line ignored in convinfo due to use flag  ps              132           0
          -1
```

••••••••

```
READ_CONVINFO: gps       3    0  1   3.00000       0   0   0  10.00000     10.00000
1.000000     10.00000      0.00000        0  0.00000      0.00000        0
 line ignored in convinfo due to use flag  gps             821           0
          -1
 line ignored in convinfo due to use flag  gps             440           0
          -1
 line ignored in convinfo due to use flag  pm2_5           102           0
          -1
```

The next 272 lines show the content of ozinfo:

```
OZINFO_READ:  jpch_oz=    272
  1 sbuv6_n14            lev =    1 use = -1 pob =     0.240 gross =   1.000 error =
1.000 b_oz =  10.000 pg_oz =   0.000
  2 sbuv6_n14            lev =    2 use = -1 pob =     0.490 gross =   1.000 error =
1.000 b_oz =  10.000 pg_oz =   0.000
  3 sbuv6_n14            lev =    3 use = -1 pob =     0.980 gross =   1.000 error =
1.000 b_oz =  10.000 pg_oz =   0.000
  4 sbuv6_n14            lev =    4 use = -1 pob =     1.950 gross =   1.000 error =
1.000 b_oz =  10.000 pg_oz =   0.000
```

••••••••

```
271 mls30_aura           lev =   54 use = -1 pob =   999.999 gross =   9.999 error =
9.999 b_oz =  10.000 pg_oz =   0.000
272 mls30_aura           lev =   55 use = -1 pob =   999.999 gross =   9.999 error =
9.999 b_oz =  10.000 pg_oz =   0.000
```

The next 2680 lines show the content of satinfo and starts reading in the radiance bias correction coefficients:

```
RADINFO_READ:  jpch_rad=   2680
   1 amsua_n15            chan=    1 var=   3.000 varch_cld=  9.100 use=  1 ermax=
4.500 b_rad=   10.00 pg_rad=   0.00 icld_det=-2
   2 amsua_n15            chan=    2 var=   2.000 varch_cld= 13.500 use=  1 ermax=
4.500 b_rad=   10.00 pg_rad=   0.00 icld_det=-2
   3 amsua_n15            chan=    3 var=   2.000 varch_cld=  7.100 use=  1 ermax=
4.500 b_rad=   10.00 pg_rad=   0.00 icld_det=-2
   4 amsua_n15            chan=    4 var=   0.600 varch_cld=  1.300 use=  1 ermax=
2.500 b_rad=   10.00 pg_rad=   0.00 icld_det=-2
```

••••••••

```
2678 avhrr3_metop-b       chan=    3 var=   0.450 varch_cld=  0.000 use= -1 ermax=
6.000 b_rad=   10.00 pg_rad=   0.00 icld_det=-1
```

```
2679 avhrr3_metop-b       chan=    4 var=   0.550 varch_cld=  0.000 use= -1 ermax=
6.000 b_rad=   10.00 pg_rad=   0.00 icld_det=-1
2680 avhrr3_metop-b       chan=    5 var=   0.600 varch_cld=  0.000 use= -1 ermax=
6.000 b_rad=   10.00 pg_rad=   0.00 icld_det=-1
```

The majority of next near 3000 lines show the content of radiance bias correction coefficients:

```
RADINFO_READ:  read satbias_angle file
 ***WARNING file scaninfo not found, use default

RADINFO_READ:  guess air mass bias correction coefficients below
   1            amsua_n15    0.472353   -0.231512    0.291223    0.000634   -
0.148959    0.000000    0.000000    0.000000
   2            amsua_n15   -0.677697    0.382025    1.424922   -0.000061
0.016514    0.000000    0.000000    0.000000
   3            amsua_n15   -2.631062    0.134578    2.968469   -0.004946
1.213581    0.000000    0.000000    0.000000
••••••••
2679       avhrr3_metop-b    0.000000    0.000000    0.000000    0.000000
0.000000    0.000000    0.000000    0.000000
2680       avhrr3_metop-b    0.000000    0.000000    0.000000    0.000000
0.000000    0.000000    0.000000    0.000000
```

Among the lines describing the radiance bias correction coefficients, the EnKF also starts to inventory the observation number and types for both conventional and radiance data. Observations of various types are read in (from the diag** files) and the number of the observations and time spent reading in the observations are shown:

Start to check the radiance observations:

```
    1             amsua_n15  nkeep=         0  num_obs_tot=         0
    2             amsua_n18  nkeep=      1467  num_obs_tot=      1467
    3             amsua_n19  nkeep=         0  num_obs_tot=      1467
    6            amsua_aqua  nkeep=         0  num_obs_tot=      1467
••••••••
```

Start to check the ozone observations:

```
    1             sbuv2_n16  nread=         0  nkeep=         0  num_obs_tot=     0
    2             sbuv2_n17  nread=         0  nkeep=         0  num_obs_tot=     0
   15          goes_img_g11  nkeep=         0  num_obs_tot=      1467

••••••••
   26             ssmis_f18  nkeep=         0  num_obs_tot=      1467
```

Start to check the conventional observations:

```
       3275  obs in diag_conv_ges file
 columns below obtype,nread, nkeep
    t        544        544
    q        196        196
   ps        835        835
   uv       1700       1700
  sst          0          0
  gps          0          0
   pw          0          0
   dw          0          0
  srw          0          0
   rw          0          0
  tcp          0          0
    6          gome_metop-a  nread=         0  nkeep=         0  num_obs_tot=     0
```

```
••••••••
   59              cris_npp  nkeep=         0  num_obs_tot=      1467
```

Give a summary on total number of convertional (1st number 3272), ozone (2nd number 0), and radiance observations (3rd number 1467) in diag files:

```
 nobs_conv, nobs_oz, nobs_sat =         3275           0        1467
```

Also, the following normalization factors for the radiance bias predictors could be inside the bias correction coefficient lines:

```
1 biasprednorm =    1.000000
        ••••••••
    8 biasprednorm =    1.000000
```

Then, the time spent in reading in the observations and the total number of observations kept and rejected are shown:

```
max time in mpireadobs  =   0.2323714
        4742 obs kept
           0 total obs rejected
 compressed total number of obs         4742 (        4742 )
 time in read_obs =   1.19471000880003      on proc           0
```

In the standard output, regional averaged (northern hemisphere – NH, southern hemisphere – SH, and tropics – TR) statistics of the ensemble priors' fit to all observations are provided, partly for checking if the inflation is appropriate. The details of this part are discussed in next Section 4.2.

Next, the analysis variable fields and the observations are distributed to the different processors. The following lines show the maximum and minimum number of observations and grid on subdomain and the time used to setup those decompositions:

```
 npts =        12321
 min/max number of points per proc =          339         440
 time to do model space decomp =   1.026771962642670E-003
 nobsgood =         4742
 min/max number of obs per proc =          148         149
 time to do ob space decomp =   2.333894371986389E-005
 sending out observation prior ensemble perts from root ...
 ... took   4.037544131278992E-004  secs
 time in load_balance =  9.612187743186951E-003 on proc           0
```

Then, the ensemble background members of the analysis variables are read in and the maximum and minimum values of the fields at each vertical level are displayed. The maximum and minimum values are useful for a quick confirmation that the background fields have been read successfully. The size of the real array of ensemble perturbations updated on each processor and the time spent to read in and distribute the background ensemble are also shown:

```
     anal_chunk size =      2992650
```

```
READGRIDDATA_ARW: U           1  -20.63564       14.28071
        ••••••••
READGRIDDATA_ARW: U          56  -17.74460       22.63066
        ••••••••
READGRIDDATA_ARW: V          57  -13.06189       19.37896
        ••••••••
READGRIDDATA_ARW: V         112  -15.13866       17.09937
        ••••••••
READGRIDDATA_ARW: T         113  -8.530156       16.55089
        ••••••••
READGRIDDATA_ARW: T         168   495.2496       544.1279
        ••••••••
READGRIDDATA_ARW: QVAPOR          169  9.8409411E-03  2.1045785E-02
        ••••••••
READGRIDDATA_ARW: QVAPOR          224  3.5570574E-06  6.7701626E-06
        ••••••••
READGRIDDATA_ARW: PH          225   3.366851       8.819902
        ••••••••
READGRIDDATA_ARW: PH          280   13110.12       15289.34
READGRIDDATA_ARW: MU          281  -891.6020       1907.053

time in readgridata on root  0.46434     secs
time to scatter state on root  0.47210     secs
time in read_ensemble =  0.94238  on proc           0
```

In EnSRF, observations can be skipped/not assimilated due to the adaptive observation thinning. The following lines show how many observations are skipped or not assimilated by this thinning. In this case, paoverpb_thresh = 0.99, which lead to 423 of the observations being skipped.

```
assimilate obs in order they were read in
         423  out of        4742 obs skipped
        2877  out of        4319  same lat/long

 1 timing on proc     0  =    0.61    0.14    0.00    0.01    0.46    0.01    0
 1 timing on proc    31  =    0.61    0.15    0.00    0.01    0.45    0.01    0

 time to broadcast obfit_post =   1.241117715835571E-004  secs, niter =
           1
 time to broadcast obsprd_post =   7.869303226470947E-005
```

Next, the update to the analysis variables is performed and the time for the updating is shown:

```
   time in enkf_update =  0.611589696258307      on proc           0
```

After updating the analysis variable fields, the multiplicative inflation to the spreads of analysis fields are done. The maximum and minimum of the actual inflation values to the analysis variables are shown. Users can check if the inflation values are reasonable.

```
min/max var 1 inflation =    1.000000       3.757024
min/max var 2 inflation =    1.000000       3.591980
min/max var 3 inflation =    1.000000       2.607338
min/max spfh inflation =    1.000000       2.986892
min/max ps inflation =    1.000000       3.694926
```

The regional averaged statistics of the inflation values are also shown for surface pressure:

```
global ps prior std. dev min/max =    36.07194       205.9449
```

```
 NH mean ps prior standard deviation =    76.92429
 NH mean ps posterior standard deviation (before inflation)=    52.80397
 NH mean ps posterior standard deviation (after inflation) =    74.21063
 NH mean ps inflation =    1.743494
 TR mean ps prior standard deviation =    77.48872
 TR mean ps posterior standard deviation (before inflation)=    62.23104
 TR mean ps posterior standard deviation (after inflation) =    75.86496
 TR mean ps inflation =    1.310104
 time in inflate_ens =  7.312259450554848E-002 on proc           0
```

After EnKF analysis, similar innovation statistics of the ensemble analyses are shown as ones for the ensemble priors' fit to all observations:

```
innovation statistics for posterior:
 conventional obs
 region, obtype, nobs, bias, innov stdev, sqrt(S+R), sqrt(S), sqrt(R):
NH    all ps    552 -0.364E-01  0.720E+00  0.159E+01  0.167E+00  0.158E+01
TR    all ps    283 -0.338E+00  0.943E+00  0.180E+01  0.278E+00  0.177E+01
NH     all t    317 -0.205E+00  0.127E+01  0.195E+01  0.411E+00  0.190E+01
TR     all t    227  0.211E+00  0.104E+01  0.946E+01  0.377E+00  0.946E+01
NH    all uv   1068 -0.281E-01  0.224E+01  0.425E+01  0.914E+00  0.415E+01
TR    all uv    632 -0.807E-01  0.248E+01  0.400E+01  0.855E+00  0.391E+01
NH     all q    112 -0.198E-01  0.144E+00  0.170E+00  0.390E-01  0.165E+00
TR     all q     84  0.234E-01  0.107E+00  0.183E+00  0.414E-01  0.178E+00
 satellite brightness temp
 instrument, channel #, nobs, bias, innov stdev, sqrt(S+R), sqrt(S), sqrt(R):
          amsua_n18   1    77  0.744E+00  0.244E+01  0.109E+02  0.130E+01  0.108E+02
          amsua_n18   2    72  0.519E+00  0.206E+01  0.103E+02  0.918E+00  0.103E+02
          amsua_n18   3    90 -0.244E+01  0.265E+01  0.773E+01  0.473E+00  0.772E+01
          amsua_n18   4    97 -0.353E+00  0.520E+00  0.142E+01  0.106E+00  0.142E+01
          amsua_n18   5    97  0.116E+00  0.268E+00  0.462E+00  0.726E-01  0.456E+00
          amsua_n18   6   154  0.276E+00  0.303E+00  0.325E+00  0.514E-01  0.321E+00
          amsua_n18   7   263  0.209E+00  0.261E+00  0.326E+00  0.467E-01  0.322E+00
          amsua_n18   8   279 -0.137E+00  0.250E+00  0.358E+00  0.730E-01  0.350E+00
          amsua_n18  10   213 -0.449E-01  0.246E+00  0.531E+00  0.145E+00  0.511E+00
          amsua_n18  11    71  0.974E-01  0.277E+00  0.694E+00  0.212E+00  0.661E+00
          amsua_n18  15    54 -0.219E+01  0.263E+01  0.772E+01  0.850E+00  0.768E+01
```

Finally, the minimum and maximum of the analysis increments are shown as below. The analysis increments should be within a reasonable range. The computational time of these steps are also shown:

```
 time to gather state on root  0.136818923056126      secs
 ens. mean anal. increment min/max ps  -269.1015       62.77728
 ens. mean anal. increment min/max var           1  -11.01143       8.155481
 ens. mean anal. increment min/max var           2  -6.704343       8.965038
 ens. mean anal. increment min/max var           3  -12.04501       3.471832
 ens. mean anal. increment min/max var           4 -0.3514615      0.2903060
 ens. mean anal. increment min/max var           5  -117.4268       273.7598
 time to gather ens mean increment on root  4.615368694067001E-002 secs
 time in writegriddata on root  0.219785105437040      secs
 time in write_ensemble =  0.402844041585922      on proc           0
```

The next line indicates the time of the analysis finish:

```
      ENDING DATE-TIME    JAN 16,2015  12:46:01.911   16  FRI   2457039
      PROGRAM ENKF_ANL HAS ENDED.
   * . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * .
```

The next lines show the computer resources used in the analysis and a sign that EnKF has finished, "all done":

```
* . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * .
****************RESOURCE STATISTICS*******************************
The total amount of wall time                        = 3.027633
The total amount of time in user mode                = 2.493620
The total amount of time in sys mode                 = 0.393940
The maximum resident set size (KB)                   = 163840
Number of page faults without I/O activity           = 51626
Number of page faults with I/O activity              = 0
Number of times filesystem performed INPUT           = 0
Number of times filesystem performed OUTPUT          = 0
Number of Voluntary Context Switches                 = 12899
Number of InVoluntary Context Switches               = 74
*****************END OF RESOURCE STATISTICS*************************

 all done!
```

4.2 Tuning of inflation and localization

Proper inflation and localization values need to be determined by experimenting with different inflation values. The goal is to make the total ensemble spreads of priors match the innovations as much as possible. In the tuning process, the vertical and horizontal structure of the match should be carefully examined. The 3D distribution of the inflation values can be plotted offline from the output file "covinflate.dat".

The tuning processes can vary for different resolutions and physics of the models as well as the observation types assimilated. The performance of the tuning may be better examined after multiple assimilation cycles allowing their effects to be accumulated and converged in cycling assimilation. Please also note that proper settings of observation error estimations are also critical for the inflation tuning.

Next, the test case is taken as an example to check if the inflation is appropriate. In the following lines of the standard output (*stdout*), the number (column `nobs`), mean (column `bias`), and standard deviation (column `innov stdev`) of the ensemble background fits to Ps, wind, temperature, and water vapor observations in the NH and TR are shown:

```
innovation statistics for prior:
 conventional obs
 region, obtype, nobs, bias, innov stdev, sqrt(S+R), sqrt(S), sqrt(R):
NH    all ps    552 -0.130E+01  0.152E+01  0.172E+01  0.674E+00  0.158E+01
TR    all ps    283 -0.748E+00  0.131E+01  0.189E+01  0.667E+00  0.177E+01
NH     all t    317 -0.282E+00  0.190E+01  0.201E+01  0.661E+00  0.190E+01
TR     all t    227  0.830E-01  0.140E+01  0.948E+01  0.689E+00  0.946E+01
NH    all uv   1068 -0.458E+00  0.369E+01  0.451E+01  0.176E+01  0.415E+01
TR    all uv    632 -0.272E+00  0.372E+01  0.425E+01  0.166E+01  0.391E+01
NH     all q    112 -0.707E-01  0.180E+00  0.191E+00  0.964E-01  0.165E+00
TR     all q     84  0.292E-01  0.134E+00  0.205E+00  0.101E+00  0.178E+00

 satellite brightness temp
 instrument, channel #, nobs, bias, innov stdev, sqrt(S+R), sqrt(S), sqrt(R):
           amsua_n18   1    77 -0.341E+00  0.243E+01  0.111E+02  0.259E+01  0.108E+02
           amsua_n18   2    72 -0.163E+00  0.213E+01  0.104E+02  0.179E+01  0.103E+02
           amsua_n18   3    90 -0.271E+01  0.293E+01  0.777E+01  0.905E+00  0.772E+01
           amsua_n18   4    97 -0.256E+00  0.481E+00  0.143E+01  0.151E+00  0.142E+01
           amsua_n18   5    97  0.273E+00  0.369E+00  0.467E+00  0.101E+00  0.456E+00
           amsua_n18   6   154  0.420E+00  0.439E+00  0.330E+00  0.752E-01  0.321E+00
           amsua_n18   7   263  0.294E+00  0.351E+00  0.333E+00  0.833E-01  0.322E+00
```

```
amsua_n18   8   279 -0.896E-01  0.334E+00  0.376E+00  0.137E+00  0.350E+00
amsua_n18  10   213  0.412E-01  0.484E+00  0.605E+00  0.324E+00  0.511E+00
amsua_n18  11    71  0.172E+00  0.370E+00  0.845E+00  0.526E+00  0.661E+00
amsua_n18  15    54 -0.266E+01  0.293E+01  0.789E+01  0.180E+01  0.768E+01
```

It would be desirable that the total ensemble spreads of prior/background match the innovation standard deviations as close as possible (Equation 15). In practice, one should check if there is any substantial ensemble spread deficiency. In the above sample standard output lines, the ensemble prior spreads are shown in the column *sqrt(S),* and the total spreads in the column *sqrt(S+R). Here, sqrt(R)* is the observation error.

The above sample statistics shows that in NH, the total spread of temperature observation priors (0.201E+01) is close to the innovation standard deviation (0.190E+01) of the priors fit to the observations. In the tropics, however, the total spread of temperature observation priors (0.948E+01) is much larger that standard deviation (0.140E+01), because of the very large observation error assigned (0.946E+01).

4.3 Tuning EnKF through key Namelist Options

The namelist parameters controlling the EnSRF analysis are set in the namelist sections: /*enkf_nml/,* /satobs_*enkf/,* and /*nam_wrf/ (all in* the file *enkf.nml*). To obtain a successful analysis, it is very important to setup/tune properly the namelist variables, particularly those related to inflation and localization, for each specific application configuration of various models and observation types.

4.3.1 Set up the analyses time and data location

The analysis time and working location are setup by the following parameters:

datein: analysis time (YYYYMMDDHH)

datapath: path to data directory (include trailing slash). In most cases, this should point to the current directory because the run scripts have setup the run environment in the working directory including copying the EnKF executable and data into the working directory.

4.3.2 Set up analysis algorithm

In the current implementation, a few variations/flavors of the EnKF are available, including the ensemble square root Kalman filter (EnSRF, *Whitaker et al., 2010, etc*), the perturbed observations EnKF, and the local transformed Kalman filter (LETKF). These options are set in the namelists below:

deterministic = true, use EnSRF without perturbed observations;

= false, use perturbed observations version of EnKF.

sortinc — if *deterministic* = false, re-order observations to minimize regression errors as described in *Anderson (2003)*.

letkf_flag — if true, use the LETKF scheme.

boxsize — observation box size for LETKF (deg)

iassim_order — = 0, assimilation of observations in the order they are read in;
= 1 in random order;
= 2 in order of predicted posterior variance reduction.

paoverpb_thresh — the threshold of the ratio of observation space posterior variance divided by prior variance (<= 1.0), that is, the expected reduction of prior variance by the observations.

Smaller values mean only those observations with stronger impact will be assimilated.

If =1, no thinning of observations is done. Typical values: 0.99 or 0.98.

4.3.3 Set up the analyses variables

The analyses variables for both global and regional models are hardcoded in the EnSRF. There are several options using the following parameters:

nvars: number of 3d model variables to update.

For hydrostatic global models, there are typically 5, including *U, V, T, QVAPOR, Ozone*. For non-hydrostatic regional models, like WRF, different combinations of the 3D and 2D analyses variables are set as below:

For ARW (*arw* =true)
nvars =3: U, V, T, and MU
=4: U, V, T, QVAPOR, and MU
=5: U, V, T, QVAPOR, PH, and MU
=6: U, V, W, T, QVAPOR, PH, and MU

For NMM (*nmm* =true)
nvars =3: U, V, T, and PD
=4: U, V, T, QVAPOR, and PD
=5: U, V, T, QVAPOR, CWM, and PD

The analyses variable list can be adjusted or additional variables, such as moisture related variables, can be added to the list by modifying the subroutines "*gridinfo.F90*" and "*gridio.F90*" accordingly.

pseudo_rh: use or not 'pseudo-rh' analysis variable, as in GSI.

cliptracers: if true, tracers are clipped to zero when read in, and just before they are written out.

4.3.4 Set up the ensemble backgrounds

The following parameters define which background fields will be used in the analyses and their dimensions:

regional = true, perform regional EnSRF analyses using either ARW or NMM inputs as the background.
=false, perform a global analysis. If either the parameter nmm or wrf is set to true, it will be set to true.

nmm if true, background comes from WRF NMM. When using other background fields, set it to false.

wrf if true, background comes from WRF ARW. When using other background fields, set it to false.

nlons number of grid points in longitude of the model background

nlats number of grid points in latitude of the model background

nlevs number of vertical levels of the model background

nanls number of ensemble members

4.3.5 Set up localization distances

In the current EnKF implementation, distances for localization can be set separately in the northern hemisphere, tropics and southern hemisphere, and in the horizontal, vertical and time dimensions, and for different observation types using namelist parameters. The length scales should be given in km for the horizontal, hours for time, and 'scale heights' (units of $-log(P/P_{ref})$) in the vertical.

There are two options to setup the localization distances in horizontal and vertical. These are decided by the namelist variable "*readin_localization"*:

readin_localization:	=.true., customized horizontal and vertical localization values varying with model levels are read in from the external text file "*hybens_locinfo*".
	= .false., the horizontal and vertical localization distances are set by the following parameters
corrlengthnh:	length for horizontal localization in km in the northern hemisphere (25N-90N, NH)
corrlengthtr:	length for horizontal localization in km in the tropics (25S-25N, TR)
corrlengthsh:	length for horizontal localization in km in the southern hemisphere (25S-90S, SH)
lnsigcutoffnh:	scale height for vertical localization in $-\log(P/P_{ref})$. in NH.
lnsigcutofftr:	scale height for vertical localization in $-\log(P/P_{ref})$. in TR.
lnsigcutoffsh:	scale height for vertical localization in $-\log(P/P_{ref})$. in SH.

The text file "*hybens_locinfo*" contains vertical profile of horizontal and vertical localization length scales (in e-folding scale). These scales are converted to the full distance width (*1/0.388 in km) of Gaspari-Cohn function, where it goes to zero. The horizontal and vertical scales can be adjusted for each observation types easily in the subroutine "*read_locinfo.f90*".

For satellite radiance and surface pressure observations, the vertical localization distances are set separately using the following parameters:

lnsigcutoffsatnh, lnsigcutoffsattr, lnsigcutoffsatsh; and
lnsigcutoffpsnh, lnsigcutoffpstr, lnsigcutoffpssh

There is also one option to setup the time localization window, which is the time away from the analyses time. This is decided by the following namelist variables:

obtimelnh:	time in hours away from the analyses time to move 2800 km at 30 ms^{-1} in the northern hemisphere
obtimeltr:	same as *obtimelnh* but for the tropics
obtimelsh:	same as *obtimelnh* but for the southern hemisphere

The empirical determination/tuning of proper localization distances is important for successful analyses with ensemble data assimilation. In general, large horizontal scale observations, like GPS radio occultation and radiosonde observations, require larger horizontal localization distances to fully take advantage of the observations. On the other hand, high horizontal resolution observations, like radar observations, may need shorter horizontal localization distances. The same is true for the localization distance in vertical.

4.3.6 Set up adaptive posterior inflation parameters

The inflation can be set up by the following parameters:

anapertwtnh:	inflation parameter in NH.
anapertwttr:	inflation parameter in TR.
anapertwtsh:	inflation parameter in SH.
The parameters	= 0 means no inflation. = 1 means inflation all the way back to prior spread. These are the key tuning parameters for the performance of EnSRF. Typical values: ~0.95.

The inflation factor fields can be smoothed out using the following parameter:

smoothparm:	parameter for smoothing inflation factor, = -1 for no smoothing. > 0, the estimated inflation factor is smoothed using a Gaussian spectral filter with an e-folding scale of the

parameter.

latbound: where the transition latitude starts (=25N or 25S)

delat: latitude width of transition zone where the inflation parameter is smoothed.

The minimum and maximum inflation values allowed can be controlled by the following parameters:

covinflatemin: minimum inflation factor
covinflatemax: maximum inflation factor

4.3.7 Satellite observations related parameters

The following parameters are used to setup the adaptive satellite bias correction:

lupd_satbias: = .true., update the satellite bias correction coefficients using the adaptive bias correction scheme (when the iterations of *numiter* > 1).
= .false., the bias correction coefficients are not updated.

numiter: number of iterations for state/bias correction coefficients update. (only relevant when satellite radiances assimilated, i.e. nobs_sat>0)

biasvar: background error variance for radiance bias coefficients (used in *radbias.f90*). The default is GSI value.

saterrfact: factor to multiply satellite radiance errors.

sattypes_rad: list of satellite observation types to be assimilated.

sattypes_oz: list of ozone observation types to be assimilated.

4.3.8 Observation QC parameters

In addition to the observation quality flags decided in the GSI observer, the EnSRF itself also conducts a gross check on the observations.

The outlier test computes the difference between the observation value and the prior ensemble mean. It then computes a standard deviation by taking the square root of the sum of the observation error variance and the prior ensemble variance for the observation. If the difference between the ensemble mean and the observation value is more than the specified number of standard deviations, then the observation is tossed. The threshold of the check is set as following:

sprd_tol: tolerance for gross check of observations.

Reject a observation if the prior mean is more than this many sqrt(S+R) from the observation, where S is ensemble prior variance and R is observation error variance. Typical value: ~3.0

The following parameters are used to control if the GSI quality control procedure is performed in EnSRF:

varqc: = .false., no variational quality control performed (default).

huber: = .true., use huber norm instead of "flat-tail" (default: .false.).

zhuberleft: departure parameter of huber norm.

zhuberright: departure parameter of huber norm.

Chapter 5: Applications for Regional and Global EnKF

In this chaper, the elements from the previous chapters will be applied to demonstrate how to run a regional and global case using the GSI observer and EnKF. These examples are intended to give users a clear idea of how to set up the GSI observer and EnKF for a particular application and properly check the run status and analysis results in order to determine if the run was successful. Note that the regional example focuses on WRF ARW, however WRF NMM and NMMB runs are similar, but require different background ensemble and namelist options. Similarly, the global example features a single global configuration (T254), however users may wish to use a different configuration, again requiring different background ensemble and namelist options.

It is assumed that the reader has successfully compiled GSI and EnKF on a local machine. For ***regional case studies***, users should have the following data available:

1. Ensemble for background
   - Ensemble files from WRF-ARW, WRF-NMM, NMM-B forecast files may be used. For this case, WRF-ARW ensemble members will be used, which are generated from the GFS ensemble, following the naming convention: wrfarw.mem00nn. Where nn corresponds to each ensemble member ('ensmean' appended for ensemble mean).
2. Conventional, GPSRO and radiance data
   - Real time GDAS and NAM PrepBUFR data can be obtained from the server: ftp://ftpprd.ncep.noaa.gov/pub/data/nccf/com/gfs/prod/ ftp://ftpprd.ncep.noaa.gov/pub/data/nccf/com/nam/prod/
3. Fixed files
   - Fixed files are located in the comGSIv3.4-EnKFv1.0/fix directory

For ***global case studies***, users should have the following data available:

1. Ensemble for background
   - Ensemble files from GFS forecast files may be used. GFS ensemble members corresponding to various spectral resolutions may be used, following the naming convention: sfg_yyyymmddhh_fhrff_mem0nn. The yyyymmddhh corresponds to the initialization date, ff corresponds to the forecast hour, and nn corresponds to each ensemble member ('ensmean' appended for ensemble mean).
2. Conventional data
   - Real time GDAS PrepBUFR (and BUFR) data can be obtained from the server: ftp://ftpprd.ncep.noaa.gov/pub/data/nccf/com/gfs/prod/
3. Fixed files
   - Fixed files are located in the comGSIv3.4-EnKFv1.0/fix/global directory

The following cases will give users an example of how to run the EnKF for a regional case with various data sources, as well as a simple global application case. Users are welcome to download these examples from the EnKF User's webpage (online case for release v1.0) or create a new background ensemble and obtain observation data from the above server.

The background ensemble files and observations used in the regional case studies are as follows:

1. Background files:
   wrfarw.mem0001 - wrfarw.mem0020, wrfarw.ensmean
   - The horizontal grid spacing is 45-km with 51 vertical sigma levels.

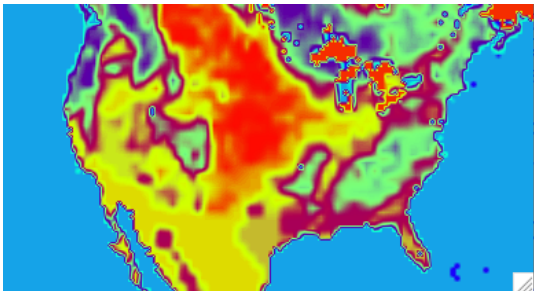

Figure 5.1: Background used for regional case study

2. Conventional, GPSRO and radiance data from 00 UTC 13 February 2014
   - Conventional file: *gdas1.t00z.prepbufr.nr*
   - GPSRO file: *gdas1.t00z.gpsro.tm00.bufr_d*
   - Radiance files: *gdas1.t00z.1bamua.tm00.bufr_d*
     gdas1.t00z.1bhrs4.tm00.bufr_d
     gdas1.t00z.1bmhs.tm00.bufr_d

3. Fixed files:
   - Fixed files are located in the GSI/EnKF community code package, under /comGSIv3.4-EnKFv1.0/fix
   - For observation control:
     convinfo: conventional data (PrepBUFR, GPSRO) info file
     ozinfo: ozone retrival info file
     satinfo: satellite radiance info file
   - Adaptive radiance bias correction (although example files are provided in the /fix directory, it is better to obtain bias correction files valid for the proper date. For this case, GFS bias correction coefficents are provided with the case data):
     satbias_angle: satellite angle dependent file
     satbias_in: satellite bias correction coefficient file

     note: *for recent cases (starting in 2015), the operational global satellite bias correction coefficient file format has changed to a single file. Please refer to GSI User's Guide for details and description of namelist changes.*

This case study was run on a linux cluster. GSI no longer (from GSI v3.2) requires btye-swapping to little endian format. BUFRLIB can automatically handle byte order issues.

For the ***regional ARW case***, assume the following locations:

Path to the background ensemble files:
`/scratch/casedata/arw_2014021300/bk/`

the path to the observations:
`/scratch/casedata/arw_2014021300/obs`

and the GSI release version 3.4/EnKF version 1.0 is located at:
`/scratch/comGSIv3.4-EnKFv1.0`

For the ***global GFS case***, assume locations as follows:

Path to background ensemble files:
`/scratch/casedata/enkf_glb_t254/bk`

the path to the observations:
`/scratch/casedata/enkf_glb_t254/obs`

and the GSI release version 3.4/EnKF version 1.0 is located at:
`/scratch/comGSIv3.4-EnKFv1.0`

5.1 Running GSI Observer for Regional Applications

5.1.1 Run Script

With both GSI and EnKF compiled and the background ensemble files and observations acquired, the next step is to work with the gsi observer run script, *run_gsi_regional.ksh*. The location of this script is under *comGSIv3.4-EnKFv1.0/run*. This run script is the same as the one used for a GSI analysis run, with a few specific options selected in order to loop through all the ensemble members and generate the ensemble observation priors for each member and the ensemble mean. In addition to the GSI observer specific options, other user-specific modifications need to be made:

- Set up batch queueing system
  To run GSI with multiple processors, a job queuing head must be added to the run_gsi_regional.ksh script. The set up of the job queue is dependent on the machine and the job control system. Refer to the GSI Users' Guide, section 3.2.2.1, for more examples of the setup section of this script. The following example is setup to run on a Linux cluster with LSF:

```
# LSF batch script to run an MPI application
#
#BSUB -P ?????????             # project code
#BSUB -W 00:30                 # wall-clock time (hrs:mins)
#BSUB -n 4                     # number of tasks in job
#BSUB -R "span[ptile=16]"      # run 16 MPI tasks per node
#BSUB -J gsi                   # job name
#BSUB -o gsi.%J.out
#BSUB -e gsi.%J.err
#BSUB -q small                 # queue
```

- Set up number of processors and the job queue system used. For this example, 'LINUX_LSF' and 4 processors are used:

```
GSIPROC=4
ARCH='LINUX_LSF'
```

- Set up the case data, analysis time, fix files, GSI executable, and CRTM coefficents:

  Set up analysis time:

```
ANAL_TIME=2014021300
```

  Set up working directory, which will hold the analysis results (all ensemble members will be run in this directory). This directory must have the proper write permissions as well as enough space to hold the output.

```
WORK_ROOT=/scratch/${user}/comGSIv3.4-EnKFv1.0/run/gsidiag_2014021300
```

  Set path to background ensemble files:

```
BK_ROOT=/scratch/${user}/casedata/arw_2014021300/bk
```

  Set file for background ensemble mean:

```
BK_FILE=${BK_ROOT}/wrfarw.ensmean
```

  Set path to the observation directory and the PrepBUFR file within the observation directory. All observations to be assimilated should reside in this directory.

```
OBS_ROOT=/scratch/${user}/casedata/arw_2014021300/obs
PREPBUFR=${OBS_ROOT}/gdas1.t00z.prepbufr.nr
```

  Set the GSI system used for this case, including the paths of the fix files and the CRTM coefficents as well as the location of the GSI executable:

```
CRTM_ROOT=/scratch/${user}/CRTM_REL-2.1.3
GSI_ROOT=/scratch/${user}/comGSIv3.4-EnKFv1.0/
FIX_ROOT=${GSI_ROOT}/fix
GSI_EXE=${GSI_ROOT}/run/gsi.exe
GSI_NAMELIST=${GSI_ROOT}/run/comgsi_namelist.sh
```

- Set which background and background error file to use

```
bk_core=ARW
bkcv_option=NAM
if_clean=clean
```

  This example uses the ARW NetCDF background; therefore `bk_core` is set to `ARW`. The regional background error covariance file is used in this case, as set by

`bkcv_option=NAM`. Finally, the run scripts are set to clean the run directory to delete all temporary intermediate files.

- Choose to run GSI observer, set up background ensemble information

```
if_observer=Yes
no_member=20
BK_FILE_mem=${BK_ROOT}/wrfarw.mem
```

The option `if_observer=Yes` is the switch that enables *run_gsi_regional.ksh* to run the GSI observer (rather than GSI analysis). In this example, 20 ensemble members are selected with the naming convention: `wrfarw.memnnnn`. Note that `memnnnn` which is assocated with each ensemble member (nnnn), is not included and will be appended later in the script.

- Link observations

```
# Link to the prepbufr data
 ln -s ${PREPBUFR} ./prepbufr

# Link to the radiance data
 ln -s ${OBS_ROOT}/gdas1.t00z.1bamua.tm00.bufr_d amsuabufr
 ln -s ${OBS_ROOT}/gdas1.t00z.1bhrs4.tm00.bufr_d hirs4bufr
 ln -s ${OBS_ROOT}/gdas1.t00z.1bmhs.tm00.bufr_d mhsbufr
 ln -s ${OBS_ROOT}/gdas1.t00z.gpsro.tm00.bufr_d gpsrobufr
```

Past the arch selection, environment variable checks, and creation of working directory, users will find the location where the observations are linked. For this case, we can see that the conventional PrepBUFR observations have been linked, as well as three different satellite radiance BUFR files (AMSU-A, HIRS4, and MHS) and a GPS RO BUFR file. These files will be linked to the working directory and separate observation innovation (diag) files will be generated for each observation.

In the run script, the proper anavinfo file is selected based on the core and background error covariance used for the case:

```
   echo ' Use NAM background error covariance'
  BERROR=${FIX_ROOT}/${BYTE_ORDER}/nam_nmmstat_na.gcv
  OBERROR=${FIX_ROOT}/nam_errtable.r3dv
  if [ ${bk_core} = NMM ] ; then
     ANAVINFO=${FIX_ROOT}/anavinfo_ndas_netcdf
  fi
  if [ ${bk_core} = ARW ] ; then
     ANAVINFO=${FIX_ROOT}/anavinfo_arw_netcdf
  fi
  if [ ${bk_core} = NMMB ] ; then
     ANAVINFO=${FIX_ROOT}/anavinfo_nems_nmmb
  fi
```

The anavinfo file used for this case is `anavinfo_arw_netcdf`, because the background is NetCDF ARW and the regional background error covariance has been selected. It is

important to check this file, located in the `./fix` directory. The number of vertical levels must match those for the background. In this example, the case has 50 vertical levels with the default value in `anavinfo_arw_netcdf` is 40. Change these values accordingly:

```
  met_guess::
!var     level    crtm_use    desc              orig_name
  ps        1      -1         surface_pressure     ps
  z         1      -1         geopotential_height  phis
  u        50       2         zonal_wind           u
  v        50       2         meridional_wind      v
  div      50      -1         zonal_wind           div
  vor      50      -1         meridional_wind      vor
  tv       50       2         virtual_temperature  tv
  q        50       2         specific_humidity    sphu
  oz       50       2         ozone                ozone
...
```

5.1.2 Run GSI observer and check run status

Once the run script is set up properly for the case and the machine, GSI can be run through the run script. The following command will submit the run:

```
$ BSUB < run_gsi_regional.ksh
```

While the job is running, move to the working directory and check the details. Given the following working directory setup:

```
WORK_ROOT=/scratch/${user}/comGSIv3.4-EnKFv1.0/run/gsidiag_2014021300
```

Go to directory `WORK_ROOT=/scratch/${user}/comGSIv3.4-EnKFv1.0/` and check the run directory. A directory named `gsidiag_2014021300` should have been created. This directory is the run directory for this GSI observer case study. The directory will be populated with many files:

```
amsua_n18.TauCoeff.bin        ssmi_f15.SpcCoeff.bin
amsua_n18.SpcCoeff.bin        ssmi_f15.TauCoeff.bin
imgr_g11.SpcCoeff.bin
imgr_g11.TauCoeff.bin
```

These files are CRTM coefficients that have been linked to this run directory through the GSI run script. Similar to running the GSI analysis, many other files are linked or copied to this run directory, such as:

`gsiparm.anl`:	GSI namelist
`prepbufr`:	PrepBUFR file for conventional observation
`convinfo`:	data usage control for conventional data
`satbias_in`:	satellite bias correction coefficient file
`satinfo`:	data usage and channel control for satellite radiance data
`berror_stats`:	background error file
`errtable`:	observation error file

Additionally, for the GSI observer many files are generated as a result of the lread_obs_save and lread_obs_skip options in the namelist for writing/reading collective observation selection information:

`obs_input.nnnn:` During ensemble mean (save=True, skip=False), save all observation preprocessing to this file. Each file (nnnn) is for a different variable type

`pennnn.obs_setup:` When looping through ensemble members (save=False, skip=True), all observation preprocessing is skipped, `obs_input.nnnn` files are read and `pennnn.obs_setup` is written for each processor (nnnn) for all observations.

As the GSI observer is running for the ensemble mean as well as looping through each member, files are for each ensemble member, as well as the ensemble mean:

`pennnn.conv_01:` Files generated after O-B, generated for conventional observations as well as one file per sensor (e.g. AMSU-A n18). These files are trimmed for resulting diag files and cleaned.

`list_run_directory:` Directory listing after ensemble mean is run, before directory is cleaned for running the ensemble members.

`list_run_directory_memnnn:` As as above, for each ensemble member (nnn).

`stdout:` Standard output file for ensemble mean.

`stdout_memnnn:` Standard output file for each ensemble member (nnn).

The presence of the standard output files in the directory suggest the GSI observer run scripts have successfully set up a run environment for the GSI observer, properly looping through the ensemble members, and the GSI executable is running on each ensemble member. Once GSI have finished running, diag files should be generated for each observation type for each member as well as the ensemble mean:

```
diag_conv_ges.ensmean            diag_conv_ges.mem001
diag_conv_ges.mem002             diag_conv_ges.mem003
diag_conv_ges.mem004             diag_conv_ges.mem005
…                                diag_conv_ges.mem020

diag_amsua_metop-a_ges.ensmean   diag_amsua_metop-a_ges.mem001
diag_amsua_metop-a_ges.mem002    diag_amsua_metop-a_ges.mem003
diag_amsua_metop-a_ges.mem004    diag_amsua_metop-a_ges.mem005
…                                diag_amsua_metop-a_ges.mem020

diag_amsua_n18_ges.ensmean       diag_amsua_n18_ges.mem001
diag_amsua_n18_ges.mem002        diag_amsua_n18_ges.mem003
```

```
diag_amsua_n18_ges.mem004        diag_amsua_n18_ges.mem005
…                                diag_amsua_n18_ges.mem020

diag_hirs4_n19_ges.ensmean       diag_amsua_n15_ges.mem001
diag_hirs4_n19_ges.mem002        diag_amsua_n15_ges.mem003
diag_hirs4_n19_ges.mem004        diag_amsua_n15_ges.mem005
…                                diag_amsua_n15_ges.mem020
```

5.1.3 Check for successful GSI completion

The presence of the diag files and standard out files for the ensemble mean and each member indicates that the GSI observer has run without crashing, but does not necessary indicate a successful analysis. It is important to check the stdout files in the run directory to make sure the GSI observer completed each step without any obvious problems. The following are several important areas of the standard out file to check:

1. Read in the anavinfo and namelist

```
 state_vectors*init_anasv:   2D-STATE VARIABLES ps          sst
 state_vectors*init_anasv:  3D-STATE VARIABLES u           v
 tv          tsen        q           oz          cw          prse
 state_vectors*init_anasv:  ALL STATE VARIABLES u           v
 tv          tsen        q           oz          cw          prse
 ps          sst
      ...
&SETUP
 GENCODE =   78.0000000000000     ,
 FACTQMIN        =  0.000000000000000E+000,
 FACTQMAX        =  0.000000000000000E+000,
     ...
```

2. Read in the background field

The following lines in stdout immediately following the namelist section, indicate that GSI is reading the background fields. Checking that the range of the max and min values will indicate if particular background fields are normal.

```
  dh1  =            3
  iy,m,d,h,m,s=        2014           2          13           0
0
           0
  dh1  =            3
 rmse_var = SMOIS
 ndim1 =           3
 ordering = XYZ
 staggering =  N/A
 start_index =           1           1           1           0
 end_index =         129          70           4           0
 WrfType =         104
 ierr  =           0
   ...
rmse_var = T ndim1=           3
  WrfType =         104  WRF_REAL=         104 ierr  =           0
  ordering = XYZ staggering =  N/A
  start_index =           1           1           1           0
end_index =
```

```
        129          70           50           0
  k,max,min,mid T=           1   312.7004       238.3343       273.0790
  k,max,min,mid T=           2   312.8147       238.8450       273.3524
  k,max,min,mid T=           3   313.0251       239.7868       273.8565
  k,max,min,mid T=           4   313.3909       241.4314       274.7360
  k,max,min,mid T=           5   313.9359       243.8975       276.0528
  k,max,min,mid T=           6   314.6469       247.1448       277.7831
  k,max,min,mid T=           7   314.9745       250.4735       280.1970
  k,max,min,mid T=           8   315.3399       251.3270       283.2472
  k,max,min,mid T=           9   315.4477       252.4649       286.3434
  k,max,min,mid T=          10   315.6670       253.5486       288.9474
  k,max,min,mid T=          11   316.5586       256.9481       291.7451
  k,max,min,mid T=          12   317.7124       260.9537       294.6829
  k,max,min,mid T=          13   320.3354       265.7097       296.8029
  k,max,min,mid T=          14   322.1675       270.8215       298.5765
  k,max,min,mid T=          15   323.3958       274.9277       300.1816
  k,max,min,mid T=          16   324.0006       278.8837       302.2858
  k,max,min,mid T=          17   324.9014       282.7533       305.0905
  k,max,min,mid T=          18   325.9820       285.7436       307.1180
  k,max,min,mid T=          19   327.2946       287.9378       309.9308
  k,max,min,mid T=          20   328.1814       289.9290       312.5236
   …
```

3. Read in observational data

The `stdout` (ensemble mean) and `stdout_memnnn` (ensemble members) contains distinct differences for the reading in the observational data controlled by the `lread_obs_save` and `lread_obs_skip` options in the namelist.

a. Ensemble mean

```
read_obs_check: bufr file cris      npp        not available crisbufr
read_obs_check: bufr file date is   2014021300 prepbufr q
read_obs_check: bufr file date is   2014021300 gpsrobufr gps_ref
read_obs_check: bufr file date is   2014021300 amsuabufr amsua     n18
...
READ_OBS:  read    6 uv         uv         using ntasks=   1   6   1
READ_OBS:  read   10 sst        sst        using ntasks=   1   7   1
READ_OBS:  read   11 gps_ref    gps        using ntasks=   1   0   1
READ_OBS:  read   21 hirs4      hirs4_n19  using ntasks=   1   1   1
...
READ_BUFRTOVS:  file=hirs4bufr  type=hirs4    sis=hirs4_metop-a    nread=
11780 ithin= 2 rmesh=  60.000000 isfcalc= 0 ndata=    6992 ntask=  2
...
READ_GPS : file=gpsrobufr       type=gps_ref    sis=gps       nread=
4910 ithin= 0 rmesh= 120.000000 isfcalc= 0 ndata=      2387 ntask=  1
...
READ_PREPBUFR: file=prepbuf   type=t    sis=t     nread=     23460 ithin=
0 rmesh= 120.000000 isfcalc= 0 ndata=     23009 ntask=  1
...
READ_OBS:  write collective obs selection info to obs_input.common
```

For the ensemble mean, the same procedure for observation processing as the GSI analysis is followed (`read_obs_check`, `READ_OBS`, `READ_BUFRTOVS`, `READ_GPS`,

READ_PREPBUFR …). Finally, it is indicated that the collective observations selection information is written out for use with the ensemble memebers.

b. Ensemble members
For the stdout_memnnn, the observation processing sets are skipped and observation information is read in from the ensemble mean:

```
OBSERVER_SET:  read collective obs selection info from obs_input.common
```

Finally, for both the ensemble mean and members, the following lines will appear:

```
OBS_PARA: ps    102   1003   1015   2289   2867   2339   3250   3725

OBS_PARA: t     222   1938   1665   3307   4215   3070   4169   5211

OBS_PARA: q     176   1506   1381   2691   3997   2738   3484   4051

OBS_PARA: pw     14    225     95    155    111     44    106    136

OBS_PARA: uv    241   1937   1979   3292   4829   3403   5705   5273

OBS_PARA: gps_ref   174    300    0   300    776    546    210   461
```

This table is important to check if the observations have been read in, which types of observations have been read in, and the distribution of observations in each subdomain.

4. Indication that the GSI observer has successfully run:

```
observer_final: successfully finalized
glbsoi: complete
000]gsisub(): : complete.

    ENDING DATE-TIME    JUL 22,2015  17:57:15.350  203  WEN   2457226
    PROGRAM GSI_ANL HAS ENDED.
* . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * . *
```

After looking over each section of the standard output files, it can be concluded that the GSI observer ran without issues. Note that because the outer loop was set to 0 (miter=0) for the GSI observer, no minimization occurred and therefore no analysis results were produced.

5.2 Running EnKF for Regional Applications

5.1.1 Run Script

Once the GSI observer has been run successfully, the next step is to setup the EnKF run script, *run_enkf_wrf.ksh*. This script is located under *comGSIv3.4-EnKFv1.0/run*. This run script uses the diag files generated by the GSI observer script as observation input, and generates the EnKF analysis. Similar to the GSI observer script, several user-specific modifications need to be made:

- Set up batch queueing system

```
#BSUB -P ?????????             # project code
#BSUB -W 00:30                 # wall-clock time (hrs:mins)
#BSUB -n 32                    # number of tasks in job
#BSUB -R "span[ptile=16]"      # run 16 MPI tasks per node
#BSUB -J enkf                  # job name
#BSUB -o enkf.%J.out           # output file name
#BSUB -e enkf.%J.err           # error file
```

- Set up number of processors and the job queue system used:

```
GSIPROC=32
ARCH='LINUX_LSF'
```

One difference from the GSI observer script is that the number ls of processors used should be greater than at least the number of ensemble members for running the EnKF. In this case we have 20 ensemble members and have requested 32 cores.

- Set up the analysis time, fixed files, EnKF executable

```
ANAL_TIME=2014021300
WORK_ROOT/scratch/${user}/GSIv3.4-EnKFv1.0/run/enkf_2014021300
diag_ROOT=/scratch/${user}/GSIv3.4-EnKFv1.0/run/gsidiag_2014021300
BK_ROOT=/scratch/${user}/casedata/arw_2014021300/bk
BK_FILE=${BK_ROOT}/wrfarw.ensmean
GSI_ROOT=/scratch/${user}/GSIv3.4-EnKFv1.0
FIX_ROOT=${GSI_ROOT}/fix
ENKF_EXE=${GSI_ROOT}/src/main/enkf/wrf_enkf
CRTM_ROOT=/scratch/${user}/CRTM_REL-2.1.3
ENKF_NAMELIST=${GSI_ROOT}/run/enkf_wrf_namelist.sh
```

- Set background file and location of the diag files (from GSI observer)

```
diag_ROOT=/scratch/${user}/GSIv3.4-EnKFv1.0/run/gsidiag_2014021300
BK_FILE=${BK_ROOT}/wrfarw.ensmean
```

Two modifications to note for the EnKF are the `diag_ROOT` and `BK_FILE`. The `diag_ROOT` points to the working directory where the GSI observer run. This directory contains diag* files which will be linked to the EnKF working directory. `BK_FILE` points to the background ensemble mean.

- Select ensemble parameters

```
 NMEM_ENKF=20
 BK_FILE_mem=${BK_ROOT}/wrfarw.mem
 NLONS=129
 NLATS=70
 NLEVS=50
 IF_ARW=.true.
 IF_NMM=.false.
```

The next section sets information about the ensemble, including the number of member, background ensemble members, domain specifications, dynamical core, and list of observations to be used in the EnKF. Note that the ensemble members (`BK_FILE_mem`) do not include the 3-digit member number. This will be appended to the name specified for this field later in the script.

- Select observations

```
list="conv amsua_n18 amsua_metop-a amsua_n15"
```

The previous line is contained with the ensemble parameter section. For the observations to be assimilated, be aware that only observations/platforms that have been run through the GSI observer (and therefore a diag* file exists) are valid.

5.1.2 Run EnKF and check run status

Once the run script is set up properly for the case and the machine, EnKF can be run through the run script. The following command will submit the run:

```
$ BSUB < run_enkf_wrf.ksh
```

While the job is running, move to the working directory and check the details. Given the following working directory setup:

```
WORK_ROOT=/scratch/${user}/comGSIv3.4-EnKFv1.0/run/enkf_2014021300
```

Go to directory `WORK_ROOT=/scratch/${user}/comGSIv3.4-EnKFv1.0/` and check the run directory. A directory named `enkf_2014021300` should have been created. This directory is the run directory for this EnKF case study. The directory will be populated with many links:

```
diag_conv_ges.ensmean              diag_conv_ges.mem001
diag_conv_ges.mem002               diag_conv_ges.mem003
diag_conv_ges.mem004               diag_conv_ges.mem005
…                                  diag_conv_ges.mem020

diag_amsua_metop-a_ges.ensmean     diag_amsua_metop-a_ges.mem001
diag_amsua_metop-a_ges.mem002      diag_amsua_metop-a_ges.mem003
diag_amsua_metop-a_ges.mem004      diag_amsua_metop-a_ges.mem005
…                                  diag_amsua_metop-a_ges.mem020

diag_amsua_n18_ges.ensmean         diag_amsua_n18_ges.mem001
diag_amsua_n18_ges.mem002          diag_amsua_n18_ges.mem003
diag_amsua_n18_ges.mem004          diag_amsua_n18_ges.mem005
…                                  diag_amsua_n18_ges.mem020

diag_hirs4_n19_ges.ensmean         diag_amsua_n15_ges.mem001
diag_hirs4_n19_ges.mem002          diag_amsua_n15_ges.mem003
diag_hirs4_n19_ges.mem004          diag_amsua_n15_ges.mem005
…                                  diag_amsua_n15_ges.mem020
```

The diag files are linked from the GSI observer working directory. These links are specified in the run script. Note that ozone diag files (gome, omi, sbuv) will appear in the directory without links (zero length files) if ozone diags are not present from the GSI observer.

```
firstguess.ensmean
firstguess.mem001       firstguess.mem002
firstguess.mem003       firstguess.mem004
firstguess.mem005       firstguess.mem006
firstguess.mem007       firstguess.mem008
firstguess.mem009       firstguess.mem010
firstguess.mem011       firstguess.mem012
firstguess.mem013       firstguess.mem014
firstguess.mem015       firstguess.mem016
firstguess.mem017       firstguess.mem018
firstguess.mem019       firstguess.mem020
```

The first guess files are also linked into the directory from the run script, pointing to the path of the ensemble mean and ensemble members designated in the setup section.

Similar to running the GSI observer, many other static files are linked or copied to this run directory, such as:

`enkf.nml`:	EnKF namelist
`convinfo`:	data usage control for conventional data
`satbias_in`:	satellite bias correction coefficient file
`satbias_angle`:	satellite bias correction angle file
`satinfo`:	data usage and channel control for satellite radiance data
`ozinfo`:	data usage control from ozone data
`covinflate.dat`:	Three-dimensional multiplicative inflation factor fields
`stdout`:	EnKF standard output file

The presence of the standard output file in the directory suggest the EnKF run script has successfully set up a run environment for the EnKF, properly linking the first guess and diag files from the GSI observer, and the EnKF executable is running. Once EnKF has finished running, analysis files should be generated for each member as well as the ensemble mean:

```
analysis.ensmean
analysis.mem001         analysis.mem002
analysis.mem003         analysis.mem004
analysis.mem005         analysis.mem006
analysis.mem007         analysis.mem008
analysis.mem009         analysis.mem010
analysis.mem011         analysis.mem012
analysis.mem013         analysis.mem014
analysis.mem015         analysis.mem016
analysis.mem017         analysis.mem018
analysis.mem019         analysis.mem020
```

5.1.3 Check for successful EnKF completion

The presence of the analysis files for the ensemble mean and each member as well as the standard out file indicates that the EnKF has run without crashing, but does not necessary indicate a successful analysis. It is important to check the stdout file in the run directory to make sure the EnKF completed each step without any obvious problems. The following are several important areas of the standard out file to check:

1. Check namelist has been properly read in and configuration is correct:

```
namelist parameters:
 --------------------
 &NAM_ENKF
 DATESTRING      = 2014021300,
 DATAPATH        = ./
 IASSIM_ORDER    =  ls
        0,
 COVINFLATEMAX   =   100.0000    ,
 COVINFLATEMIN   =   1.000000    ,
 DETERMINISTIC   = T,
 SORTINC = T,
 CORRLENGTHNH    =   500.0000    ,
```

2. Check analysis time, number of ensemble members, as well as the actual analysis variables and the background type. The maximum and minimum values for surface pressure are printed for a sanity check:

```
analysis time 2014021300
          20  members
 first-guess forecast hour for analysis = 06
           5 3d vars to update
 total of         251  2d grids will be updated (including ps)
 using multiplicative inflation based on Pa/Pb
 Vars in Rad-Jacobian (dims)
 --------------------------
 sst                             0
 Updating U, V, T, QVAPOR, PH, and MU for WRF-ARW...
 Surface pressure (spressmn) min/max range:   678.673339843750
   1032.22473144531
```

3. Statistics of the ensemble prior

After many lines describing the bias correction coefficents and the inventory of the observation number and types from the input diag* files, as well as the time spent reading in each observation type, statistics of the ensemble priors' fit to all observations are provided for each region (NH, SH, TR):

```
innovation statistics for prior:
 conventional obs
 region, obtype, nobs, bias, innov stdev, sqrt(S+R), sqrt(S), sqrt(R):
NH  all ps 14083 -0.591E-02  0.860E+00  0.190E+01  0.601E+00  0.180E+01
TR  all ps    50 -0.629E-01  0.422E+00  0.115E+01  0.367E+00  0.109E+01
NH  all t   7766 -0.228E+00  0.163E+01  0.936E+01  0.480E+00  0.935E+01
```

```
TR  all t    138 -0.130E+00   0.118E+01  0.184E+01  0.234E+00  0.183E+01
NH  all uv  21680  0.136E-01  0.295E+01  0.504E+01  0.140E+01  0.484E+01
TR  all uv    302 -0.379E+00  0.251E+01  0.419E+01  0.124E+01  0.401E+01
NH  all q    2553 -0.323E-01  0.146E+00  0.153E+04  0.868E-01  0.153E+04
TR  all q      68 -0.721E-01  0.202E+00  0.299E+00  0.582E-01  0.293E+00
satellite brightness temp
instrument, channel #,nobs, bias, innov stdev, sqrt(S+R),sqrt(S),sqrt(R):
amsua_n18 1  636   0.152E+00  0.200E+01  0.134E+02  0.163E+01  0.133E+02
amsua_n18 2  637  -0.519E-01  0.228E+01  0.144E+02  0.687E+00  0.144E+02
amsua_n18 3  648  -0.179E+01  0.239E+01  0.873E+01  0.399E+00  0.872E+01
amsua_n18 4  708  -0.181E+00  0.481E+00  0.265E+01  0.122E+00  0.264E+01
amsua_n18 5  708  -0.332E-01  0.291E+00  0.832E+00  0.957E-01  0.827E+00
amsua_n18 6  1606 -0.500E-01  0.197E+00  0.351E+00  0.586E-01  0.346E+00
amsua_n18 7  1668 -0.146E+00  0.289E+00  0.329E+00  0.500E-01  0.325E+00
amsua_n18 8  1636 -0.621E-01  0.313E+00  0.360E+00  0.668E-01  0.354E+00
amsua_n18 10 1357  0.574E+00  0.620E+00  0.525E+00  0.436E-01  0.523E+00
amsua_n18 11 164   0.634E+00  0.651E+00  0.705E+00  0.573E-01  0.703E+00
amsua_n18 15 396  -0.102E+01  0.263E+01  0.121E+02  0.168E+01  0.120E+02
```

This table should be checked in order to determine if the inflation is appropriate. The goal is to make the total ensemble spreads of priors (*sqrt*(S+R) match the innovations (*innov std)* as much as possible. We can see in certain regions, particularly in the NH the differences are noteable. These differences are associated with higher observation error (*sqrt*(R)). Users should consider tuning of inflation and localization, which is typically determined using cases with multiple assimilation cycles. Refer to section 4.2 of this User's Guide for more information on tuning.

4. Domain and observation partition:

```
 npts =         9030
 min/max number of points per proc =          250         320
 time to do model space decomp =   3.016927279531956E-003
 nobsgood =        56804
 min/max number of obs per proc =         1775        1776
 time to do ob space decomp =   6.784386932849884E-004
 sending out observation prior ensemble perts from root ...
 ... took   1.752869039773941E-002  secs
    time in load_balance =  0.100384402088821      on proc           0
```

The analysis variables and the observations are distributed to different processors. We can see in this case that the min/max number of points per processor are 250 and 320, respectively. Similarly, we can see that the min/max number of observations per processor are 1775 and 1776, respectively, indicating that the observations are well dispersed among the processors.

```
 READGRIDDATA_ARW: U           1  -20.55222       20.72632
 READGRIDDATA_ARW: U           2  -20.87878       21.20316
 READGRIDDATA_ARW: U           3  -21.61046       22.36177
 READGRIDDATA_ARW: U           4  -22.62561       24.08465
 READGRIDDATA_ARW: U           5  -24.07165       26.47827
```

Additionally, check the minimum and maximum values of the fields at each vertical level as a quick sanity check.

5.1.4 Diagnose EnKF analysis results

At the bottom of the standard output file, there are several output statistics and tables that are helpful for users to diagnose the quality of the EnKF analysis.

5. Statistics of the ensemble analysis

```
 min/max var 1 inflation =    1.000000       7.171966
 min/max var 2 inflation =    1.000000       7.262142
 min/max var 3 inflation =    1.000000       6.592534
 min/max spfh inflation =     1.000000       4.910290
 min/max ps inflation =       1.000000       8.952715
 global ps prior std. dev min/max =    21.52301       172.3270

 NH mean ps prior standard deviation =    57.82042
 NH mean ps posterior standard deviation (before inflation)=    29.29152
 NH mean ps posterior standard deviation (after inflation) =    54.58800
 NH mean ps inflation =    2.433801
 TR mean ps prior standard deviation =    31.09247
 TR mean ps posterior standard deviation (before inflation)=    28.32949
 TR mean ps posterior standard deviation (after inflation) =    30.79514
 TR mean ps inflation =    1.105389
 time in inflate_ens =  5.486726248636842E-002 on proc               0
```

This section is important for checking whether the values of the inflation are reasonable by looking at a summary of the maximum and minimum values. Also, viewing the regional averaged statistics before and after inflation. For this case we can see both the NH and TR show larger standard deviation after the inflation, which is more consistent with the prior standard deviation.

6. Spread inflation of the analysis ensemble

```
innovation statistics for posterior:
 conventional obs
 region, obtype, nobs, bias, innov stdev, sqrt(S+R), sqrt(S), sqrt(R):
NH all ps 14083  0.130E-02  0.755E+00  0.181E+01  0.143E+00  0.180E+01
TR all ps    50 -0.530E-01  0.399E+00  0.111E+01  0.195E+00  0.109E+01
NH all t   7766 -0.777E-01  0.145E+01  0.935E+01  0.262E+00  0.935E+01
TR all t    138 -0.107E+00  0.115E+01  0.184E+01  0.199E+00  0.183E+01
NH all uv 21680  0.409E-01  0.267E+01  0.488E+01  0.653E+00  0.484E+01
TR all uv   302 -0.261E+00  0.236E+01  0.406E+01  0.636E+00  0.401E+01
NH all q   2553 -0.174E-01  0.114E+00  0.153E+04  0.445E-01  0.153E+04
TR all q     68 -0.280E-01  0.168E+00  0.296E+00  0.413E-01  0.293E+00
 satellite brightness temp
instrument,channel #,nobs,bias,innov stdev, sqrt(S+R), sqrt(S), sqrt(R):
 amsua_n18 1   636  0.165E-01  0.174E+01  0.133E+02  0.761E+00  0.133E+02
 amsua_n18 2   637 -0.892E-01  0.221E+01  0.144E+02  0.326E+00  0.144E+02
 amsua_n18 3   648 -0.180E+01  0.238E+01  0.872E+01  0.206E+00  0.872E+01
 amsua_n18 4   708 -0.157E+00  0.445E+00  0.264E+01  0.765E-01  0.264E+01
 amsua_n18 5   708  0.204E-02  0.262E+00  0.829E+00  0.613E-01  0.827E+00
 amsua_n18 6  1606 -0.172E-01  0.160E+00  0.348E+00  0.418E-01  0.346E+00
 amsua_n18 7  1668 -0.121E+00  0.238E+00  0.327E+00  0.338E-01  0.325E+00
 amsua_n18 8  1636 -0.541E-01  0.256E+00  0.356E+00  0.442E-01  0.354E+00
 amsua_n18 10 1357  0.548E+00  0.594E+00  0.524E+00  0.382E-01  0.523E+00
```

```
amsua_n18 11  164  0.618E+00  0.637E+00  0.705E+00  0.539E-01  0.703E+00
amsua_n18 15  396 -0.115E+01  0.232E+01  0.120E+02  0.797E+00  0.120E+02
```

After the EnKF analysis, it is important to check the innovation statistics, much like the prior' fit to all observation.

5.3 Running GSI Observer for Global Applications

5.3.1 Run Script

With both GSI and EnKF compiled (global executable for EnKF) and the background ensemble files and observations acquired, the next step is to work with the gsi observer run script, *run_gsi_global.ksh*. The location of this script is under *comGSIv3.4-EnKFv1.0/run*. This run script is the same as the one used for a GSI global analysis run, with a few specific options selected in order to loop through all the ensemble members and generate the ensemble observation priors for each member and the ensemble mean. For the global examples, users must choose a case that corresponds to the spectral resolution of the background ensemble. For this example, we will be using T254. In addition to the GSI observer specific options, other user-specific modifications need to be made:

- Set up batch queueing system
  To run GSI with multiple processors, a job queuing head must be added to the run_gsi_global.ksh script. The set up of the job queue is dependent on the machine and the job control system. Refer to the GSI Users' Guide, section 3.2.2.1, for more examples of the setup section of this script. The following example is setup to run on a Linux cluster with LSF:

```
# LSF batch script to run an MPI application
#
#BSUB -P ?????????            # project code
#BSUB -W 00:30                # wall-clock time (hrs:mins)
#BSUB -n 24                    # number of tasks in job
#BSUB -R "span[ptile=16]"     # run 16 MPI tasks per node
#BSUB -J gsi                  # job name
#BSUB -o gsi.%J.out
#BSUB -e gsi.%J.err
#BSUB -q regular              # queue
```

- Set up number of processors and the job queue system used. For this example, 'LINUX_LSF' and 24 processors are used:

```
GSIPROC=24
ARCH='LINUX_LSF'
```

- Set up the case data, analysis time, fix files, GSI executable, and CRTM coefficents:

  Set up analysis time, guess time and select global case:

```
ANAL_TIME= 2014040512
GUESS_TIME = 2014040506
GFSCASE=enkf_glb_t254
```

Set up working directory, which will hold the analysis results (all ensemble members will be run in this directory). This directory must have the proper write permissions as well as enough space to hold the output.

```
WORK_ROOT=/scratch/${user}/comGSIv3.4-EnKFv1.0/run/gsidiag_${GFS_CASE}
```

Set path to background ensemble files:

```
BK_ROOT=/scratch/${user}/casedata/enkf_glb_t254/bk
```

Set path to the observation directory and the PrepBUFR file within the observation directory. All observations to be assimilated should reside in this directory.

```
OBS_ROOT=/scratch/${user}/casedata/enkf_glb_t254/obs
PREPBUFR=${OBS_ROOT}/gdas1.t12z.prepbufr.nr
```

Set the GSI system used for this case, including the paths of the fix files and the CRTM coefficents as well as the location of the GSI executable:

```
CRTM_ROOT=/scratch/${user}/CRTM_REL-2.1.3
GSI_ROOT=/scratch/${user}/comGSIv3.4-EnKFv1.0/
FIX_ROOT=${GSI_ROOT}/fix
GSI_EXE=${GSI_ROOT}/run/gsi.exe
GSI_NAMELIST=${GSI_ROOT}/run/comgsi_namelist_gfs.sh
```

Note that the GSI namelist generation script used for the global case (comgsi_namelist_gfs.sh) is different from the one used for the regional example.

- Choose to run GSI observer, set up background ensemble information

```
if_observer=Yes
no_member=10
BK_FILE_mem=${BK_ROOT}/sfg_${GUESS_DATE}
```

The option `if_observer=Yes` is the switch that enables *run_gsi_global.ksh* to run the GSI observer (rather than GSI analysis). In this example, 10 ensemble members are selected with the naming convention: `sfg_${GUESS_DATE}_memnnnn`. Note that `_memnnnn` which is assocated with each ensemble member (nnnn), is not included and will be appended later in the script.

- Set the JCAP resolution for the case:

```
if [[ "$GFSCASE" = "T62" ]]; then
  JCAP=62
  JCAP_B=62
elif [[ "$GFSCASE" = "T126" ]]; then
  JCAP=126
  JCAP_B=126
elif [[ "$GFSCASE" = "enkf_glb_t254" ]]; then
  JCAP=254
  JCAP_B=254
elif [[ "$GFSCASE" = "T254" ]]; then
  JCAP=254
```

```
  JCAP_B=574
elif [[ "$GFSCASE" = "T574" ]]; then
  JCAP=574
  JCAP_B=1534
else
   echo "INVALID case = $GFSCASE"
   exit
fi
  LEVS=64
```

Note that this selection is filled based on selection of the GFSCASE. This example used the enkf_glb_t254 case. All cases use 64 levels.

Further in the run script *run_gsi_global.ksh*, the resolution parameters are set based on the requested case (resolution). For this case, we are using JCAP=254:

```
elif [[ "$JCAP" = "254" ]]; then
   LONA=512
   LATA=256
   DELTIM=1200
   resol=2
```

- Link satellite bias correction coefficents, background ensemble files, and observations:

```
if [[ "$GFSCASE" = "enkf_glb_t254" ]]; then
  cp $OBS_ROOT/gdas1.t12z.abias      ./satbias_in
  cp $OBS_ROOT/gdas1.t12z.satang     ./satbias_angle

  cp $BK_ROOT/sfcanl_${GUESS_TIME}_fhr03_ensmean  ./sfcf03
  cp $BK_ROOT/sfcanl_${GUESS_TIME}_fhr06_ensmean  ./sfcf06
  cp $BK_ROOT/sfcanl_${GUESS_TIME}_fhr06_ensmean  ./sfcf09

  cp $BK_ROOT/sfg_${GUESS_TIME}_fhr03_mem001    ./sigf03
  cp $BK_ROOT/sfg_${GUESS_TIME}_fhr06_mem001    ./sigf06
  cp $BK_ROOT/sfg_${GUESS_TIME}_fhr09_mem001    ./sigf09
...
# Link to the prepbufr data
 ln -s ${PREPBUFR} ./prepbufr

# Link to the radiance data
if [[ "$GFSCASE" = "enkf_glb_t254" ]]; then
  obsfile_amua=gdas1.t12z.1bamua.tm00.bufr_d
  obsfile_amub=gdas1.t12z.1bamub.tm00.bufr_d
else
```

Past the arch selection, environment variable checks, and creation of working directory, users will find the location where the observations are linked. For this case, we can see that the conventional PrepBUFR observations have been linked, as well as AMSU-A and AMSU-B satellite radiance BUFR files. These files will be linked to the working directory and separate observation innovation (diag) files will be generated for each observation.

5.3.2 Run GSI observer and check run status

Once the run script is set up properly for the case and the machine, GSI can be run through the run script. The following command will submit the run:

```
$ BSUB < run_gsi_global.ksh
```

While the job is running, move to the working directory and check the details. Given the following working directory setup:

```
WORK_ROOT=/scratch/${user}/comGSIv3.4-EnKFv1.0/run/gsidiag_${GFSCASE}
```

Go to directory `WORK_ROOT=/scratch/${user}/comGSIv3.4-EnKFv1.0/` and check the run directory. A directory named `gsidiag_enkf_glb_t254` should have been created. This directory is the run directory for this GSI observer case study. The directory will be populated with many files such as:

```
amsua_n18.TauCoeff.bin        ssmi_f15.SpcCoeff.bin
amsua_n18.SpcCoeff.bin        ssmi_f15.TauCoeff.bin
imgr_g11.SpcCoeff.bin
imgr_g11.TauCoeff.bin
```

These files are CRTM coefficients that have been linked to this run directory through the GSI run script. Similar to running the GSI analysis, many other files are linked or copied to this run directory, such as:

`gsiparm.anl`:	GSI namelist
`prepbufr`:	PrepBUFR file for conventional observation
`convinfo`:	data usage control for conventional data
`satbias_in`:	satellite bias correction coefficient file
`satinfo`:	data usage and channel control for satellite radiance data
`berror_stats`:	background error file
`errtable`:	observation error file

Additionally, for the GSI observer many files are generated as a result of the lread_obs_save and lread_obs_skip options in the namelist for writing/reading collective observation selection information:

`obs_input.nnnn`: During ensemble mean (save=True, skip=False), save all observation preprocessing to this file. Each file (nnnn) is for a different variable type

`pennnn.obs_setup`: When looping through ensemble members (save=False, skip=True), all observation preprocessing is skipped, `obs_input.nnnn` files are read and `pennnn.obs_setup` is written for each processor (nnnn) for all observations.

As the GSI observer is running for the ensemble mean as well as looping through each member, files are for each ensemble member, as well as the ensemble mean:

`pennnn.conv_01:`	Files generated after O-B, generated for conventional observations as well as one file per sensor (e.g. AMSU-A n18). These files are trimmed for resulting diag files and cleaned.
`list_run_directory:`	Directory listing after ensemble mean is run, before directory is cleaned for running the ensemble members.
`list_run_directory_memnnn:`	As as above, for each ensemble member (nnn).
`stdout:`	Standard output file for ensemble mean.
`stdout_memnnn:`	Standard output file for each ensemble member (nnn).

The presence of the standard output files in the directory suggest the GSI observer run scripts have successfully set up a run environment for the GSI observer, properly looping through the ensemble members, and the GSI executable is running on each ensemble member. Once GSI has finished running, diag files should be generated for each observation type for each member as well as the ensemble mean:

```
diag_conv_ges.ensmean            diag_conv_ges.mem001
diag_conv_ges.mem002             diag_conv_ges.mem003
diag_conv_ges.mem004             diag_conv_ges.mem005
…                                diag_conv_ges.mem010

diag_amsua_metop-a_ges.ensmean   diag_amsua_metop-a_ges.mem001
diag_amsua_metop-a_ges.mem002    diag_amsua_metop-a_ges.mem003
diag_amsua_metop-a_ges.mem004    diag_amsua_metop-a_ges.mem005
…                                diag_amsua_metop-a_ges.mem010

diag_amsua_n18_ges.ensmean       diag_amsua_n18_ges.mem001
diag_amsua_n18_ges.mem002        diag_amsua_n18_ges.mem003
diag_amsua_n18_ges.mem004        diag_amsua_n18_ges.mem005
…                                diag_amsua_n18_ges.mem010

diag_hirs4_n19_ges.ensmean       diag_amsua_n15_ges.mem001
diag_hirs4_n19_ges.mem002        diag_amsua_n15_ges.mem003
diag_hirs4_n19_ges.mem004        diag_amsua_n15_ges.mem005
…                                diag_amsua_n15_ges.mem010
```

5.3.3 Check for successful GSI completion

The presence of the diag files and standard out files for the ensemble mean and each member indicates that the GSI observer has run without crashing, but does not necessary indicate a successful analysis. It is important to check the stdout files in the run directory to make sure the GSI observer completed each step without any obvious problems. The following are several important areas of the standard out file to check:

5. Read in the anavinfo and namelist

```
READ_FILES:  analysis date,minutes         2014           4           5
          12           0    19070640
 gsi_metguess_mod*init_:  2D-MET STATE VARIABLES:
 ps
 z
 gsi_metguess_mod*init_:  3D-MET STATE VARIABLES:
 u
 v
      ...
&SETUP
 GENCODE =   78.0000000000000     ,
 FACTQMIN        =  0.000000000000000E+000,
 FACTQMAX        =  0.000000000000000E+000,
     ...
```

6. Read in the background field

The following lines in stdout immediately following the namelist section, indicate that GSI is reading the background fields. Checking that the range of the max and min values will indicate if particular background fields are normal.

```
GESINFO:  Read NCEP sigio format file, sigf06
GESINFO:  jcap_b= 254, levs= 64, latb=  256, lonb=  512, ntrac=  3,
ncldt=  1, idvc=  2, nvcoord=  2, idvm=  0, idsl=  0, idpsfc=  0, idthrm=
0
   k,ak,bk,ck,tref=  1   0.00000000000       1.00000000000
0.00000000000       300.000000000
   k,ak,bk,ck,tref=  2   0.00000000000      0.994671165943
0.00000000000       300.000000000
   k,ak,bk,ck,tref=  3  0.574999988079E-03  0.988626599312
0.00000000000       300.000000000
   k,ak,bk,ck,tref=  4  0.574100017548E-02  0.981742262840
0.00000000000       300.000000000
   k,ak,bk,ck,tref=  5  0.215160007477E-01  0.973867595196
0.00000000000       300.000000000
   k,ak,bk,ck,tref=  6  0.557120018005E-01  0.964827597141
0.00000000000       300.000000000
   k,ak,bk,ck,tref=  7  0.116899002075      0.954434096813
0.00000000000       300.000000000
   k,ak,bk,ck,tref=  8  0.214014999390      0.942491054535
0.00000000000       300.000000000
   k,ak,bk,ck,tref=  9  0.356222991943      0.928797304630
0.00000000000       300.000000000
   k,ak,bk,ck,tref= 10  0.552719970703      0.913151025772
0.00000000000       300.000000000
   k,ak,bk,ck,tref= 11  0.812489013672      0.895354986191
0.00000000000       300.000000000
   ...
```

7. Read in observational data

The `stdout` (ensemble mean) and `stdout_memnnn` (ensemble members) contains distinct differences for the reading in the observational data controlled by the `lread_obs_save` and `lread_obs_skip` options in the namelist.

c. Ensemble mean

```
read_obs_check: bufr file date is      2014040512 amsuabufr amsua n15
read_obs_check: bufr file date is      2014040512 prepbufr uv
read_obs_check: bufr file date is  2014040512 prepbufr ps
read_obs_check: bufr file date is  2014040512 prepbufr t
read_obs_check: bufr file date is  2014040512 prepbufr q
read_obs_check: bufr file date is  2014040512 amsuabufr amsua n19
...
READ_OBS:  read    6 uv         uv         using ntasks=   1   6   1
READ_OBS:  read   10 sst        sst        using ntasks=   1   7   1
READ_OBS:  read   11 gps_ref    gps        using ntasks=   1   0   1
READ_OBS:  read   21 hirs4      hirs4_n19  using ntasks=   1   1   1
...
READ_BUFRTOVS: file=amsuabufr       type=amsua      sis=amsua_metop-a
nread= 202755 ithin= 2 rmesh=1500.000000 isfcalc= 0 ndata=       602
ntask=  4
READ_BUFRTOVS: file=amsuabufr       type=amsua      sis=amsua_n15
nread= 202575 ithin= 2 rmesh=1500.000000 isfcalc= 0 ndata=       533
ntask=  4
...
READ_PREPBUFR: file=prepbufr        type=uv         sis=uv
nread=    178374 ithin= 0 rmesh=1450.000000 isfcalc= 0 ndata=    131588
ntask=  1
...
READ_OBS:  write collective obs selection info to obs_input.common
```

For the ensemble mean, the same procedure for observation processing as the GSI analysis is followed (`read_obs_check`, `READ_OBS`, `READ_BUFRTOVS`, `READ_GPS`, `READ_PREPBUFR` …). Finally, it is indicated that the collective observations selection information is written out for use with the ensemble memebers.

d. Ensemble members

For the `stdout_memnnn`, the observation processing sets are skipped and observation information is read in from the ensemble mean:

```
OBSERVER_SET:  read collective obs selection info from obs_input.common
```

Finally, for both the ensemble mean and members, the following lines will appear:

```
OBS_PARA:  ps    24   400   1658   2678   58   24   1644   514   82  402
637   258   36   67   328   864   44   224   5210   1397    69   414   50
633

OBS_PARA:  t   48   708  4088   6417  174  510   4390  1851  84   910
3136  1173  4  216  1056  2613  155   982   19875   3413   16   1629
1624  3095

OBS_PARA:  q   16  370   3033   4850  154  383   3359   1745   51  594
2147      1109   0   161   697  1951   140   850  11882   3001  155
1525  1104  1510

OBS_PARA:  pw   0    0    0    0    0    0    0    0  0   0    3  0   0
0  42   25    0    0  251    32    0     0    0   4
```

```
OBS_PARA:  uv   52    71    513    7196    239   1045   5256   1861   86
1739   4226  1343  3   85   1051   2637   184  1644   19825   3841   26
319   2547   3609
```

This table is important to check if the observations have been read in, which types of observations have been read in, and the distribution of observations in each subdomain.

8. Indication that the GSI observer has successfully run:

```
observer_final: successfully finalized
glbsoi: complete
000]gsisub(): : complete.

     ENDING DATE-TIME    JUL 13,2015  18:40:00.009  194  MON   2457217
     PROGRAM GSI_ANL HAS ENDED.
* . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * . *
```

After looking over each section of the standard output files, it can be concluded that the GSI global observer ran without issues. Note that because the outer loop was set to 0 (`miter=0`) for the GSI observer, no minimization occurred and therefore no analysis results were produced.

5.4 Running EnKF for Global Applications

5.4.1 Run Script

Once the GSI global observer has been run successfully, the next step is to setup the EnKF run script, *run_enkf_global.ksh*. This script is located under *comGSIv3.4-EnKFv1.0/run*. This run script uses the diag files generated by the GSI global observer script as observation input, and generates the global EnKF analysis. Similar to the GSI observer script, several user-specific modifications need to be made:

- Set up batch queueing system

```
#BSUB -P ?????????             # project code
#BSUB -W 00:30                 # wall-clock time (hrs:mins)
#BSUB -n 80                    # number of tasks in job
#BSUB -R "span[ptile=16]"      # run 16 MPI tasks per node
#BSUB -J enkf                  # job name
#BSUB -o enkf.%J.out           # output file name
#BSUB -e enkf.%J.err           # error file
```

- Set up number of processors and the job queue system used:

```
GSIPROC=80
ARCH='LINUX_LSF'
```

One difference from the GSI observer script is that the number ls of processors used should be greater than at least the number of ensemble members for running the EnKF. In this case we have 10 ensemble members and have requested 80 cores due to the large global domain.

- Set up the analysis time, global case, fixed files, EnKF executable

```
ANAL_TIME= 2014040506
GFS_CASE=enkf_glb_t254
WORK_ROOT/scratch/${user}/GSIv3.4-EnKFv1.0/run/enkf_${GFSCASE}
diag_ROOT=/scratch/${user}/GSIv3.4-EnKFv1.0/run/gsidiag_${GFSCASE}
BK_ROOT=/scratch/${user}/casedata/enkf_glb_t254/bk
GSI_ROOT=/scratch/${user}/GSIv3.4-EnKFv1.0
FIX_ROOT=${GSI_ROOT}/fix
ENKF_EXE=${GSI_ROOT}/src/main/enkf/global_enkf
CRTM_ROOT=/scratch/${user}/CRTM_REL-2.1.3
```

- Set location of the diag files (from GSI observer)

```
diag_ROOT=/scratch/${user}/GSIv3.4-EnKFv1.0/run/gsidiag_${GFSCASE}
```

One modifications to note for the EnKF is the `diag_ROOT`. The `diag_ROOT` points to the working directory where the GSI observer runs. This directory contains diag* files which will be linked to the EnKF working directory.

- Select ensemble and case parameters

```
 NMEM_ENKF=10
NLEVS=64
NVARS=5

elif [[ "$GFSCASE" = "enkf_glb_t254" ]]; then
  JCAP=254
  JCAP_B=254
elif [[ "$JCAP" = "254" ]]; then
   LONA=512
   LATA=256
   DELTIM=1200
   resol=2
```

This section sets information about the number of background ensemble members, domain specifications. This information mirrors that of the *run_gsi_global.ksh* script.

- Read in namelist

```
cat << EOF > enkf.nml
 &nam_enkf
  datestring="$ANAL_TIME",datapath="$tmpdir/",
  analpertwtnh=0.85,analpertwtsh=0.85,analpertwttr=0.85,
  covinflatemax=1.e2,covinflatemin=1,pseudo_rh=.true.,iassim_order=0,
  corrlengthnh=2000,corrlengthsh=2000,corrlengthtr=2000,
  lnsigcutoffnh=2.0,lnsigcutoffsh=2.0,lnsigcutofftr=2.0,
...
```

Note that unlike the regional enkf (*run_enkf_wrf.ksh*), the global EnKF builds the namelist within the run script rather than pulling from an outside script for namelist generation.

- Select observations

```
list="conv"
```

The previous line is contained with the ensemble looping section. For the observations to be assimilated, be aware that only observations/platforms that have been run through the GSI observer (and therefore a diag* file exists) are valid. For this example case, we will only run EnKF for conventional observations.

5.4.2 Run EnKF and check run status

Once the run script is set up properly for the case and the machine, EnKF can be run through the run script. The following command will submit the run:

```
$ BSUB < run_enkf_global.ksh
```

While the job is running, move to the working directory and check the details. Given the following working directory setup:

```
WORK_ROOT=/scratch/${user}/comGSIv3.4-EnKFv1.0/run/enkf_${GFSCASE}
```

Go to directory `WORK_ROOT=/scratch/${user}/comGSIv3.4-EnKFv1.0/` and check the run directory. A directory named `enkf_enkf_glb_t264` should have been created. This directory is the run directory for this EnKF case study. The directory will be populated with many links:

```
diag_conv_ges.ensmean              diag_conv_ges.mem001
diag_conv_ges.mem002               diag_conv_ges.mem003
diag_conv_ges.mem004               diag_conv_ges.mem005
…                                  diag_conv_ges.mem010

diag_amsua_metop-a_ges.ensmean     diag_amsua_metop-a_ges.mem001
diag_amsua_metop-a_ges.mem002      diag_amsua_metop-a_ges.mem003
diag_amsua_metop-a_ges.mem004      diag_amsua_metop-a_ges.mem005
…                                  diag_amsua_metop-a_ges.mem010

diag_amsua_n18_ges.ensmean         diag_amsua_n18_ges.mem001
diag_amsua_n18_ges.mem002          diag_amsua_n18_ges.mem003
diag_amsua_n18_ges.mem004          diag_amsua_n18_ges.mem005
…                                  diag_amsua_n18_ges.mem010

diag_hirs4_n19_ges.ensmean         diag_amsua_n15_ges.mem001
diag_hirs4_n19_ges.mem002          diag_amsua_n15_ges.mem003
diag_hirs4_n19_ges.mem004          diag_amsua_n15_ges.mem005
…                                  diag_amsua_n15_ges.mem010
```

The diag files are linked from the GSI global observer working directory. These links are specified in the run script. Note that ozone diag files (gome, omi, sbuv) will appear in the directory without links (zero length files) if ozone diags are not present from the GSI observer.

```
sfg_2014040506_fhr06_ensmean
sfg_2014040506_fhr06_mem001  sfg_2014040506_fhr06_mem002
sfg_2014040506_fhr06_mem003  sfg_2014040506_fhr06_mem004
sfg_2014040506_fhr06_mem005  sfg_2014040506_fhr06_mem006
sfg_2014040506_fhr06_mem007  sfg_2014040506_fhr06_mem008
```

```
sfg_2014040506_fhr06_mem009  sfg_2014040506_fhr06_mem010
```

The first guess files are also linked into the directory from the run script, pointing to the path of the ensemble mean and ensemble members designated in the setup section.

Similar to running the GSI observer, many other static files are linked or copied to this run directory, such as:

`enkf.nml`:	EnKF namelist
`convinfo`:	data usage control for conventional data
`satbias_in`:	satellite bias correction coefficient file
`satbias_angle`:	satellite bias correction angle file
`satinfo`:	data usage and channel control for satellite radiance data
`ozinfo`:	data usage control from ozone data
`covinflate.dat`:	Three-dimensional multiplicative inflation factor fields
`stdout`:	EnKF standard output file

The presence of the standard output file in the directory suggest the EnKF run script has successfully set up a run environment for the EnKF, properly linking the first guess and diag files from the GSI observer, and the EnKF executable is running. Once EnKF has finished running, analysis files should be generated for each member:

```
sanl_2014040506_mem001  sanl_2014040506_mem003  sanl_2014040506_mem005
sanl_2014040506_mem007  sanl_2014040506_mem009  sanl_2014040506_mem002
sanl_2014040506_mem004  sanl_2014040506_mem006  sanl_2014040506_mem008
sanl_2014040506_mem010
```

5.4.3 Check for successful EnKF completion

The presence of the EnKF analysis files for each member as well as the standard out file indicates that the EnKF has run without crashing, but does not necessary indicate a successful analysis. It is important to check the stdout file in the run directory to make sure the EnKF completed each step without any obvious problems. The following are several important areas of the standard out file to check:

1. Check namelist has been properly read in and configuration is correct:

```
namelist parameters:
 --------------------
 &NAM_ENKF
 DATESTRING      = 2014040506,
 DATAPATH        = /enkf_glb_t254/enkf_glb_t254_tmp/
 IASSIM_ORDER    =           0,
 COVINFLATEMAX   =   100.0000    ,
 COVINFLATEMIN   =   1.000000    ,
 DETERMINISTIC   = T,
 SORTINC = T,
 CORRLENGTHNH    =   2000.000    ,
 CORRLENGTHTR    =   2000.000    ,
```

2. Check analysis time, number of ensemble members, as well as the actual analysis variables and the background type. The maximum and minimum values for surface pressure are printed for a sanity check:

```
analysis time 2014040506
          10  members
 first-guess forecast hour for analysis = 06
           5 3d vars to update
 total of         321  2d grids will be updated (including ps)
 using multiplicative inflation based on Pa/Pb
 Vars in Rad-Jacobian (dims)
 --------------------------
 sst                            0
 ensemble mean first guess surface pressure:
   516.109537854119        1051.19857562343
```

3. Statistics of the ensemble prior

After many lines describing the bias correction coefficents and the inventory of the observation number and types from the input diag* files, as well as the time spent reading in each observation type, statistics of the ensemble priors' fit to all observations are provided for each region (NH, SH, TR):

```
innovation statistics for prior:
 conventional obs
 region, obtype, nobs, bias, innov stdev, sqrt(S+R), sqrt(S), sqrt(R):
NH all ps 12196 -0.754E-01  0.122E+01  0.264E+01  0.498E+00  0.259E+01
TR all ps  2192  0.402E+00  0.118E+01  0.225E+01  0.476E+00  0.219E+01
SH all ps   849 -0.102E+00  0.125E+01  0.230E+01  0.563E+00  0.223E+01
NH all t  31102  0.874E-01  0.151E+01  0.583E+01  0.372E+00  0.582E+01
TR all t   6904  0.215E+00  0.148E+01  0.410E+01  0.412E+00  0.408E+01
SH all t   1524  0.119E+00  0.161E+01  0.576E+01  0.451E+00  0.575E+01
NH all uv 70420  0.196E+00  0.304E+01  0.388E+01  0.951E+00  0.376E+01
TR all uv 22620  0.900E-01  0.331E+01  0.382E+01  0.125E+01  0.361E+01
SH all uv  5388 -0.111E+00  0.366E+01  0.386E+01  0.113E+01  0.369E+01
NH all q   7936 -0.335E-01  0.181E+00  0.161E+04  0.534E-01  0.161E+04
TR all q   2951 -0.119E-01  0.180E+00  0.322E+03  0.770E-01  0.322E+03
SH all q    543 -0.163E-01  0.212E+00  0.251E+04  0.730E-01  0.251E+04
```

This table should be checked in order to determine if the inflation is appropriate. As mentioned in the regional EnKF example, the goal is to make the total ensemble spreads of priors (*sqrt*(S+R) match the innovations (*innov std)*ohh as much as possible. Because this is a global run using only conventional observation data, we can see that all regions (NH,TR,SH) are listed and there are no statistics for radiances. We can see certain obtypes, particularly in the uv is fairly close values. On the other hand, there are still regions/obtypes that have noteable differences, mainly assocaiated with larger observation errors (*sqrt*(R)). Users should consider tuning of inflation and localization, which is typically determined using cases with multiple assimilation cycles. Refer to section 4.2 of this User's Guide for more information on tuning.

4. Domain and observation partition:

```
npts =        83734
min/max number of points per proc =          970        1110
time to do model space decomp =   1.058491191361099E-002
nobsgood =       164625
min/max number of obs per proc =         2057        2058
time to do ob space decomp =   6.587638054043055E-004
sending out observation prior ensemble perts from root ...
... took   8.096099831163883E-003  secs
time in load_balance =  0.710615877760574      on proc           0
```

The analysis variables and the observations are distributed to different processors. We can see in this case that the min/max number of points per processor are 970 and 1110, respectively. Similarly, we can see that the min/max number of observations per processor are 2057 and 2058, respectively, indicating that the observations are well dispersed among the processors.

```
min/max pressi           1   516.1096       1051.199
min/max pressi           2   513.3593       1045.597
min/max pressi           3   510.2454       1039.249
min/max pressi           4   506.7440       1032.063
min/max pressi           5   502.8375       1023.943
min/max pressi           6   498.5139       1014.783
min/max pressi           7   493.7615       1004.469
min/max pressi           8   488.5688       992.8854
min/max pressi           9   482.9234       979.9127
```

Additionally, check the minimum and maximum values of the fields at each vertical level as a quick sanity check.

5.4.4 Diagnose EnKF analysis results

At the bottom of the standard output file, there are several output statistics and tables that are helpful for users to diagnose the quality of the EnKF analysis.

5. Statistics of the ensemble analysis

```
min/max var 1 inflation =    1.000000       10.55177
min/max var 2 inflation =    1.000000       10.39050
min/max var 3 inflation =    1.000000       11.43062
min/max spfh inflation =     1.000000       10.68039
min/max ps inflation =       1.000000       11.74329
global ps prior std. dev min/max =   7.3517814E-02   3.006298

NH mean ps prior standard deviation =   0.5570135
NH mean ps posterior standard deviation (before inflation)=   0.2696404
NH mean ps posterior standard deviation (after inflation) =   0.5094000
NH mean ps inflation =    2.480218
SH mean ps prior standard deviation =   0.6046332
SH mean ps posterior standard deviation (before inflation)=   0.5087804
SH mean ps posterior standard deviation (after inflation) =   0.5890263
SH mean ps inflation =    1.253871
TR mean ps prior standard deviation =   0.4883641
TR mean ps posterior standard deviation (before inflation)=   0.3675643
```

```
 TR mean ps posterior standard deviation (after inflation) =   0.4678227
 TR mean ps inflation =    1.462317
 time in inflate_ens =  0.560212401556782      on proc           0
```

This section is important for checking whether the values of the inflation are reasonable by looking at a summary of the maximum and minimum values. Also, viewing the regional averaged statistics before and after inflation. For this case we can see for all regions, larger standard deviation after the inflation are reported, which is more consistent with the prior standard deviation.

6. Spread inflation of the analysis ensemble

```
innovation statistics for posterior:
 conventional obs
 region, obtype, nobs, bias, innov stdev, sqrt(S+R), sqrt(S), sqrt(R):
NH  all ps 12196  0.857E-01  0.937E+00  0.259E+01  0.132E+00  0.259E+01
TR  all ps  2192  0.231E+00  0.103E+01  0.221E+01  0.221E+00  0.219E+01
SH  all ps   849 -0.714E-01  0.992E+00  0.225E+01  0.330E+00  0.223E+01
NH  all t  31102  0.689E-01  0.137E+01  0.582E+01  0.136E+00  0.582E+01
TR  all t   6904  0.174E+00  0.131E+01  0.408E+01  0.194E+00  0.408E+01
SH  all t   1524  0.715E-01  0.138E+01  0.575E+01  0.216E+00  0.575E+01
NH  all uv 70420  0.870E-01  0.271E+01  0.377E+01  0.354E+00  0.376E+01
TR  all uv 22620  0.166E-01  0.278E+01  0.365E+01  0.519E+00  0.361E+01
SH  all uv  5388 -0.860E-01  0.303E+01  0.373E+01  0.543E+00  0.369E+01
NH  all q   7936 -0.340E-01  0.172E+00  0.161E+04  0.187E-01  0.161E+04
TR  all q   2951  0.288E-02  0.152E+00  0.322E+03  0.298E-01  0.322E+03
SH  all q    543 -0.191E-01  0.186E+00  0.251E+04  0.334E-01  0.251E+04
```

After the EnKF analysis, it is important to check the innovation statistics, as previously discussed in the regional test case.

Chapter 6: EnKF Basic Concepts and Code Structure

This chapter briefly describes basic concepts used in the current implementation of the NOAA Ensemble Kalman Filter (EnKF) system in the form of an Ensemble Square Root Filter (EnSRF, Whitaker and Hamill, 2002, 2012). Also, the main code structure is documented.

6.1 Basic concepts

6.1.1 Analysis variables

In theory, EnKF can use any of the model prognostic variables as analysis variables as long as there exist meaningful/clear correlations between the variables and observations. Typically, for hydrostatic global models, horizontal wind components, temperature, water vapor, surface pressure, and ozone are used as analysis variables. For non-hydrostatic meso-scale models, like WRF, the vertical component of wind, rain/cloud related water content variables, and surface variables could be used as additional analysis variables.

6.1.2 Update of analysis variables

The minimum error-variance estimate of the analyzed variables $\mathbf{X}^a$ is given by the traditional Kalman filter update equation,

$$\mathbf{X}^a = \mathbf{X}^b + \mathbf{K}(\mathbf{y}^o - \mathbf{H}\mathbf{X}^b), \qquad (1)$$

$$\mathbf{K} = \mathbf{P}^b\mathbf{H}^T(\mathbf{H}\mathbf{P}^b\mathbf{H}^T + \mathbf{R})^{-1} \qquad (2)$$

Where:

- $\mathbf{X}^b$: an *m-dimensional* background model forecast (i.e., prior)
- $\mathbf{X}^a$: an *m-dimensional* analyses at model grids (i.e., posterior)
- $\mathbf{y}^o$: a *p-dimensional* set of observations
- $\mathbf{H}$: the operators that convert the model state to the observation space
- $\mathbf{P}^b$: the $m \times m$*-dimensional* background-error covariance matrix
- $\mathbf{R}$: the $p \times p$*-dimensional* observation-error covariance matrix
- $\mathbf{K}$: the *Kalman gain*

Here, the Kalman gain is the multivariate covariance between the priors/backgrounds of observations and the model analysis variables and it maps the observational increments (O-B) at observations' locations to analyses variables at model grids. This update process is basically similar to a simple optimal interpolation (OI) scheme, where the Kalman gain is set to static.

The update equations (1) and (2) can be solved using ensemble technique. The Kalman gain can be estimated and propagated using a set of ensemble forecasts. Expressing the model state vector of the analysis variables as an ensemble mean (denoted by an overbar) and a deviation from the mean (denoted by a prime), the update equations for series ensemble square root filter (EnSRF; Whitaker and Hamill 2002) may be written as:

$$\overline{\mathbf{X}}^a = \overline{\mathbf{X}}^b + \mathbf{K}(\mathbf{y}^o - \mathbf{H}\overline{\mathbf{X}}^b) \qquad (3)$$

$$\mathbf{X}'^a = \mathbf{X}'^b - \tilde{\mathbf{K}}\mathbf{H}\mathbf{X}'^b \qquad (4)$$

$$\tilde{\mathbf{K}} = \alpha\mathbf{K} \qquad (5)$$

$$\alpha = 1/(1+\sqrt{R/(\mathbf{H}\mathbf{P}^b\mathbf{H}^T + R)}) \qquad (6)$$

Where

$\mathbf{K}$ is the Kalman gain defined by (2) and calculated using the ensemble method as below:
$\tilde{\mathbf{K}}$ is the gain used to update ensemble deviations from the ensemble mean.

Here, the observational error covariance is assumed uncorrelated, that is, $\mathbf{R}$ is diagonal. Then observations can be assimilated serially, one at a time, so that the analysis after assimilation of the *Nth* observation becomes the background estimate for assimilating the (*N*+1)*th* observation.

For an individual observation, *R* and $\mathbf{H}\mathbf{P}^b\mathbf{H}^T$ are scalars, $\mathbf{K}$ and $\tilde{\mathbf{K}}$ are vectors of the same dimension as the model state vector (before applying localization). Both $\mathbf{K}$ and $\tilde{\mathbf{K}}$ are calculated from the prior ensemble of the observation being assimilated and each of the analyses variables on the model grids individually.

Here, $\mathbf{P}^b\mathbf{H}^T$ and $\mathbf{H}\mathbf{P}^b\mathbf{H}^T$ are estimated statistically from an ensemble of model forecasts/background. Specifically, these two quantities are obtained as:

$$\mathbf{P}^b\mathbf{H}^T = \overline{\mathbf{X}'^b(\mathbf{H}\mathbf{X}'^b)^T} = \sum_{i=1}^{n}\mathbf{X}_i'^b(\mathbf{H}\mathbf{X}_i'^b)^T/(n-1) \qquad (7)$$

$$\mathbf{H}\mathbf{P}^b\mathbf{H}^T = \overline{\mathbf{H}\mathbf{X}'^b(\mathbf{H}\mathbf{X}'^b)^T} = \sum_{i=1}^{n}\mathbf{H}\mathbf{X}_i'^b(\mathbf{H}\mathbf{X}_i'^b)^T/(n-1) \qquad (8)$$

where:
n is the ensemble size of model forecasts; *i* is the index of each individual ensemble member.

The expected analyses error covariance at the model grids after assimilation is given by:

$$\mathbf{P}^a = (\mathbf{I} - \mathbf{K}\mathbf{H})\mathbf{P}^b \qquad (9)$$

6.1.3 Updates of observation priors

In a series assimilation of observations, the model first-guess or backgrounds at model grids are updated by one single observation at a time. For the assimilation of next observations, the first-guess of the next observations (observation priors) needs to be re-computed using the updated model background and a forward observation operator. This process is straightforward for running on a single processor computer, but is not efficient in parallel computing environment, particularly when the first-guess of the observations are all pre-calculated.

Alternatively, the first-guess of the next observations can also be updated by the observation being assimilated, similar to the update to the model variables, so that re-computing the first-guess using the observational operators is not needed (see Anderson and Collins, 2007). After the update of the first guess of model variables, the *Nth* observation being assimilated is also used to update the first-guess of the *(N+1)th* and next observations within the localization distance. This process can be expressed in the following update equations, in similarity to Eqs (3)-(6):

$$\overline{\mathbf{Z}}^a = \overline{\mathbf{Z}}^b + \mathbf{K}_z(y^o - \mathbf{H}\overline{\mathbf{X}}^b) \qquad (10)$$

$$\mathbf{Z}'^a = \mathbf{Z}'^b - \alpha\mathbf{K}_z\mathbf{H}\mathbf{X}'^b \qquad (11)$$

$$\mathbf{K}_z = \mathbf{Z}'^b(\mathbf{H}\mathbf{X}'^b)^T(\mathbf{H}\mathbf{P}^b\mathbf{H}^T + R)^{-1} \qquad (12)$$

Where

y^o is the observation being assimilated.

$\mathbf{Z}^b$ is the first-guess of the next unassimilated observations.

$\mathbf{Z}^a$ is the updated first-guess of the next unassimilated observations.

$\mathbf{K}_z$ is the Kalman gain between the first-guess of the observation being assimilated and next unassimilated observations.

6.1.4 Assimilation order and adaptive thinning of observations

In realistic assimilation systems, observations may have non-linear relation with the analysis variables, and there are sampling errors due to the limited ensemble sizes, as a result, the analysis can depend on the order of the observations assimilated.

In the implementation, there are three options for choosing the orders of the observations for assimilation:

a). assimilate observations in the order they are read in (default). This seems a reasonable choice for assimilating "BEST" observation types first (like radiosonde winds).

b). shuffle the observations randomly before assimilating;

c). assimilate in order of increasing predicted observation analysis variance relative to the prior.

For the 3rd option, the predicted observational analyses variance against the first-guess at the observational space is defined as (for details, see Whitaker et al., 2008, 2012):

$$\mathbf{H}\mathbf{P}^a\mathbf{H}^T / \mathbf{H}\mathbf{P}^b\mathbf{H}^T = R / (\mathbf{H}\mathbf{P}^b\mathbf{H}^T + R) \qquad (13)$$

Note that the predicted variance is based on observation priors' variance as if the observation is assimilated alone, and does not include the effect of the assimilation of other observations.

The observations can be further thinned adaptively. It is done via the online updated estimation of the predicted analyses variance of next observations. If the predicted analyses variance for the observation is very close to the prior of the observation, that is, the impact of the observation is expected very small, then this observation can be skipped. The threshold is set by namelist parameter *paoverpb_thresh*.

6.1.5 Ensemble spread inflation

A multiplicative inflation (Whitaker and Hamill, 2012) is used to inflate analyses/posterior ensemble spread back to the one of first-guess. The amount of inflation is given at each analysis grid point by:

$$\begin{aligned} \sigma^b &= \sqrt{\sum_{i=1}^{n} (\mathbf{X}_i^{'b})^2 /(n-1)} \\ \sigma^a &= \sqrt{\sum_{i=1}^{n} (\mathbf{X}_i^{'a})^2 /(n-1)} \\ r &= (\beta \frac{\sigma^b - \sigma^a}{\sigma^a} + 1) \\ r\mathbf{X}_i^{'a} &\rightarrow \mathbf{X}_i^{'a} (\text{inflated}) \end{aligned} \qquad (14)$$

where σ^b is the prior/first-guess ensemble standard deviation.

σ^a is posterior/analyses ensemble standard deviation (before inflation).

r is the inflation factor applied to each ensemble member deviation from the ensemble mean.

β is a tunable namelist parameter (*analpertwt*) defined in module params. If β=1, ensemble is inflated so posterior standard deviation becomes the same as prior. If β =0, there is no inflation.

For a given value of β, the inflation factor is proportional to the amount of ensemble spread reduction by assimilation of observations, normalized by the analyses ensemble spread. As a result, the inflation is in general larger where observations are denser or have larger impact. The actual inflation factor can be quite different for each variable, level and horizontal grid point. If the smoothing namelist parameter (*smoothparm*) > 0, the estimated inflation factor is smoothed using a Gaussian spectral filter with an e-folding scale of *smoothparm*. The minimum and maximum values allowed can be controlled by namelist parameters.

In addition to the above inflation, an additive inflation by adding random noise from a climatology distribution of the model errors is shown beneficial (Whitaker et al., 2012). The amount of the random noises to be added is a tunable parameter too.

The amount of inflation from the two inflation schemes needs to be tuned to satisfy the following relation (Houtekamer, et al., 2005), as close as possible:

$$\left\langle (\mathbf{y}^o - \mathbf{H}\bar{\mathbf{X}}^b)(\mathbf{y}^o - \mathbf{H}\bar{\mathbf{X}}^b) \right\rangle = (\mathbf{H}\mathbf{P}^b\mathbf{H}^T + \mathbf{R}) \qquad (15)$$

Satisfaction of this equation ensures that the total ensemble spreads (ensemble spreads plus observational error covariance, right side of the equation) is a reasonable estimation of the RMS errors of observation priors against observations (left side of the equation). This could justify that the ensemble system works reasonably well.

In addition, the ensemble spreads should be tuned appropriately relative to observation errors (assuming these errors are correctly set). Substantially smaller ensemble spreads could lead to underweight of observations. As a result, EnKF may divergence or ignore the observations.

6.1.6 Covariance localization

To reduce the impact of spurious ensemble covariance on the update of both analyses variables at model grids and the first-guess of observations (observation priors), localization is applied to the covariance (Kalman gains) in the Equations 7 and 12 in both horizontal and vertical. The function of Gaspari and Cohn (1999) is used in horizontal localization. It uses a 5th order compact polynomial and the impact of observations is gradually reduced to zero at the specified cutoff distance.

In vertical localization, the scale height is used, in units of -log(P/P_{ref}).

The localization distances are important tunable parameters for the successful analysis of EnSRF. The tuning may depend on several factors, such as, the ensemble sizes used, model grid resolutions, weather systems, etc. In general, larger ensemble sizes can allow larger localization distances. But assimilation with smaller ensemble sizes may benefit from reduced localization distances to reduce the impact of spurious covariance. In addition,

assimilation with higher-resolution observations, like radar data, may need much smaller localization distances.

6.1.7 Adaptive radiance bias correction with EnSRF

An adaptive radiance bias correction procedure is used with the assimilation of satellite radiance data in EnSRF. The first-guess of the radiance observations are updated by the radiance observations using the assimilation procedure outlined in A. 3. The updated innovations (O-B) are then used to update the coefficients of the radiance bias correction scheme. The first-guess of the radiance observations are updated again using the updated bias correction coefficients. This process may be repeated/iterated multiple times until the updated first-guess of the radiance observations and bias correction coefficients converge.

6.2 EnSRF code structure and key functions

6.2.1 Main code tree

The code structure for main code enkf_main.f90:

a) Read the namelist (enkf.nml), observations, observation priors for each ensemble member (from diag** files generated by GSI forward operators). Generate innovation statistics for the prior ensemble.
b) Read in model horizontal grid information (latitude/longitude of each grid point) and pressure at each vertical level.
c) Decomposition of the model horizontal grid points and observation priors.
d) Read in the analyses variables of prior ensemble, and distribute to each processor.
e) Update the radiance observation priors and bias correction coefficients
f) Update analysis variables, observation priors.
g) Inflate posterior ensemble using the multiplicative inflation scheme.
h) Generate the innovation statistics of the posterior and the statistics of inflation applied to the posterior.
i) Write out the analysis ensemble files and updated radiance bias correction coefficients.
j) Add the additive inflation to the posterior ensemble (offline).

6.2.2 Update of radiance observation priors and bias correction coefficients

In the analyses update process of EnSRF, the radiance observation priors and the bias correction coefficients are updated first. This may involve multiple iterations, before other observation types are assimilated. The code for this function is *ens_update* of enkf.F90:

a) Determine the order of radiance observations for assimilation according to the namelist choice.
b) Read in the radiance prior/first-guess ensemble
c) Apply the radiance bias correction to the observation priors and gross errors of VARQC type check.
d) Start assimilation loop over the radiance observations, one observation at a time.
e) For each of radiance observations being assimilated, find out other radiance observations within the specified localization distances.
f) Updates of the first-guess of these close radiance observations,
g) Go to back to Step d) for assimilation of next radiance observation.
h) Finish the observation prior update for all of the radiance observations. The bias coefficients are revised using the revised first-guess of all radiance observations.
i) Go back to Step c) and repeat the process using the revised bias correction coefficients and the updated radiance observation priors for multiple times until these become stable and converged.

6.2.3 Model variables and observation prior update

At the final iteration for the update of radiance bias correction coefficients, the priors of model analyses variables and observation priors are updated by all observation types. The code for this is *ens_update*_of enkf_F90 and *inflate_ens* of inflation.f90:

a). Determine the order of all observations for assimilation according to the namelist choice.
b). Start assimilation loop over all observations, one observation at a time.
c). For each observation being assimilated, find out the model grids points and next observations within the specified localization distances.
d). The ensemble mean and perturbations of these close model analyses variables are updated by the observation being assimilated. The ensemble mean and perturbations of first guess

of the next and close observations are also updated by the observation being assimilated.

e). An adaptive observation thinning can be turned on to decide if the observation being assimilated can be skipped based on the predicted analyses ensemble spread of the observation.

f). Go back to Step c) for assimilation of next observation.

g). After assimilation of all observations, inflate the analyses ensemble spreads using the multiplicative inflation scheme.

6.2.4 Key code functions

a) Calculate the predicted analyses ensemble spread of the observations (Equation 8). The variable is denoted in the code as *paoverpb_chunk*. The related code are lines 281-286 (enkf.F90):

```
if (iassim_order == 2) then
   do nob1=1,numobsperproc(nproc+1)
     nob2 = indxproc_obs(nproc+1,nob1)
     paoverpb_chunk(nob1) =
       oberrvaruse(nob2)/(oberrvaruse(nob2)+obsprd_prior(nob2))
   enddo
endif
```

b) Calculation of the Kalman gain (K, Equations 5 and 6) and mapping the analyses increments of the observation being assimilated to the ensemble mean and perturbations of analyses variables at model grids (Equations 3 and 4). The code variables are: *kfgain, ensmean_chunk, and anal_chunk.* The corresponding code is lines 489-494 of enkf.f90:

```
    kfgain=taperv(nnn)*sum(anal_chunk(:,i,nn)*anal_obtmp)
! update ensemble mean.
    ensmean_chunk(i,nn) = ensmean_chunk(i,nn) +kfgain*obinc_tmp
! update ensemble perturbations.
    anal_chunk(:,i,nn) = anal_chunk(:,i,nn) + kfgain*obganl(:)
```

c) Calculation of the Kalman gain (K_z) and mapping the analyses increments of the observation being assimilated to the nearby observation priors (Equations 12, 10, and 11). The code variables are: *kfgain, ensmean_obchunk, and anal_obchunk.* The corresponding code is lines 514-521 of enkf.f90:

```
        kfgain = taper_disob(nob1)* &
           taper(lnsig*lnsiglinv)*taper(obt*obtimelinv)* &
           sum(anal_obchunk(:,nob2)*anal_obtmp)*hpfhtcon
  ! update ensemble mean of observation priors.
        ensmean_obchunk(nob2) = ensmean_obchunk(nob2)
                                +kfgain*obinc_tmp
  ! update ensemble perturbations of observation priors.
        anal_obchunk(:,nob2) = anal_obchunk(:,nob2) + kfgain*obganl
```

Appendix A: Content of Namelist

The following are lists and explanations of the EnKF namelist variables. Users can also check file "*params.f90*" for the details.

Section **nam_enkf**

Variable Name	Description	Data Type	Default
datein	Analysis date in YYYYMMDDHH	integer	0
datapath	path to data directory (include trailing slash)	Character (len=500)	" "
iassim_order	= 0 for the order they are read in, =1 for random order = 2 for order of predicted posterior variance reduction (based on prior)	integer	0
covinflatemax	maximum inflation	real(r_single)	1.e30
covinflatemin	minimum inflation	real(r_single)	1.0
deterministic	if true, use EnSRF w/o perturbed obs. if false, use perturbed obs EnKF.	logical	true
sortinc	if false, re-order obs to minimize regression errors as described in Anderson (2003).	logical	true
corrlengthnh	length for horizontal localization (in km) in north hemisphere	real(r_single)	2800
corrlengthtr	length for horizontal localization (in km) in tropic	real(r_single)	2800
corrlengthsh	length for horizontal localization (in km) in south hemisphere	real(r_single)	2800
varqc	Turn on varqc	logical	false
huber	use huber norm instead of "flat-tail"	logical	false
nlons	number of lons	integer	0
nlats	Number of lats	integer	0
smoothparm	smoothing parameter for inflation (-1 for no smoothing)	real(r_single)	-1
readin_localization	If true, read in localization length scales from an external file	logical	false
zhuberleft	Parameter for "huber norm" QC	real(r_single)	1.e30
zhuberright	Parameter for "huber norm" QC	real(r_single)	1.e30
obtimelnh	observation time localization in hours over north hemisphere	real(r_single)	25.925
obtimeltr	observation time localization in hours over tropic	real(r_single)	25.925
obtimelsh	observation time localization in hours over south	real(r_single)	25.925

	hemisphere		
reducedgrid	Do smooth in a reduced grid with a variable number on longitudes per latitude. The number of longitudes is chosen so that the zonal grid spacing is approximately the same as at the equator	logical	false
lnsigcutoffnh	length for vertical localization in ln(p) over north hemisphere for conventional observation	real(r_single)	2.0
lnsigcutofftr	length for vertical localization in ln(p) over tropic for conventional observation	real(r_single)	2.0
lnsigcutoffsh	length for vertical localization in ln(p) over south hemisphere for conventional observation	real(r_single)	2.0
lnsigcutoffsatnh	length for vertical localization in ln(p) over north hemisphere for satellite radiance observation	real(r_single)	-999.0
lnsigcutoffsattr	length for vertical localization in ln(p) over tropic for satellite radiance observation	real(r_single)	-999.0
lnsigcutoffsatsh	length for vertical localization in ln(p)over south hemisphere for satellite radiance observation	real(r_single)	-999.0
lnsigcutoffpsnh	length for vertical localization in ln(p) over north hemisphere for surface pressure observation	real(r_single)	-999.0
lnsigcutoffpstr	length for vertical localization in ln(p) over tropic for surface pressure observation	real(r_single)	-999.0
lnsigcutoffpssh	length for vertical localization in ln(p) over south hemisphere for surface pressure observation	real(r_single)	-999.0
analpertwtnh	adaptive posterior inflation parameter over north hemisphere:	real(r_single)	0.0
	1 means inflate all the way back to prior spread		
analpertwtsh	adaptive posterior inflation parameter over tropic:	real(r_single)	0.0
	1 means inflate all the way back to prior spread		
analpertwttr	adaptive posterior inflation parameter over south hemisphere:	real(r_single)	0.0
	1 means inflate all the way back to prior spread		
sprd_tol	tolerance for background check: observations are not used if they are more than sqrt(S+R) from mean, where S is ensemble variance and R is observation error variance.	real(r_single)	9.9e31
nlevs	total number of levels	integer	0
nanals	number of ensemble members	integer	0
nvars	number of 3d variables to update. For hydrostatic models, typically 5 (u,v,T,q,ozone).	integer	5
saterrfact	factor to multiply sat radiance errors	real(r_single)	1.0
univaroz	If true, ozone observations only affect ozone	logical	true
regional	If true, analysis is for regional	logical	false
use_gfs_nemsio	If true, GFS background is in NEMS format	logical	false

paoverpb_thresh	if observation space posterior variance divided by prior variance less than this value, observation is skipped during serial processing. 1.0 = don't skip any obs	real(r_single)	1.0
latbound	definition of tropics and mid-latitudes (for inflation).	real(r_single)	25.0
delat	width of transition zone	real(r_single)	10.0
pseudo_rh	use 'pseudo-rh' analysis variable, as in GSI	logical	false
numiter	number of times to iterate state/bias correction update. (only relevant when satellite radiances assimilated, i.e. nobs_sat>0)	integer	1.0
biasvar	background error variance for rad bias coeffs (used in radbias.f90). Default is (old) GSI value. if negative, bias coeff error variace is set to -biasvar/N, where N is number of obs per instrument/channel. if newpc4pred is .true., biasvar is not used - the estimated analysis error variance from the previous cycle is used instead (same as in the GSI).	real(r_single)	0.1
lupd_satbiasc	if performing satellite bias correction update	logical	true
cliptracers	if true, tracers are clipped to zero when read in, and just before they are written out.	logical	true
simple_partition	partition obs for enkf using Graham's rule	logical	true
adp_anglebc	turn off or on the variational radiance angle bias correction	adp_anglebc	false
angord	order of polynomial for angle bias correction	Integer	0
newpc4pred	controls preconditioning due to sat-bias correction term	logical	
nmmb	If true, ensemble forecast is NMMB	logical	false
iau		logical	false
nhr_anal	background forecast time for analysis	integer	6
letkf_flag	If true, do LETKF	logical	false
boxsize	Observation box size for LETKF (deg)	real(r_single)	90.0
massbal_adjust	mass balance adjustment for GFS	logical	false
use_edges	logical to use data on scan edges (.true.=to use)	logical	true
emiss_bc	If true, turn on emissivity bias correction	logical	false

Section **nam_wrf**

Varaible Name	Description	Data Type	Default
arw	regional dynamical core ARW	logical	false
nmm	regional dynamical core NMM	logical	true

doubly_periodic	logical	true

Section **satobs_enkf**

Varaible Name	Description	Data Type	Default
sattypes_rad	strings describing the satellite data type (which form part of the diag* filename).	character(len=20) array (nsatmax_rad)	' '
dsis	strings corresponding to sattypes_rad which correspond to the names in the NCEP global_satinfo file.	character(len=20) array (nsatmax_rad)	' '

Section **ozobs_enkf**

Varaible Name	Description	Data Type	Default
sattypes_oz	strings describing the ozone satellite data type (which form part of the diag* filename)	character(len=20) array (nsatmax_oz)	' '