GSI Gridpoint Statistical Interpolation

COMMUNITY VERSION 3.4

User's Guide

July 2015

Developmental Testbed Center

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

Foreword

This document is the 2015 Gridpoint Statistical Interpolation (GSI) User's Guide geared particularly for beginners. It describes the fundamentals of using GSI Version 3.4, including basic skills of installing, running, and diagnosing GSI. Advanced features of GSI as well as details of assimilation of specific data types can be found in the *Advance GSI User's Guide*, released together with this document and the Version 3.4 code release.

This version of GSI was released on July 31, 2015. Its code and supplemental libraries and files are based on a revision of the GSI repository in March 2015.

This User's Guide includes five chapters and two appendices:

Chapter 1: Overview

Chapter 2: Software Installation

Chapter 3: Running GSI

Chapter 4: GSI Diagnostics and Tuning

Chapter 5: GSI Applications for Reginal 3Dvar and Hybrid

Appendix A: GSI Community Tools

Appendix B: Content of Namelist Section OBS_INPUT

Appendix C: GSI Namelist: Name, Default value, Explanation

For the latest version of this document, please visit the GSI User's Website at

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

For referencing this document, please use:

```
Developmental Testbed Center, 2015: Gridpoint Statistical
Interpolation (GSI) User's Guide for Version 3.4. Available at
http://www.dtcenter.org/com-GSI/users/docs/index.php, 143 pp.
```

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

Editors: Ming Hu, Hui Shao, Donald Stark, Kathryn Newman, Chunhua Zhou

Contributors to this guide:

NOAA/NCEP/EMC: John Derber, Russ Treadon, Mike Lueken, Wan-Shu Wu, Andrew Collard, Ed Safford

NCAR: Xiang-Yu Huang, Syed Rizvi, Zhiquan Liu, Tom Auligne, Arthur Mizzi

NOAA/ESRL/GSD: Steve Weygandt, Dezso Devenyi, Joseph Olson

Acknowledgements:
We thank the U.S. Air Force and the National Oceanic and Atmospheric Administration 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

GSI history and background:

The Gridpoint Statistical Interpolation (GSI) system is a unified data assimilation (DA) system for both global and regional applications. It was initially developed by the National Centers for Environmental Prediction (NCEP) Environmental Modeling Center (EMC) as a next generation analysis system based on the then operational Spectral Statistical Interpolation (SSI) analysis system. Instead of being constructed in spectral space like the SSI, the GSI is constructed in physical space and is designed to be a flexible, state-of-art system that is efficient on available parallel computing platforms. Starting with a three-dimensional variational (3D-Var) data assimilation technique, current GSI can be run as a data assimilation system of 2D-Var (for surface data analysis), 3D-Var, hybrid ensemble-variational, 4D EnVar, or 4D-Var if coupled with an adjoint model of GSI supported forecast systems.

After initial development, the GSI analysis system became operational as the core of the North American Data Assimilation System (NDAS) for the North American Mesoscale (NAM) system in June 2006 and the Global Data Assimilation System (GDAS) for the Global Forecasting System (GFS) in May 2007 at National Oceanic and Atmospheric Administration (NOAA). Since then, the GSI system has been adopted in various operational systems, including the National Aeronautics and Space Administration (NASA) Goddard Earth Observing System Model (GEOS), the Air Force Weather Agency (AFWA) mesoscale data assimilation system, the NOAA Real-Time Mesoscale Analysis (RTMA) system, the Hurricane WRF (HWRF), and the RAPid Refresh (RAP) system, etc. The number of groups involved in operational GSI development has also been expanded to include more development groups, including NASA Goddard Global Modeling and Assimilation Office (GMAO), NOAA Earth System Research Laboratory (ESRL), and National Center for Atmospheric Research (NCAR).

GSI becomes community code:

In 2007, the Developmental Testbed Center (DTC) began collaborating with major GSI development groups to transform the operational GSI system into a community system and support distributed development. The DTC has complemented the development groups in providing GSI documentation, porting GSI to multiple platforms, and testing GSI in an independent and objective environment, while still maintaining functionally equivalent to operational centers. Working with EMC, the DTC is maintaining a community GSI repository that is equivalent to the operational developmental repository and facilitates community users to develop GSI. Based on the repository, the DTC releases GSI code annually with updated documentation. The first community version of the GSI system was released by the DTC in 2009. Since then, the DTC has provided community support through the GSI helpdesk (gsi_help@ucar.edu), community GSI webpage

(http://www.dtcenter.org/com-GSI/users/index.php), and community GSI tutorials and workshops.

GSI code management and Review Committee:

A GSI review committee was formed in 2010 and expanded in 2011 to coordinate distributed development of the GSI system. In 2014, EMC and DTC decided to merge their GSI code repository with the code repository of the NOAA Ensemble Kalman Filter system. Such a merging enabled coordinated development of both systems and joint community support. Following the repository merging, the GSI Review Committee was transitioned to a joint Data Assimilation Review Committee (DRC), incorporating new numbers representing the EnKF development and applications. Currently, DRC contains members from NCEP/EMC, NASA/GMAO, NOAA/ESRL, NCAR/MMM, NOAA's Satellite and Information Service (NESDIS), Air Force (AF), University of Maryland, and the DTC. As one of the committee members, the DTC represents the research community with GSI/EnKF users from all over the world. More information about this EnKF system can be found from a separate documentation, the Community EnKF User's Guide published by the DTC (http://www.dtcenter.org/EnKF/users/index.php).

The DRC primarily steers distributed GSI/EnKF development and community code management and support. The responsibilities of the 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 coding standard followed by all GSI/EnKF 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). Prospective contributors of GSI code should contact the DTC through the GSI helpdesk (*gsi_help@ucar.edu*). The DTC will help the prospective contributors go through a scientific review from the DRC to avoid any potential conflict in code

development. Upon the approval of the committee, the DTC will work with the prospective contributors in the preparation and integration of their code into the code repository.

About this GSI release:

As a critical part of the GSI user support, this GSI User's Guide is provided to assist users in applying GSI to data assimilation and analysis studies. It was composed by the DTC and reviewed by the DRC members. Please note the major focuses of the DTC are currently on testing and evaluation of GSI for regional Numerical Weather Prediction (NWP) applications though the instructions and cases for GSI global applications and RTMA applications are available with this release. This documentation describes the July 2015 version 3.4 release, which includes new capabilities and enhancements as well as bug fixes. This version of GSI is based on a revision of the community GSI repository from March 2015.

The GSI version 3.4 can be used either as a three-dimensional variational (3D-Var) system or a hybrid ensemble-3D-Var system. Both configurations have been running in daily operations (e.g., GFS, HWRF, RAP). Coupled with a forecast model and its adjoint model, GSI can be turned into a 4D-var system with embedded 4D-Var features (e.g., GEOS).

Observations used by this version GSI:

GSI is being used by various applications on multiple scales. Therefore, the types of observations GSI can assimilate vary from conventional to aerosol observations. Users should use observations with caution to fit their specific applications. The GSI version 3.4 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

What is new in this release version:

The following lists some of the new functions and changes included in the GSI release version 3.4 versus version 3.3:

- Sat wind thinning algorithm update
- Update to aircraft data usage:
  - Inflated observation error to the AIREP near surface data
  - Update to the variational aircraft temperature bias correction
- Update to soil adjustment
- Update to Cloud analysis
- Added capability to use surface data uselist
- Added NSST (Near-Surface Sea Temperature) calculation for SST analysis
- Code changes for building a 4D EnVar capability

- Added real single radiance observation test capability:
- 2D-Var surface analysis capability update:
    - Improved quality control for temperature observations via buddy check and a terrain-aware gross error adjustment
    - Use of new observations (e.g. sky cover data and METOP-B ASCAT winds)
    - New analysis variables (total cloud amount, lowest cloud base, 2m Td, min and max 2m T, significant wave height, pmsl, 10m wind speed)
    - Ability to perform separate analyses for most of analysis variables over land and water (and merge the two)
    - Retuning/adjustment of background errors
- Code cleanup and optimization, e.g., reducing memory usage and improving threading.
- Bug fixes and others

Please note due to the version update, some diagnostic files and static information files have been changed as well.

The structure of this User's Guide:

The User's Guide is organized as follows:

Chapter 1 provides a background introduction of GSI.

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

Chapter 3 focuses on the input files needed to run GSI and how to configure and run GSI through a sample run script. Also provides example of a successful GSI run and explanations of often used namelist variables.

Chapter 4 includes information about diagnostics and tuning of the GSI system through GSI standard output, statistic fit files, and some diagnostic tools.

Chapter 5 illustrates how to setup configurations to run GSI with conventional, radiance, and GPSRO data and how to diagnose the results. Also includes introductions to GSI hybrid application.

Appendix A introduces the community tools available for GSI users.

Appendix B is content of the GSI namelist section OBS_INPUT.

Appendix C contains a complete list of the GSI namelist with explanations and default values.

Chapter 2: Software Installation

2.1 Introduction

The DTC community GSI is a community distribution of NOAA's operational GSI. The community GSI expands the portability of the operational code by adding a flexible build system and providing example run scripts that allow GSI to be compiled and run on many common platforms. The current version of GSI is fully supported (e.g. builds and runs) on Linux platforms using the Intel and PGI compilers. Build rules are provided for the following non-supported systems: IBM AIX supercomputers using the xlf compiler, and Intel based Macintosh computers using the PGI compiler.

This chapter describes how to build and install the DTC community GSI on your computing resources. These instructions apply only to the DTC community GSI. While the community GSI source code is identical to NCEP's GSI trunk code when frozen for release, the community build system is more general in order to support a wide variety of computing platforms.

The GSI building process consists of four general steps:

- Obtain the source code for GSI and WRF.
- Build the WRF model (see the WRF users guide).
- Set the appropriate environment variables for the GSI build.
- Configure and compile the GSI source code.

This chapter is organized as follows: Section 2.2 describes how to obtain the source code. Section 2.3 covers the directory structure and supplemental NCEP libraries included with the distribution. Section 2.4 starts with an outline of the build example and then goes into a more detailed discussion of setting up the build environment and the configure and compile steps. Section 2.5 illustrates the build process for the two most common compilers (Intel and PGI) on the NCAR supercomputer Yellowstone. Section 2.6 covers the system requirements (tools, libraries, and environment variable settings) and currently supported platforms in detail. Section 2.7 discusses what to do if you have problems with the build and where to get help.

For beginning users, sections 2.2 and 2.4 provide the necessary steps to obtain the code and build GSI on most systems. The remainder of the chapter provides background needed for completeness. Advanced topics, such as customizing the build, porting to new platforms, and debugging can be found in the GSI advanced user's guide.

2.2 Obtaining and setting up the Source Code

The community GSI resources, including source code, build system, utilities, practice data, and documentation, are available from the DTC community GSI users website, located at

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

The source code is available by first selecting the ***Download*** tab on the vertical menu located on the left column of the page, and then selecting the ***GSI/EnKF System*** submenu. 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.tar` tarball to download the most recent version of the source code (as of July 2015). Selecting the newest release of the community GSI is critical for having the most recent capabilities, supplemental libraries, and bug fixes.

To analyze satellite radiance observations, GSI requires CRTM coefficients. Use **only** the version of the CRTM coefficients provided by GSI website, which are available as separate tarfiles. They can be downloaded by selecting the link to the ***CRTM 2.1.3 Big Endian coefficients tarball*** from the web page. For all compilers use the big endian byte order coefficients found in the first CRTM link.

The download page also contains links to two separate tarballs with the fixed files necessary for running two special configurations of GSI:

- Global configuration ***(fix files to run Global GSI)***
- RTMA (***fix files to run RTMA GSI***)

The community GSI version 3.4 comes in a tar file named `comGSIv3.4-EnKFv1.0.tar`. The tar file may be unpacked by using the 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 Directory Structure, Source code and Supplemental Libraries

The GSI system includes the GSI source code, build system, supplemental libraries, fixed files, and run scripts. The following table lists the system components found inside of the root GSI directory.

Directory Name	Content
src/main/	GSI source code and makefile
src/libs/	Source code for supplemental libraries
fix/	Fixed input files required by a GSI analysis, such as background error covariances, observation error tables; excluding CRTM coefficients
include/	Include files created by the build system
lib/	Contain compiled supplemental libraries, created by the build
run/	Directory for executable ***gsi.exe*** and sample run scripts
arch/	Build options and machine architecture specifics (see Advanced GSI User's Guide)
util/	Tools for GSI diagnostics

For the convenience of the user, supplemental NCEP libraries for building GSI are included in the src/libs/ directory. These libraries are built when GSI is built. These supplemental libraries are listed in the table below.

Directory Name	Content
bacio/	NCEP BACIO library
bufr/	NCEP BUFR library
crtm_2.1.3/	JCSDA community radiative transfer model v2.1.3
gsdcloud/	GSD Cloud analysis library
misc/	Misc support libraries
nemsio/	NEMS I/O library
sfcio/	NCEP GFS surface file i/o module
sigio/	NCEP GFS atmospheric file i/o module
sp/	NCEP spectral - grid transforms
w3emc_v2.0.5/	NCEP/EMC W3 library (date/time manipulation, GRIB)
w3nco_v2.0.6/	NCEP/NCO W3 library (date/time manipulation, GRIB)

The one "library" not included with the source code, are the WRF IO API's. Please note that the release version of WRF/EnKF has only been tested using the previous two release versions of WRF. Older versions of WRF may provide unpredictable results.

The WRF code, and full WRF documentation, can be obtained from the WRF Users Page,

http://www.mmm.ucar.edu/wrf/users/

following a registration process similar to that for downloading GSI.

2.4 Compiling GSI

This section starts with a quick outline of how to build GSI (2.4.1), followed by a more detailed discussion of the build process (2.4.2 & 2.4.3). Typically GSI will build "straight out of the box" on any system that successfully builds the WRF model. Should the user experience any difficulties with the default build, check the build environment against the requirements described at the end of section 2.6.

To proceed with the GSI build, it is assumed that the WRF model has already been built on the current system. GSI uses the WRF I/O API libraries to read the background file. These I/O libraries are created as part of the WRF build, and are linked into GSI during the GSI build process. In order to successfully link the WRF I/O libraries with the GSI source, it is crucial that both WRF and GSI are built using the same compilers. This means that if WRF is built with the Intel Fortran compiler, then GSI must also be built with the Intel Fortran compiler.

2.4.1 Build Overview

This section provides a quick outline of the steps necessary to build the GSI code. The following steps describe that build process.

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. **Set the environment variables:** The first path on this list will always need to be set. The remaining two will depend on your choice of compiler and how your default environment is configured.
    a. `WRF_DIR` the path to the compiled WRF directory (to always be set)
    b. `NETCDF` the path to the NETCDF libraries
    c. `LAPACK_PATH` the path to the LAPACK math libraries
3. **Run the configure script**
4. **Run the compile script**

2.4.2 Environment Variables

Before configuring the GSI code to be built, at least one, and no more than three environment variables must be set.

- **`WRF_DIR:`** defines the path to the root of the WRF build directory. Setting this is mandatory. This variable tells the GSI build system where to find the WRF I/O libraries. The process for setting the environment variables varies according to the login shell in use. To set the path variable `WRF_DIR` for csh/tcsh, type;

```
setenv WRF_DIR /path_to_WRF_root_directory/
```

for bash/ksh, the syntax is;

```
export WRF_DIR=/path_to_WRF_root_directory/
```

- **NETCDF** : The second environment variable specifies the local path to NetCDF library. The environment variable for NETCDF may be checked by "echo'ing" the variable using:

  ```
  echo $NETCDF
  ```

  If the command returns with the response that the variable is undefined, such as

  ```
  NETCDF: Undefined variable.
  ```

  it is then necessary to manually set this variable. If your system uses modules or a similar mechanism to set the environment, do this first. If a valid path is returned by the echo command, no further action is required.

- **LAPACK_PATH:** defines the path to the LAPACK library. Typically, this variable will only need to be set on systems without a vendor provided version of LAPACK. IBM systems typically come installed with the LAPACK equivalent ESSL library that links automatically. Likewise, the PGI compiler often comes with a vendor provided version of LAPACK that links automatically with the compiler. Experience has shown that the following situations make up the majority of cases where the LAPACK variable needed to be set:

  1. On stripped down versions of the IBM operating system that come without the ESSL libraries.
  2. Linux environments using Intel Fortran compiler.
  3. Building with Gfortran.
  4. On systems where the path variables are not properly set.

  Of the four, the second of these is the most common. The Intel compiler *usually* comes with a vendor provided mathematics library known as the "Mathematics Kernel Libraries" or MKL for short. While most installations of the Intel compiler typically come with the MKL libraries installed, the ifort compiler does not automatically load the library. It is therefore necessary to set the LAPACK_PATH variable to the location of the MKL libraries when using the Intel compiler. You may need to ask your system administrator for the correct path to these libraries.

On super-computing systems with multiple compiler options, these variables may be set as part of the module settings for each compiler. On the NCAR supercomputer Yellowstone, the Intel build environment can be specified through setting the appropriate modules. When this is done, the MKL library path is available through a local environment variable, MKLROOT. The LAPACK environment may be set for csh/tcsh with the Unix commands

```
setenv LAPACK_PATH $MKLROOT
```

and for bash/ksh by

```
export LAPACK_PATH=$MKLROOT
```

Once the environment variables have been set, the next step in the build process is to first run the configure script and then the compile script.

2.4.3 Configure and Compile

Once the environment variables have been set, building the GSI source code requires two additional steps:

1. Run the configure script and select a compiler option.
2. Run the compile script

Change into the `comGSIv3.4-EnKFv1.0/` directory and issue the configure command:

```
./configure
```

The `./configure` command uses user input to create a platform specific configuration file called `configure.gsi`. The script starts by echoing the `NETCDF` and `WRF_DIR` paths set in the previous section. It then examines the current system and queries the user to select from multiple build options.

For 64-bit Linux the options will be the following:

```
   ------------------------------------------------------------------------
   Please select from among the following supported platforms.

    1.  Linux x86_64, PGI compilers (pgf90 & pgcc)  (dmpar,optimize)
    2.  Linux x86_64, PGI compilers (pgf90 & gcc)  (dmpar,optimize)
    3.  Linux x86_64, GNU compilers (gfortran & gcc)  (dmpar,optimize)
    4.  Linux x86_64, Intel/gnu compiler (ifort & gcc)    (dmpar,optimize)
    5.  Linux x86_64, Intel compiler (ifort & icc)  (dmpar,optimize)
    6.  Linux x86_64, Intel compiler (ifort & icc), IBM POE (EXPERIMENTAL)
(dmpar,optimize)
    7.  Linux x86_64, Intel compiler (ifort & icc), SGI MPT (EXPERIMENTAL)
(dmpar,optimize)

Enter selection [1-7] :
```

Looking at the list, there are two things to note. First is that the GNU C-compiler (gcc) may be paired with other compilers. This allows the build to use the GNU C-compiler in place of the Intel (icc) or PGI (pgcc) C-compiler.

The second thing to notice is that there are separate build targets for vendor supplied versions of MPI such as IBM POE and SGI MPT. This was added due to some computing

hardware vendors creating non-standard `mpif90` wrappers for their vendor supplied version of MPI. If uncertain about which to choose, select the default option first. If that option fails with an error referencing a bad argument for `mpif90` then try the option with "vendor supplied MPI."

On selecting an option, the process reports a successful configuration with the banner:

```
------------------------------------------------------------------------
Configuration successful. To build the GSI, type: compile
------------------------------------------------------------------------
```

Failure to get this banner means that the configuration step failed to complete. The most typical reason for a failure is an error in one of the paths set to the environment variables.

After selecting a build option, run the compile script:

```
./compile >& compile.log
```

It is recommended to capture the build information to a log file by redirecting the output.

To conduct a complete clean, which removes ALL built files in ALL directories, as well as the `configure.gsi`, type:

```
./clean -a
```

A complete clean is necessary if the compilation failed or if the configuration file is changed.

Following a successful compile, the GSI executable `gsi.exe` can be found in the `run/` directory. If the executable is not found, check the compilation log file. If the build failed, search for the first instance of the word "Error" with a capital "E" to locate the section of the log with the failure.

2.5 Example of Build

To illustrate the build process, the following section describes the steps necessary to build GSI on the NCAR supercomputer Yellowstone using both the Intel compiler and then the PGI compiler. Other platforms will be similar.

2.5.1 Intel Build

Steps to build GSI on Yellowstone using the Intel compiler:

1. Select the Intel compiler environment by using the module commands:

   ```
   module load intel
   ```

```
module load impi mkl ncarcompilers ncarbinlibs netcdf
```

These module commands have specified the compiler, mpi, the version of the LAPACK library (MKL) and the netcdf library.

2. For this case two of the paths must be set. The path to the WRF directory must always be specified, and the Intel Mathematics Kernal Library (MKL) will be used in place of the LAPACK library. Note that on Yellowstone, the variable `MKLROOT` is set to the path to the MKL libraries by loading the mkl module. To set the paths in a C-shell environment use:

```
setenv WRF_DIR /PATH TO WRF DIRECTORY/
setenv LAPACK_PATH $MKLROOT
```

3. To run the configure script, type `./configure` inside the top of the GSI directory. If the first three steps were completed successfully, a table of compiler options should appear. Select the desired compiler combination, which in this case is either 4 or 5. The alternative options (6 & 7) are needed for certain platforms that have a vendor supplied custom version of MPI. Try the default build options for MPI first, and only if it fails should the second option be used.

4. To compile the code, enter in a C-shell: `./compile >& compile.log`. If the build completes successfully, an executable named `gsi.exe will` be created in the `./run` directory.

2.5.2 PGI Build

Steps to build GSI on Yellowstone using the PGI compiler:

1. The PGI compiler environment is selected using the module commands:

```
module load pgi
module load impi ncarcompilers ncarbinlibs netcdf
```

These module commands have specified the compiler, mpi, and the netcdf library.

2. For this case only the path to the WRF directory must be set. The PGI compiler comes with its own version of LAPACK that it finds automatically. It is not necessary to set the `LAPACK` path. In a C-shell environment use:

```
setenv WRF_DIR /PATH TO WRF DIRECTORY/
```

3. Similar to the Intel example, pick compiler options listed in a table. In this case, the desired compiler combination is either 1 or 2.
4. To compile the code, enter in a C-shell: `./compile >& compile.log`. If the build completes successfully, an executable named `gsi.exe will` be created in the `./run` directory.

2.6 System Requirements and External Libraries

The source code for GSI is written in FORTRAN, FORTRAN 90, and C. In addition, the parallel executables require some flavor of MPI and OpenMP for the distributed memory parallelism. Lastly the I/O relies on the NetCDF I/O libraries. Beyond standard shell scripts, the build system relies on the Perl scripting language and makefiles.

The basic requirements for building and running the GSI system are the following:

- FORTRAN 95+ compiler
- C compiler
- MPI v1.2+
- OpenMP
- Perl
- NetCDF V3.6.3 or V4.2+
- LAPACK and BLAS mathematics libraries, or equivalent
- WRF V3.5+

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.

2.6.1 Compilers tested for release

Version 3.4 of the DTC community GSI system has been successfully tested on a variety of Linux platforms with many versions of the Intel and PGI fortran compilers.

Legacy build rules are also available for IBM AIX and Mac Darwin platforms. Because the DTC does not have the ability to test on these plaforms, they are no longer supported. Also, Linux GNU gfortran option is added in this version.

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 13.10,13.2,13.9,14.10,14.3, 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.

2.7 Getting Help and Reporting Problems

Should the user experience any difficulty building GSI on their system, please first confirm that all the required software is properly installed (section 2.4). Next check that the external libraries exist and that their paths are correct. Lastly, check the resource file `configure.gsi` for errors in any of the paths or settings. Should all these check out, feel free to contact the community GSI supporters at gsi_help@ucar.edu for assistance.

At a minimum, when reporting code building problems to the helpdesk, please include with your email a copy of the build log, and the contents of the `configure.gsi` file.

Chapter 3: Running GSI

This chapter discusses the issues of running GSI. It starts with introductions to the input data required to run GSI. Then proceeds with a detailed explanation of an example GSI run script and introductions to the result files produced by a successful GSI run. It concludes with some frequently used options from the GSI namelist.

3.1 Input Data Required to Run GSI

In most cases, three types of input data (background, observation, and fixed files) must be available before running GSI. In some special idealized cases, such as a pseudo single observation test, GSI can be run without any observations. If running GSI with 3DVAR-Ensemble hybrid option, global or regional ensemble forecasts are also needed.

- **Background or first guess field**

  As with other data analysis systems, the background or first guess fields may come from a model forecast conducted separately or from a previous data assimilation cycle. The following is a list of the types of background files that can be used by this release version of GSI:

  a) WRF NMM input fields in binary format
  b) WRF NMM input fields in NetCDF format
  c) WRF ARW input fields in binary format
  d) WRF ARW input fields in NetCDF format
  e) GFS input fields in binary format
  f) NEMS-NMMB input fields
  g) RTMA input files (2-dimensional binary format)

  The WRF is a community model system, including two dynamical cores: the Advanced Research WRF (ARW) and the nonhydrostatic Mesoscale Model (NMM). The GFS (Global Forecast System), NEMS (National Environmental Modeling System)-NMMB (Nonhydrostatic Mesoscale Model B-Grid), and RTMA (Real-Time Mesoscale Analysis) are operational systems of NCEP. The DTC mainly supports GSI for regional community model WRF. Therefore, most of the multiple platform tests were conducted using WRF netcdf background files (b, d). The DTC also supports the GSI in global and RTMA applications with limited resources. The following backgrounds have been tested for the release:

  1. NMM NetCDF (b) and ARW NetCDF (d) were tested with multiple cases
  2. GFS (e) was tested with multiple NCEP cases
  3. RTMA (g) was tested with a single case
  4. NEMS-NMMB(f) was tested with a single case.

- **Observations**

  GSI can analyze many types of observational data, including conventional data, satellite radiance observations, GPS, and radar data et al. The default observation file names given in released GSI namelist, the corresponding observations included in each files and sample BUFR files downloaded from the NCEP website are listed in table 3.1 on the next page.

  The observations are complex and many observations need format converting and quality control before being used by GSI. GSI ingests observations saved in the BUFR format (with NCEP specified features). The NCEP processed PrepBUFR and BUFR files can be used directly. If users need to introduce their own data into GSI, please check the following website for BUFR/PreBUFR processing User's Guide and exmaples:

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

  DTC supports data BUFR/PrepBUFR processing and quality control as part of GSI community tasks.

  GSI can analyze all of the data types in table 3.1, but each GSI run (for both operation or case study) only uses a subset of the data. Some data may be outdated and not available, some are on monitoring mode, and some data may have quality issues during certain periods. Users are encouraged to check the data quality issues prior to running an analysis. The following NCEP links provide resources that include data quality history:

  http://www.emc.ncep.noaa.gov/mmb/data_processing/Satellite_Historical_Documentation.htm
  http://www.emc.ncep.noaa.gov/mmb/data_processing/Non-satellite_Historical_Documentation.htm

  Because the current regional models do not have ozone as a prognostic variable, ozone data are not assimilated on the regional scale.

  GSI can be run without any observations to see how the moisture constraint modifies the first guess (background) field. GSI can be run in a pseudo single observation mode, which does not require any BUFR observation files. In this mode, users should specify observation information in the namelist section `SINGLEOB_TEST` (see Section 4.2 for details). As more data files are used, additional information will be added through the GSI analysis.

Tabel 3.1 GSI observation file name, content, and examples

GSI Name	Content	Example file names
prepbufr	Conventional observations, including ps, t, q, pw, uv, spd, dw, sst	gdas1.t12z.prepbufr.nr
satwndbufr	satellite winds observations	gdas1.t12z.satwnd.tm00.bufr_d
amsuabufr	AMSU-A 1b radiance (brightness temperatures) from satellites NOAA-15, 16, 17,18, 19 and METOP-A/B	gdas1.t12z.1bamua.tm00.bufr_d
amsubbufr	AMSU-B 1b radiance (brightness temperatures) from satellites NOAA-15, 16,17	gdas1.t12z.1bamub.tm00.bufr_d
radarbufr	Radar radial velocity Level 2.5 data	ndas.t12z. radwnd. tm12.bufr_d
gpsrobufr	GPS radio occultation and bending angle observation	gdas1.t12z.gpsro.tm00.bufr_d
ssmirrbufr	Precipitation rate observations from SSM/I	gdas1.t12z.spssmi.tm00.bufr_d
tmirrbufr	Precipitation rate observations from TMI	gdas1.t12z.sptrmm.tm00.bufr_d
sbuvbufr	SBUV/2 ozone observations from satellite NOAA-16, 17, 18, 19	gdas1.t12z.osbuv8.tm00.bufr_d
hirs2bufr	HIRS2 1b radiance from satellite NOAA-14	gdas1.t12z.1bhrs2.tm00.bufr_d
hirs3bufr	HIRS3 1b radiance observations from satellite NOAA-16, 17	gdas1.t12z.1bhrs3.tm00.bufr_d
hirs4bufr	HIRS4 1b radiance observation from satellite NOAA-18, 19 and METOP-A/B	gdas1.t12z.1bhrs4.tm00.bufr_d
msubufr	MSU observation from satellite NOAA 14	gdas1.t12z.1bmsu.tm00.bufr_d
airsbufr	AMSU-A and AIRS radiances from satellite AQUA	gdas1.t12z.airsev.tm00.bufr_d
mhsbufr	Microwave Humidity Sounder observation from NOAA-18, 19 and METOP-A/B	gdas1.t12z.1bmhs.tm00.bufr_d
ssmitbufr	SSMI observation from satellite f13, f14, f15	gdas1.t12z.ssmit.tm00.bufr_d
amsrebufr	AMSR-E radiance from satellite AQUA	gdas1.t12z.amsre.tm00.bufr_d
ssmisbufr	SSMIS radiances from satellite f16	gdas1.t12z.ssmis.tm00.bufr_d
gsnd1bufr	GOES sounder radiance (sndrd1, sndrd2, sndrd3 sndrd4) from GOES-11, 12, 13, 14, 15.	gdas1.t12z.goesfv.tm00.bufr_d
l2rwbufr	NEXRAD Level 2 radial velocity	ndas.t12z.nexrad. tm12.bufr_d
gsndrbufr	GOES sounder radiance from GOES-11, 12	gdas1.t12z.goesnd.tm00.bufr_d
gimgrbufr	GOES imager radiance from GOE- 11, 12	
omibufr	Ozone Monitoring Instrument (OMI) observation NASA Aura	gdas1.t12z.omi.tm00.bufr_d
iasibufr	Infrared Atmospheric Sounding Interfero-meter sounder observations from METOP-A/B	gdas1.t12z.mtiasi. tm00.bufr_d
gomebufr	The Global Ozone Monitoring Experiment (GOME) ozone observation from METOP-A/B	gdas1.t12z.gome.tm00.bufr_d
mlsbufr	Aura MLS stratospheric ozone data from Aura	gdas1.t12z. mlsbufr.tm00.bufr_d
tcvitl	Synthetic Tropic Cyclone-MSLP observation	gdas1.t12z.syndata.tcvitals.tm00
seviribufr	SEVIRI radiance from MET-08,09,10	gdas1.t12z. sevcsr.tm00.bufr_d
atmsbufr	ATMS radiance from Suomi NPP	gdas1.t12z. atms.tm00.bufr_d
crisbufr	CRIS radiance from Suomi NPP	gdas1.t12z. cris.tm00.bufr_d
modisbufr	MODIS aerosol total column AOD observations from AQUA and TERRA	

- **Fixed file (statistics and control files)**

  A GSI analysis also needs to read specific information from statistic files, configuration files, bias correction files, and CRTM coefficient files. We refer to these files as fixed files and they are located in a directory called *fix/* in the release package, except for CRTM coefficients.

  Table 3.2 lists fixed files required in a GSI run, the content of the files, and corresponding example files from the regional, global and RTMA applications:

Table 3.2 GSI fixed files, content, and examples

File name used in GSI	Content	Example files in *fix/*
anavinfo	Information file to set control and analysis variables	anavinfo_arw_netcdf anavinfo_ndas_netcdf global_anavinfo.l64.txt anavinfo_rtma_gust_vis_7vars
berror_stats	background error covariance	nam_nmmstat_na.gcv nam_glb_berror.f77.gcv global_berror.l64y386.f77 new_rtma_regional_nmm_berror.f77.gcv
errtable	Observation error table	nam_errtable.r3dv prepobs_errtable.global
Observation data control file (more detailed explanation in Section 4.3)		
convinfo	Conventional observation information file	global_convinfo.txt nam_regional_convinfo.txt new_rtma_regional_convinfo.txt
satinfo	satellite channel information file	global_satinfo.txt
pcpinfo	precipitation rate observation information file	global_pcpinfo.txt
ozinfo	ozone observation information file	global_ozinfo.txt
Bias correction and Rejection list		
satbias_angle	satellite scan angle dependent bias correction file	global_satangbias.txt
satbias_in	satellite mass bias correction coefficient file	sample.satbias
	combined satellite angle dependent and mass bias correction coefficient file	gdas1.t00z.abias.new
t_rejectlist, w_rejectlist,..	Rejetion list for T, wind, et al. in RTMA	new_rtma_t_rejectlist new_rtma_w_rejectlist

Because most of those fixed files have hardwired names inside the GSI, a GSI run script needs to copy or link those files (right column in table 3.2) from ./fix directory to GSI run directory with the file name required in GSI (left column in table 3.2). For

example, if GSI runs with ARW background case, the following line should be in the run script:

```
cp ${path of the fix directory}/anavinfo_arw_netcdf   anavinfo
```

Note that for this release, there is a strict rule that the numbers of vertical levels in the file `anavinfo` must match the background file (for example, `wrfinput_d01`) for the 3-dimemsional variables. Otherwise GSI will fail. To find out the correct numbers of vertical levels, users can dump out (use *ncdump -h*) the dimensions from the NetCDF background file and find the number for `bottom_top` and `bottom_top_stag`. For example, if the dimemsions for the background file is:

```
bottom_top = 50 ;
bottom_top_stag = 51 ;
```

Then the corresponding `anavinfo` file should have 51 levels for prse (3-dimemsional pressure field) and 50 levels for other three-dimemsional variables such as u, v, tv, q, oz, cw and etc. For details, users can dump out the global attributes of the background file and find the number of vertical levels for each variable. The following shows part of `anavinfo` for the above background:

```
state_derivatives::
!var  level  src
 ps   1      met_guess
 u    50     met_guess
 v    50     met_guess
 tv   50     met_guess
 q    50     met_guess
 oz   50     met_guess
 cw   50     met_guess
 prse 51     met_guess
::
```

Each operational system, such as GFS, NAM, RAP, and RTMA, has their own set of fixed files. Therefore, for each fixed file used in GSI, there are several corresponding fixed files in the directory *fix/* that users can choose. For example, for the background error covariance file, both *nam_nmmstat_na.gcv* (from the NAM system) and *regional_glb_berror.f77.gcv* (from the global forecast system) can be used. We also prepared the same background error covariance files with different byte order such as files under *./fix/Little_Endian* and *./fix/Big_Endian* directory. To help users to setup these fixed files for different GSI applications, several sample run scripts are provided with the release version.

To make ./fix directory easy to manage, this release version created 5 sub-directories to hold special group of fix files, which are introduced in table 3.3.

Table 3.3 List of sub-directories in fix directory

Directory name	Content
Little_Endian	Little Endian Bacground Error covariance (BE) files

Big_Endian	Big Endian BE files
global	Global BE files and ch4, co, co2, n2o history files
rtma	Fix files for GSI RTMA application, including both Big_endian and Little_endian files
rap	Fix files for GSI RAP application

Please note released comGSIv3.4_EnKFv1.0 tar files doesn't include ./fix/global, ./fix/Little_Endian and ./fix/rtma for space saving. Please downloand comGSIv3.4_EnKFv1.0_fix_global.tar.gz if you need to run global case, comGSIv3.4_EnKFv1.0_fix_Little_Endian.tar.gz if you need Little_endian BE files, and comGSIv3.4_EnKFv1.0_fix_rtma.tar.gz to run RTMA case from the GSI user's webpage.

Each release version GSI calls certain version of CRTM library and needs the corresponding version of CRTM coefficients to do radiance data assimilation. This version of GSI uses CRTM 2.1.3. The coefficients files are listed in table 3.4.

Table 3.4 List of radiance coefficients used by CRTM

File name used in GSI	Content	Example files
Nalli.IRwater.EmisCoeff.bin NPOESS.IRice.EmisCoeff.bin NPOESS.IRsnow.EmisCoeff.bin NPOESS.IRland.EmisCoeff.bin NPOESS.VISice.EmisCoeff.bin NPOESS.VISland.EmisCoeff.bin NPOESS.VISsnow.EmisCoeff.bin NPOESS.VISwater.EmisCoeff.bin FASTEM5.MWwater.EmisCoeff.bin	IR surface emissivity coefficients	Nalli.IRwater.EmisCoeff.bin NPOESS.IRice.EmisCoeff.bin NPOESS.IRsnow.EmisCoeff.bin NPOESS.IRland.EmisCoeff.bin NPOESS.VISice.EmisCoeff.bin NPOESS.VISland.EmisCoeff.bin NPOESS.VISsnow.EmisCoeff.bin NPOESS.VISwater.EmisCoeff.bin FASTEM5.MWwater.EmisCoeff.bin
AerosolCoeff.bin	Aerosol coefficients	AerosolCoeff.bin
CloudCoeff.bin	Cloud scattering and emission coefficients	CloudCoeff.bin
${satsen}.SpcCoeff.bin	Sensor spectral response characteristics	${satsen}.SpcCoeff.bin
${satsen}.TauCoeff.bin	Transmittance coefficients	${satsen}.TauCoeff.bin

3.2 GSI Run Script

In this release version, two sample run scripts are available for different GSI applications:

- *comGSIv3.4_EnKFv1.0/run/**run_gsi_regional.ksh*** for regional GSI
- *comGSIv3.4_EnKFv1.0/run/**run_gsi_global.ksh*** for global GSI (GFS)

These scripts will be called to generate GSI namelists:

- *comGSIv3.4_EnKFv1.0/run/comgsi_namelist.sh* for regional GSI
- *comGSIv3.4_EnKFv1.0/run/comgsi_namelist_gfs.sh* for global GSI (GFS)

We will introduce the regional run scripts (*run_gsi_regional.ksh**)*** in detail in the following sections and introduce the global run script when we introduce the GSI global application in Advanced GSI User's Guide.

Note there is also a run script for regional EnKF (*run_enkf_wrf.ksh*), a run script for global EnKF (*run_enkf_global.ksh*) and the EnKF namelist script (*enkf_wrf_namelist.sh*) in the same directory, which will be introduced in the EnKF users guide.

3.2.1 Steps in the GSI run script

The GSI run script creates a run time environment necessary for running the GSI executable. A typical GSI run script includes the following steps:

1. Request computer resources to run GSI.
2. Set environmental variables for the machine architecture.
3. Set experimental variables (such as experiment name, analysis time, background, and observation).
4. Set the script that generates the GSI namelist.

5. Check the definitions of required variables.
6. Generate a run directory for GSI (sometime called working or temporary directory).
7. Copy the GSI executable to the run directory.
8. Copy the background file to the run directory and create a index file listing the location and name of ensemble members if running the hybrid.
9. Link observations to the run directory.
10. Link fixed files (statistic, control, and coefficient files) to the run directory.
11. Generate namelist for GSI.

12. Run the GSI executable.
13. Post-process: save analysis results, generate diagnostic files, clean run directory.

14. Run GSI as observation operator for EnKF, only for `if_observer=Yes`.

Typically, users only need to modify specific parts of the run script (steps 1, 2, and 3) to fit their specific computer environment and point to the correct input/output files and directories. Users may also need to modify step 4 if changes are made to the namelist and it is under a different name or at a different location. Next section (3.2.2) covers each of these modifications for steps 1 to 3. Section 3.2.3 will dissect a sample regional GSI run script and introduce each piece of this sample GSI run script. Users should start with the run script provided in the same release package with GSI executable and modify it for their own run environment and case configuration.

3.2.2 Customization of the GSI run script

3.2.2.1 Setting up the machine environment

This section focuses on step 1 of the run script: modify the machine specific entries. Specifically, this consists of setting Unix/Linux environment variables and selecting the correct parallel run time environment (batch system with options).

GSI can be run with the same parallel environments as other MPI programs, for example:

- IBM supercomputer using LSF (*Load Sharing Facility*)
- IBM supercomputer using LoadLevel
- Linux clusters using PBS (*Portable Batch System*)
- Linux clusters using LSF
- Linux workstation (*with no batch system*)
- Intel Mac Darwin workstation with PGI complier (*with no batch system*)

Two queuing systems are listed below as examples:

Machine and queue system	Linux Cluster with LSF	Linux Cluster with PBS	Workstation

example	#BSUB -P ????????
	#BSUB -W 00:10
	#BSUB -n 4
	#BSUB -R "span[ptile=16]
	#BSUB -J gsi
	#BSUB -o gsi.%J.out
	#BSUB -e gsi.%J.err
	#BSUB -q small
		`#PBS -l procs=4`
		`#PBS -n`
		`#PBS -o gsi.out`
		`#PBS -e gsi.err`
		`#PBS -N GSI`
		`#PBS -l walltime=00:20`
		`#PBS -A ??????`
			No batch system,
			
			skip this step

In both of the examples above, environment variables are set specifying system resource management, such as the number of processors, the name/type of queue, maximum wall clock time allocated for the job, options for standard out and standard error, etc. Some

platforms need additional definitions to specify Unix environmental variables that further define the run environment.

These variable settings can significantly impact the GSI run efficiency and accuracy of the GSI results. Please check with your system administrator for the optimal settings for your computer system. Note that while the GSI can be run with any number of processors, it will not scale well with the increase of processor numbers after a certain threshold based on the case configuration and GSI application types.

3.2.2.2 Setting up the running environment

There are only two options to define in this block.

```
# GSIPROC = processor number used for GSI analysis
#-------------------------------------------------
  GSIPROC=8
  ARCH='LINUX_LSF'
# Supported configurations:
            # IBM_LSF,
            # LINUX, LINUX_LSF, LINUX_PBS,
            # DARWIN_PGI
```

The option `ARCH` selects the machine architecture. It is a function of platform type and batch queuing system. The option GSIPROC sets the number of cores used in the run. This option also decides if the job is run as a multiple core job or as a single core run. Several choices of the option ARCH are listed in the sample run script. Please check with your system administrator about running parallel MPI jobs on your system.

Option `ARCH`	Platform	Compiler	batch queuing system
`IBM_LSF`	IBM AIX	xlf, xlc, …	LSF
`LINUX`	Linux workstation	Intel/PGI/GNU	mpirun if `GSIPROC > 1`
`LINUX_LSF`	Linux cluster	Intel/PGI/GNU	LSF
`LINUX_PBS`	Linux cluster	Intel/PGI/GNU	PBS
`DARWIN_PGI`	MAC DARWIN	PGI	mpirun if `GSIPROC > 1`

3.2.2.3 Setting up an analysis case

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

```
##################################################
```

```
# 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=2014061700
  WORK_ROOT=./comGSIv3.4_EnKFv1.0/run/testgsi
  OBS_ROOT=./obs
  PREPBUFR=./obs/gdas1.t00z.prepbufr.nr
  BK_FILE=./wrfinput_d01_ARW_2014-06-17_00
  FIX_ROOT=./comGSIv3.4_EnKFv1.0/fix
  CRTM_ROOT=./CRTM_REL-2.1.3
  GSI_EXE=./comGSIv3.4_EnKFv1.0/run/gsi.exe
  GSI_NAMELIST=./comGSIv3.4_EnKFv1.0/run/comgsi_namelist.sh
```

When picking the observation BUFR files, a few cautions to be aware of are:

- GSI run will stop if the time in the background file cannot match the cycle time in the observation BUFR file used for the GSI run (there is a namelist option to turn this check off).
- Even if their contents are identical, PrepBUFR/BUFR files will differ if they were created on platforms with different endian byte order specification (Linux vs. IBM). If users obtain PrepBUFR/BUFR files from NCEP, these files must be converted before they can be used on a Linux system with gfortran compiler. Appendix A.1 discusses the conversion tool *ssrc* to byte-swap observation files. Since the release version 3.2. GSI compiled with PGI and Intel can automatically handle the byte order issue in PrepBUFR and BUFR files. Users can directly link any order BUFR file if working with Intel and PGI platform.

The next part of this block focuses on additional options that specify important aspects of the GSI configuration.

```
# 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=ARW
  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=No
```

Option `bk_core` indicates the specific dynamic core used to create the background files and is used to specify the core in the namelist. For this version, there is another `bk_core` option – NMMB in addition to WRF ARW and NMM, and also an option `if_observer=Yes` to

run GSI as observation operator for EnKF. Option `bkcv_option` specifies the background error covariance to be used in the case. Two regional background error covariance matrices are provided with the release, one from NCEP global data assimilation (GDAS), and one from NAM data assimilation system (NDAS). Please check Section 4.8 for more details about GSI background error covariance. Option `if_clean` is to tell the run script if it needs to delete temporal intermediate files in the working directory after a GSI run is completed. Option `if_observer` is to tell the run script if it needs to run GSI as observation operator for EnKF.

In most of case after the following point, users should only make minor changes:

```
  ####################################################
  # Users should NOT change script after this point
  ####################################################
  #
  BYTE_ORDER=Big_Endian
  # BYTE_ORDER=Little_Endian
```

But if GSI is running with gfortran compiler, the above byte order option should be set as "Little_Endian".

3.2.3 Description of the sample regional run script to run GSI

Listed below is an annotated regional run script (`Courier New`) with explanations on each function block.

For further details on the first 3 blocks of the script that users need to change, check section 3.2.2.1, 3.2.2.2, and 3.2.2.3:

```
#!/bin/ksh
#####################################################
# machine set up (users should change this part)
#####################################################
#
#
#run the executable
set -x

#
# GSIPROC = processor number used for GSI analysis
#-------------------------------------------------
  GSIPROC=8
  ARCH='LINUX_LSF'
# Supported configurations:
            # IBM_LSF,
            # LINUX, LINUX_LSF, LINUX_PBS,
            # DARWIN_PGI
#
####################################################
# 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=2014061700
  WORK_ROOT=./comGSIv3.4_EnKFv1.0/run/testgsi
  OBS_ROOT=./obs
  PREPBUFR=./obs/gdas1.t00z.prepbufr.nr
  BK_FILE=./wrfinput_d01_ARW_2014-06-17_00
  FIX_ROOT=./comGSIv3.4_EnKFv1.0/fix
  CRTM_ROOT=./CRTM_REL-2.1.3
  GSI_EXE=./comGSIv3.4_EnKFv1.0/run/gsi.exe
  GSI_NAMELIST=./comGSIv3.4_EnKFv1.0/run/comgsi_namelist.sh



#-----------------------------------------------
# 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=ARW
  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=No
```

At this point, users should be able to run the GSI for simple cases without changing the scripts. However, some advanced users may need to change some of the following blocks for special applications, such as use of radiance data, cycled runs, specifying certain namelist variables, or running GSI on a platform not tested by the DTC.

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

The next block sets run command to run GSI on multiple platforms. The `ARCH` is set in the beginning of the script. Option `BYTE_ORDER` has been as "Big_Endian" because GSI compiled with Intel and PGI can read in Big_Endian background error file, BUFR file and CRTM coefficient files. This option only needs to be set as "Little_Endian" when GSI compiled with gfortran.

```
  BYTE_ORDER=Big_Endian
  # BYTE_ORDER=Little_Endian

  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} " ;;
      '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 next block checks if all the variables needed for a GSI run are properly defined. These variables should have been defined in the first 3 parts of this script.

```
###############################################################
# Check GSI needed environment variables are defined and exist
#

# Make sure ANAL_TIME is defined and in the correct format
if [ ! "${ANAL_TIME}" ]; then
  echo "ERROR: \$ANAL_TIME is not defined!"
  exit 1
fi

# Make sure WORK_ROOT is defined and exists
if [ ! "${WORK_ROOT}" ]; then
  echo "ERROR: \$WORK_ROOT is not defined!"
  exit 1
fi

# Make sure the background file exists
if [ ! -r "${BK_FILE}" ]; then
  echo "ERROR: ${BK_FILE} does not exist!"
  exit 1
fi

# Make sure OBS_ROOT is defined and exists
if [ ! "${OBS_ROOT}" ]; then
  echo "ERROR: \$OBS_ROOT is not defined!"
  exit 1
fi
if [ ! -d "${OBS_ROOT}" ]; then
  echo "ERROR: OBS_ROOT directory '${OBS_ROOT}' does not exist!"
  exit 1
```

```
fi
# Set the path to the GSI static files
if [ ! "${FIX_ROOT}" ]; then
  echo "ERROR: \$FIX_ROOT is not defined!"
  exit 1
fi
if [ ! -d "${FIX_ROOT}" ]; then
  echo "ERROR: fix directory '${FIX_ROOT}' does not exist!"
  exit 1
fi

# Set the path to the CRTM coefficients
if [ ! "${CRTM_ROOT}" ]; then
  echo "ERROR: \$CRTM_ROOT is not defined!"
  exit 1
fi
if [ ! -d "${CRTM_ROOT}" ]; then
  echo "ERROR: fix directory '${CRTM_ROOT}' does not exist!"
  exit 1
fi

# Make sure the GSI executable exists
if [ ! -x "${GSI_EXE}" ]; then
  echo "ERROR: ${GSI_EXE} does not exist!"
  exit 1
fi

# Check to make sure the number of processors for running GSI was specified
if [ -z "${GSIPROC}" ]; then
  echo "ERROR: The variable $GSIPROC must be set to contain the number of
   processors to run GSI"
  exit 1
fi
```

The next block creates a working directory (`workdir`**)** in which GSI 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 GSI.

```
  ###################################################################
  # Create the ram work directory and cd into it

  workdir=${WORK_ROOT}
  echo " Create working directory:" ${workdir}

  if [ -d "${workdir}" ]; then
    rm -rf ${workdir}
  fi
  mkdir -p ${workdir}
  cd ${workdir}
```

After creating a working directory, copy the GSI executable, background, observation, and fixed files into the working directory. If hybrid is chosen, create an index file listing the location and names of ensemble members in the working directory (details in Section 5.4).

```
###################################################################

echo " Copy GSI executable, background file, and link observation bufr to
working directory"
```

```
# Save a copy of the GSI executable in the workdir
cp ${GSI_EXE} gsi.exe
```

Note: Copy the background file to the working directory as `wrf_inout`. The file `wrf_inout` will be overwritten by GSI to save analysis result.

```
# Bring over background field (it's modified by GSI so we can't link to it)
cp ${BK_FILE} ./wrf_inout
```

Note: You can link observation files to the working directory because GSI will not overwrite these files. The observations that can be analyzed in GSI are listed in the column `dfile` of the GSI namelist section `OBS_INPUT`, as specified in *run/comgsi_namelist.sh*. Most of the conventional observations are in one single file named `prepbufr`, while different radiance data are in separate files based on satellite instruments, such as AMSU-A or HIRS. All these observation files must be linked as GSI recognized file names in `dfile`. Please check table 3.1 for a detailed explanation of links and the meanings of each file name listed below.

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

# Link to the radiance data
# ln -s ${OBS_ROOT}/gdas1.t12z.1bamua.tm00.bufr_d amsuabufr
# ln -s ${OBS_ROOT}/gdas1.t12z.1bamub.tm00.bufr_d amsubbufr
# ln -s ${OBS_ROOT}/gdas1.t12z.1bhrs3.tm00.bufr_d hirs3bufr
# ln -s ${OBS_ROOT}/gdas1.t12z.1bhrs4.tm00.bufr_d hirs4bufr
# ln -s ${OBS_ROOT}/gdas1.t12z.1bmhs.tm00.bufr_d mhsbufr
# ln -s ${OBS_ROOT}/gdas1.t12z.gpsro.tm00.bufr_d gpsrobufr
```

The following block copies constant fixed files from the *fix/* directory and links CRTM coefficients. Please check Section 3.1 for the meanings of each fixed file.

```
###############################################################################

echo " Copy fixed files and link CRTM coefficient files to working
directory"

# Set fixed files
#   berror   = forecast model background error statistics
#   specoef  = CRTM spectral coefficients
#   trncoef  = CRTM transmittance coefficients
#   emiscoef = CRTM coefficients for IR sea surface emissivity model
#   aerocoef = CRTM coefficients for aerosol effects
#   cldcoef  = CRTM coefficients for cloud effects
#   satinfo  = text file with information about assimilation of brightness
temperatures
#   satangl  = angle dependent bias correction file (fixed in time)
#   pcpinfo  = text file with information about assimilation of
prepcipitation rates
#   ozinfo   = text file with information about assimilation of ozone data
#   errtable = text file with obs error for conventional data (regional
only)
#   convinfo = text file with information about assimilation of conventional
data
#   bufrtable= text file ONLY needed for single obs test (oneobstest=.true.)
#   bftab_sst= bufr table for sst ONLY needed for sst retrieval
(retrieval=.true.)
```

Note: For background error covariances, observation errors, and analysis available information files, we provide two sets of fixed files here, one set is based on GFS statistics and another is based on NAM statistics. For this release there is an additional setting of the `ANAVINFO` file for `bk_core=NMMB` for both GFS and NAM statistics.

```
if [ ${bkcv_option} = GLOBAL ] ; then
  echo ' Use global background error covariance'
  BERROR=${FIX_ROOT}/${BYTE_ORDER}/nam_glb_berror.f77.gcv
  OBERROR=${FIX_ROOT}/prepobs_errtable.global
if [ ${bk_core} = NMM ] ; then
     ANAVINFO=${FIX_ROOT}/anavinfo_ndas_netcdf_glbe
  fi
  if [ ${bk_core} = ARW ] ; then
    ANAVINFO=${FIX_ROOT}/anavinfo_arw_netcdf_glbe
  fi
  if [ ${bk_core} = NMMB ] ; then
    ANAVINFO=${FIX_ROOT}/anavinfo_nems_nmmb_glb
  fi
else
  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
fi

SATANGL=${FIX_ROOT}/global_satangbias.txt
SATINFO=${FIX_ROOT}/global_satinfo.txt
CONVINFO=${FIX_ROOT}/global_convinfo.txt
OZINFO=${FIX_ROOT}/global_ozinfo.txt
PCPINFO=${FIX_ROOT}/global_pcpinfo.txt

#  copy Fixed fields to working directory
 cp $ANAVINFO anavinfo
 cp $BERROR   berror_stats
 cp $SATANGL  satbias_angle
 cp $SATINFO  satinfo
 cp $CONVINFO convinfo
 cp $OZINFO   ozinfo
 cp $PCPINFO  pcpinfo
 cp $OBERROR  errtable

# CRTM Spectral and Transmittance coefficients
CRTM_ROOT_ORDER=${CRTM_ROOT}/${BYTE_ORDER}
emiscoef_IRwater=${CRTM_ROOT_ORDER}/Nalli.IRwater.EmisCoeff.bin
emiscoef_IRice=${CRTM_ROOT_ORDER}/NPOESS.IRice.EmisCoeff.bin
emiscoef_IRland=${CRTM_ROOT_ORDER}/NPOESS.IRland.EmisCoeff.bin
emiscoef_IRsnow=${CRTM_ROOT_ORDER}/NPOESS.IRsnow.EmisCoeff.bin
emiscoef_VISice=${CRTM_ROOT_ORDER}/NPOESS.VISice.EmisCoeff.bin
emiscoef_VISland=${CRTM_ROOT_ORDER}/NPOESS.VISland.EmisCoeff.bin
emiscoef_VISsnow=${CRTM_ROOT_ORDER}/NPOESS.VISsnow.EmisCoeff.bin
emiscoef_VISwater=${CRTM_ROOT_ORDER}/NPOESS.VISwater.EmisCoeff.bin
emiscoef_MWwater=${CRTM_ROOT_ORDER}/FASTEM5.MWwater.EmisCoeff.bin
aercoef=${CRTM_ROOT_ORDER}/AerosolCoeff.bin
```

```
cldcoef=${CRTM_ROOT_ORDER}/CloudCoeff.bin

ln -s $emiscoef_IRwater ./Nalli.IRwater.EmisCoeff.bin
ln -s $emiscoef_IRice ./NPOESS.IRice.EmisCoeff.bin
ln -s $emiscoef_IRsnow ./NPOESS.IRsnow.EmisCoeff.bin
ln -s $emiscoef_IRland ./NPOESS.IRland.EmisCoeff.bin
ln -s $emiscoef_VISice ./NPOESS.VISice.EmisCoeff.bin
ln -s $emiscoef_VISland ./NPOESS.VISland.EmisCoeff.bin
ln -s $emiscoef_VISsnow ./NPOESS.VISsnow.EmisCoeff.bin
ln -s $emiscoef_VISwater ./NPOESS.VISwater.EmisCoeff.bin
ln -s $emiscoef_MWwater ./FASTEM5.MWwater.EmisCoeff.bin
ln -s $aercoef  ./AerosolCoeff.bin
ln -s $cldcoef  ./CloudCoeff.bin
# Copy CRTM coefficient files based on entries in satinfo file
for file in `awk '{if($1!~"!"){print $1}}' ./satinfo | sort | uniq` ;do
   ln -s ${CRTM_ROOT_ORDER}/${file}.SpcCoeff.bin ./
   ln -s ${CRTM_ROOT_ORDER}/${file}.TauCoeff.bin ./
done

# Only need this file for single obs test
 bufrtable=${FIX_ROOT}/prepobs_prep.bufrtable
 cp $bufrtable ./prepobs_prep.bufrtable

# for satellite bias correction
cp ${FIX_ROOT}/sample.satbias ./satbias_in
```

Please note that in the above sample script, two fixed files related to radiance bias correction are copied from *fix/* to the work directory:

```
cp $SATANGL  satbias_angle
cp ${FIX_ROOT}/sample.satbias ./satbias_in
```

There are two options on how to perform the radiance bias correction. The first method is to do the angle dependent bias correction offline and do the mass bias correction inside the GSI analysis, therefore requiring the above two input files: `satbias_angle` corresponding to angle dependent bias correction file and `satbias_in` being the input file for mass bias correction. The second method is to combine the angle dependent and mass bias correction together and to do it within the GSI analysis, requiring one combined input file - `satbias_in`. Note that the input bias correction coefficients file `satbias_in` are different for the two options, therefore it is important to use the appropriate input file for each method. As shown in the previous section, the sample input files for the first method are provided with this release - `global_satangbias.txt` and `sample.satbias`. For using the second option – combined angle dependent and mass bias correction, a sample file `gdas1.t00z.abias.new` is also provided. Users, as a starting point, might also download a GDAS satbias coefficient file from the NOMADS ftp site as the input file (starting spring 2015, the GDAS satbias files have adopted the new format):

```
ftp://nomads.ncdc.noaa.gov/GDAS/YYYYMM/YYYYMMDD/gdas1.tHHz.abias
```

In order to use the combined angle dependent and mass bias correction, users also need to set `adp_anglebc=.true.` in the `&SETUP` section of the GSI namelist (*comgsi_namelist.sh*). For more details about the namelist, please see Appendix C in this document.

Set up some constants used in the GSI namelist. Please note that `bkcv_option` is set for background error tuning. They should be set based on specific applications. Here we provide three sample sets of the constants for different background error covariance options, one set is used in the NAM operations, one for the GFS operations and one for the NMMB operations. In this release, the capability of NMMB application is included and therefore the namelist settings for NMMB are provided in addition to NMM and ARW applications.

```
     ################################################################
     # Set some parameters for use by the GSI executable and to build the
     namelist
     echo " Build the namelist "
     # default is NAM
     #   as_op='1.0,1.0,0.5 ,0.7,0.7,0.5,1.0,1.0,'
     vs_op='1.0,'
     hzscl_op='0.373,0.746,1.50,'
     if [ ${bkcv_option} = GLOBAL ] ; then
     #   as_op='0.6,0.6,0.75,0.75,0.75,0.75,1.0,1.0'
        vs_op='0.7,'
        hzscl_op='1.7,0.8,0.5,'
     fi
     if [ ${bk_core} = NMMB ] ; then
        vs_op='0.6,'
     fi

     # default is NMM
        bk_core_arw='.false.'
        bk_core_nmm='.true.'
        bk_core_nmmb='.false.'
        bk_if_netcdf='.true.'
     if [ ${bk_core} = ARW ] ; then
        bk_core_arw='.true.'
        bk_core_nmm='.false.'
        bk_core_nmmb='.false.'
        bk_if_netcdf='.true.'
     fi
     if [ ${bk_core} = NMMB ] ; then
        bk_core_arw='.false.'
        bk_core_nmm='.false.'
        bk_core_nmmb='.true.'
        bk_if_netcdf='.false.'
     fi


The following section specifies the number of outer loops and whether to save GSI
read observations based on the setting of "if_observer".
     if [ ${if_observer} = Yes ] ; then
       nummiter=0
       if_read_obs_save='.true.'
       if_read_obs_skip='.false.'
     else
       nummiter=2
       if_read_obs_save='.false.'
       if_read_obs_skip='.false.'
     fi
```

The following section of the script is used to generate the GSI namelist called ***gsiparm.anl*** in the working directory. A detailed explanation of each variable can be found in Section 3.4 and Appendix C.

```
# Build the GSI namelist on-the-fly
. $GSI_NAMELIST
cat << EOF > gsiparm.anl

 $comgsi_namelist

EOF
```

Note: `EOF` indicates the end of GSI namelist.

The following block runs GSI and checks if GSI has successfully completed.

```
# ####################################################
#  run  GSI
####################################################
echo ' Run GSI with' ${bk_core} 'background'

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

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

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

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

The following block saves the analysis results with an understandable name and adds the analysis time to some output file names. Among them, `stdout` contains runtime output of GSI and `wrf_inout` is the analysis results.

```
######################################################################
#
#   GSI updating satbias_in
#
# GSI updating satbias_in (only for cycling assimilation)

# Copy the output to more understandable names
ln -s stdout      stdout.anl.${ANAL_TIME}
ln -s wrf_inout   wrfanl.${ANAL_TIME}
ln -s fort.201    fit_p1.${ANAL_TIME}
ln -s fort.202    fit_w1.${ANAL_TIME}
ln -s fort.203    fit_t1.${ANAL_TIME}
ln -s fort.204    fit_q1.${ANAL_TIME}
ln -s fort.207    fit_rad1.${ANAL_TIME}
```

The following block collects the diagnostic files. The diagnostic files are merged and categorized based on outer loop and data type. Setting `write_diag` to true in the namelist, directs GSI to write out diagnostic information for each observation station. This information is very useful to check analysis details. Please check Appendix A.2 for the tool to read and analyze these diagnostic files.

```
# Loop over first and last outer loops to generate innovation
# diagnostic files for indicated observation types (groups)
#
# NOTE:  Since we set miter=2 in GSI namelist SETUP, outer
#        loop 03 will contain innovations with respect to
#        the analysis.  Creation of o-a innovation files
#        is triggered by write_diag(3)=.true.  The setting
#        write_diag(1)=.true. turns on creation of o-g
#        innovation files.
#

loops="01 03"
for loop in $loops; do

case $loop in
  01) string=ges;;
  03) string=anl;;
   *) string=$loop;;
esac

#  Collect diagnostic files for obs types (groups) below
#listall="conv amsua_metop-a mhs_metop-a hirs4_metop-a hirs2_n14 msu_n14 \
#          sndr_g08 sndr_g10 sndr_g12 sndr_g08_prep sndr_g10_prep sndr_g12_prep \
#          sndrd1_g08 sndrd2_g08 sndrd3_g08 sndrd4_g08 sndrd1_g10 sndrd2_g10 \
#          sndrd3_g10 sndrd4_g10 sndrd1_g12 sndrd2_g12 sndrd3_g12 sndrd4_g12 \
#          hirs3_n15 hirs3_n16 hirs3_n17 amsua_n15 amsua_n16 amsua_n17 \
#          amsub_n15 amsub_n16 amsub_n17 hsb_aqua airs_aqua amsua_aqua \
#          goes_img_g08 goes_img_g10 goes_img_g11 goes_img_g12 \
#          pcp_ssmi_dmsp pcp_tmi_trmm sbuv2_n16 sbuv2_n17 sbuv2_n18 \
#          omi_aura ssmi_f13 ssmi_f14 ssmi_f15 hirs4_n18 amsua_n18 mhs_n18 \
#          amsre_low_aqua amsre_mid_aqua amsre_hig_aqua ssmis_las_f16 \
#          ssmis_uas_f16 ssmis_img_f16 ssmis_env_f16 mhs_metop_b \
#          hirs4_metop_b hirs4_n19 amusa_n19 mhs_n19"
listall=`ls pe* | cut -f2 -d"." | awk '{print substr($0, 0, length($0)-3)}'
  | sort | uniq `

   for type in $listall; do
   count=`ls pe*${type}_${loop}* | wc -l`
      if [[ $count -gt 0 ]]; then
         cat pe*${type}_${loop}* > diag_${type}_${string}.${ANAL_TIME}
      fi
done
done
```

The following scripts clean the temporal intermediate files

```
#  Clean working directory to save only important files
ls -l * > list_run_directory
if [ ${if_clean} = clean ]; then
  echo ' Clean working directory after GSI run'
  rm -f *Coeff.bin     # all CRTM coefficient files
  rm -f pe0*           # diag files on each processor
  rm -f obs_input.*    # observation middle files
  rm -f siganl sigf03  # background middle files
  rm -f fsize_*        # delete temperal file for bufr size
fi
```

The following block of the script runs only for `if_observer=Yes`, which runs GSI as observation operators for EnKF and without doing minimization. The script first renames the previous diagnostics files and GSI analysis file by appending .ensmean to the filenames to avoid these files being overwritten by the new GSI run.

```
################################################
# start to calculate diag files for each member
################################################
#
 if [ ${if_observer} = Yes ] ; then
   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
```

Next, the script generates the namelist for each ensemble member.

```
# 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 rest of the script loops through the ensemble members to get the background ready, run GSI and check the run status:

```
# 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`

# 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  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}
```

The following lines generate the diagnostics files for each 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
```

The following section is to move on to the next ensemble member and run GSI.

```
# next member
      (( ensmem += 1 ))

  done

fi
```

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

```
  exit 0
```

3.2.4 Run GSI with gfortran platform

When GSI is compiled with gfortran, the BUFRLIB cannot automatically handle the byte order issue. We can run GSI with the same *run_gsi_regional.ksh* but need to set

```
BYTE_ORDER=Little_Endian
```

3.3 GSI Analysis Result Files in Run Directory

Once the GSI run script is set up, it is ready to be submitted just as other batch jobs. When completed, GSI will create a number of files in the run directory. Below is an example of

the files generated in the run directory from one of the GSI test case runs. This case was run to perform a regional GSI analysis with a WRF ARW NetCDF background using conventional (prepbufr), radiance (AMSU-A, HIRS4, and MHS), and GPSRO data. The analysis time is 00Z 17 June 2014. Four processors were used. To make the run directory more readable, we turned on the clean option in the run script, which deleted all temporary intermediate files.

```
amsuabufr                            fort.201  fort.227
anavinfo                             fort.202  fort.228
berror_stats                         fort.203  fort.229
convinfo                             fort.204  fort.230
diag_amsua_metop-a_anl.2014061700    fort.205  gpsrobufr
diag_amsua_metop-a_ges.2014061700    fort.206  gsi.exe
diag_amsua_n15_anl.2014061700        fort.207  gsiparm.anl
diag_amsua_n15_ges.2014061700        fort.208  hirs4bufr
diag_amsua_n18_anl.2014061700        fort.209  l2rwbufr
diag_amsua_n18_ges.2014061700        fort.210  list_run_directory
diag_conv_anl.2014061700             fort.211  mhsbufr
diag_conv_ges.2014061700             fort.212  ozinfo
diag_hirs4_metop-a_anl.2014061700    fort.213  pcpbias_out
diag_hirs4_metop-a_ges.2014061700    fort.214  pcpinfo
diag_mhs_metop-a_anl.2014061700      fort.215  prepbufr
diag_mhs_metop-a_ges.2014061700      fort.217  prepobs_prep.bufrtable
diag_mhs_n18_anl.2014061700          fort.218  satbias_angle
diag_mhs_n18_ges.2014061700          fort.219  satbias_in
errtable                             fort.220  satbias_out
fit_p1.2014061700                    fort.221  satinfo
fit_q1.2014061700                    fort.223  stdout
fit_rad1.2014061700                  fort.224  stdout.anl.2014061700
fit_t1.2014061700                    fort.225  wrfanl.2014061700
fit_w1.2014061700                    fort.226  wrf_inout
```

It is important to know which files hold the GSI analysis results, standard output, and diagnostic information. We will introduce these files and their contents in detail in the following chapter. The following is a brief list of what these files contain:

`stdout.anl.2014061700/stdout`: standard text output file, which is a link to `stdout` with the analysis time appended. This is the most commonly used file to check the GSI analysis processes as well as basic and important information about the analyses. We will explain the contents of `stdout` in Section 4.1 and users are encouraged to read this file in detail to become familiar with the order of GSI analysis processing.

`wrfanl.2014061700/wrf_inout`: analysis results if GSI completes successfully – it exists only if using WRF for background. This is a link to `wrf_inout` with the analysis time appended. The format is the same as the background file.

`diag_conv_anl.(time)`: binary diagnostic files for conventional and GPS RO observations at the final analysis step (analysis departure for each observation).

`diag_conv_ges.(time)`: binary diagnostic files for conventional and GPS RO observations before initial analysis step (background departure for each observation)

`diag_(instrument_satellite)_anl`: diagnostic files for satellite radiance observations at final analysis step.

`diag_(instrument_satellite)_ges`: diagnostic files for satellite radiance observations before initial analysis step.

`gsiparm.anl`: GSI namelist, generated by the run script.

`fit_(variable).(time)`: links to `fort.2??` with meaningful names (variable name plus analysis time). They are statistic results of observation departures from background and analysis results according to observation variables. Please see Section 4.5 for more details.

`fort.220`: output from the inner loop minimization (in *pcgsoi.f90*). Please see Section 4.6 for details.

`anavinfo`: info file to set up control variables, state variables, and background variables. Please see Advanced GSI User's Guide for details.

`*info (convinfo, satinfo,…)`: info files that control data usage. Please see Section 4.3 for details.

`berror_stats` and `errtable`: background error file (binary) and observation error file (text).

`*bufr`: observation BUFR files linked to the run directory Please see Section 3.1 for details.

`satbias_in`: the input coefficients of mass bias correction for satellite radiance observations.

`satbias_out`: the output coefficients of mass bias correction for satellite radiance observations after the GSI run.

`satbias_angle`: the input coefficients of scan angle bias correction for satellite radiance observations.

`list_run_directory` : the complete list of files in the run directory before cleaning the run directory. This is generated by the GSI run script.

The diag files, such as `diag_(instrument_satellite)_anl.(time)` and `diag_conv_anl.(time)`, contain important information about the data used in the GSI, including observation departure from analysis results for each observation (O-A). Similarly, `diag_conv_ges` and `diag_(instrument_satellite)_ ges.(time)` include observation innovation for each observation (O-B). These files can be very helpful in understanding the detailed impact of data on the analysis. A tool is provided to process these files, which is introduced in Appendix A.2.

There are many intermediate files in this directory during the running stage or if the GSI run crashes; the complete list of files before cleaning is saved in a file *list_run_directory*. Some knowledge about the content of these files is very helpful for debugging if the GSI run crashes. Please check the following list for the meaning of these files: (Note: you may not see all the files in the list because different observational data are used. Also, the fixed files prepared for a GSI run, such as CRTM coefficient files, are not included.)

File name	Content
`sigf03`	This is a temporal file holding binary format background files (typically sigf03, sigf06 and sigf09 if FGAT used). When you see this file, at the minimum, a background file was successfully read in.
`siganl`	Analysis results in binary format. When this file exists, the analysis part has finished.
`pe????.(conv or instrument_satellite)_`	Diagnostic files for conventional and satellite radiance

(outer loop)	observations at each outer loop and each sub-domains (????=subdomain id)
obs_input.????	Observation scratch files (each file contains observations for one observation type within whole analysis domain and time window. ????=observation type id in namelist)
pcpbias_out	Output precipitation bias correction file

3.4 Introduction to Frequently Used GSI Namelist Options

The complete namelist options and their explanations are listed in Advanced GSI User's Guide Appendix A. For most GSI analysis applications, only a few namelist variables need to be changed. Here we introduce frequently used variables for regional analyses:

1. Set up the number of outer loop and inner loop

To change the number of outer loops and the number of inner iterations in each outer loop, the following three variables in the namelist need to be modified:

`miter`: number of outer loops of analysis.
`niter(1)`: maximum iteration number of inner loop iterations for the 1st outer loop. The inner loop will stop when it reaches this maximum number, reaches the convergence condition, or when it fails to converge.
`niter(2)`: maximum iteration number of inner loop iterations for the 2nd outer loop.
If `miter` is larger than 2, repeat `niter` with larger index.

2. Set up the analysis variable for moisture

There are two moisture analysis variable options. It is decided by the namelist variable:

`qoption = 1 or 2`:
If qoption=1, the moisture analysis variable is pseudo-relative humidity. The saturation specific humidity, *qsatg*, is computed from the guess and held constant during the inner loop. Thus, the RH control variable can only change via changes in specific humidity, q.
If qoption=2, the moisture analysis variable is normalized RH. This formulation allows RH to change in the inner loop via changes to surface pressure (pressure), temperature, or specific humidity.

3. Set up the background file

The following four variables define which background field will be used in the GSI analyses:

`regional`: if true, perform a regional GSI run using either ARW or NMM inputs as the background. If false, perform a global GSI analysis. If either `wrf_nmm_regional` or `wrf_mass_regional` are true, it will be set to true.

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

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

`nems_nmmb_regional`: if true, background comes from NMMB. When using other background fields, set it to false.

`netcdf`: if true, WRF files are in NetCDF format, otherwise WRF files are in binary format. This option only works for performing a regional GSI analysis.

4. Set up the output of diagnostic files

The following variables tell the GSI to write out diagnostic results in certain loops:

`write_diag(1)`: if true, write out diagnostic data in the beginning of the analysis, so that we can have information on Observation – Background (O-B) .

`write_diag(2)` : if true, write out diagnostic data at the end of the 1st (before the 2nd outer loop starts) .

`write_diag(3)`: if true, write out diagnostic data at the end of the 2nd outer loop (after the analysis finishes if the outer loop number is 2), so that we can have information on Observation – Analysis (O-A) .

Please check appendix A.2 for the tools to read the diagnostic files.

5. Set up the GSI recognized observation files

The following sets up the GSI recognized observation files for GSI observation ingest:

```
OBS_INPUT::
!  dfile          dtype       dplat     dsis                dval    dthin dsfcalc
   prepbufr       ps          null      ps                  1.0     0     0
   prepbufr       t           null      t                   1.0     0     0
   prepbufr       q           null      q                   1.0     0     0
   prepbufr       pw          null      pw                  1.0     0     0
   satwndbufr     uv          null      uv                  1.0     0     0
   prepbufr       uv          null      uv                  1.0     0     0
   prepbufr       spd         null      spd                 1.0     0     0
   prepbufr       dw          null      dw                  1.0     0     0
   radarbufr      rw          null      rw                  1.0     0     0
   prepbufr       sst         null      sst                 1.0     0     0
   gpsrobufr      gps_ref     null      gps                 1.0     0     0
```

```
   ssmirrbufr     pcp_ssmi    dmsp      pcp_ssmi            1.0    -1     0
```

`dfile`: GSI recognized observation file name. The observation file contains observations used for a GSI analysis. This file can include several observation variables from different observation types. The file name in this parameter will be read in by GSI. This name can be changed as long as the name in the link from the BUFR/PrepBUFR file in the run scripts also changes correspondingly.

`dtype`: analysis variable name that GSI can read in and handle. Please note this name should be consistent with that used in the GSI code.

`dplat`: sets up the observation platform for a certain observation, which will be read in from the file `dfile`.

`dsis`: sets up data name (including both data type and platform name) used inside GSI.

Please see Section 4.3 for examples and explanations of these variables.

6. Set up observation time window

In the namelist section `OBS_INPUT`, use `time_window_max` to set maximum half time window (hours) for all data types. In the *convinfo* file, you can use the column `twindow` to set the half time window for a certain data type (hours). For conventional observations, only observations within the smaller window of these two will be kept for further processing. For others, observations within `time_window_max` will be kept for further processing.

7. Set up data thinning

1) Radiance data thinning

Radiance data thinning is controlled through two GSI namelist variables in the section `&OBS_INPUT`. Below is an example of the section:

```
&OBS_INPUT
      dmesh(1)=120.0,dmesh(2)=60.0,dmesh(3)=30,time_window_max=1.5,ext_sonde=.true.,
 /
OBS_INPUT::
!  dfile          dtype       dplat     dsis                dval    dthin dsfcalc
   prepbufr       ps          null      ps                  1.0     0     0

   gpsrobufr      gps_ref     null      gps                 1.0     0     0
   ssmirrbufr     pcp_ssmi    dmsp      pcp_ssmi            1.0    -1     0
   tmirrbufr      pcp_tmi     trmm      pcp_tmi             1.0    -1     0

   hirs3bufr      hirs3       n17       hirs3_n17           6.0     1     0
   hirs4bufr      hirs4       metop-a   hirs4_metop-a       6.0     2     0
```

The two namelist variables that control the radiance data thinning are real array `dmesh` in the 1st line and the `dthin` values in the 6th column. The `dmesh` gives a set of the mesh sizes in unit *km* for radiance thinning grids, while the `dthin` defines if the data type it represents needs to be thinned and which thinning grid (mesh size) to use. If the value of `dthin` is:

- an integer less than or equal to 0, no thinning is needed
- an integer larger than 0, this kind of radiance data will be thinned in a thinning grid with the mesh size defined as `dmesh` (`dthin`).

The following gives several thinning examples defined by the above sample `&OBS_INPUT` section:

- Data type ps from prepbufr: no thinning because `dthin=0`
- Data type gps_ref from gpsrobufr: no thinning because `dthin=0`
- Data type pcp_ssmi from dmsp: no thinning because `dthin(01)=-1`
- Data type hirs3 from NOAA-17: thinning in a 120 km grid because `dthin=1` and `dmesh(1)=120`
- Data type hirs4 from metop-a: thinning in a 60 km grid because `dthin=2` and `dmesh(2)=60`

2) Conventional data thinning

The conventional data can also be thinned. However, the setup of thinning is not in the namelist. To give users a complete picture of data thinning, conventional data thinning is briefly introduced here. There are three columns, `ithin, rmesh, pmesh`, in the *convinfo* file (more details on this file are in Section 4.3) to configure conventional data thinning:

- `ithin`: 0 = no thinning;
  1 = thinning with grid mesh decided by `rmesh` and `pmesh`
- `rmesh`: horizontal thinning grid size in *km*
- `pmesh`: vertical thinning grid size in *mb*; if 0, then use background vertical grid.

8. Set up background error factor

In the namelist section `BKGERR`, `vs` is used to set up the scale factor for vertical correlation length and `hzscl` is defined to set up scale factors for horizontal smoothing. The scale factors for the variance of each analysis variables are set in the *anavinfo* file. The typical values used in operations for regional and global background error covariance are given and picked based on the choice of background error covariance in the run scripts and sample *anavinfo* files.

9. Single observation test

To do a single observation test, the following namelist option has to be set to true:

```
oneobtest=.true.
```

Then go to the namelist section `SINGLEOB_TEST` to set up the single observation location and variable to be tested, please see Section 4.2 for an example and details on the single observation test.

Chapter 4: GSI Diagnostics and Tuning

The guidance in this chapter will help users to understand how and where to check the output from GSI to determine whether a run was successful. Properly checking the GSI output will also provide useful information to diagnose potential errors in the system. The chapter starts with an introduction to the content and structure of the GSI standard output (**stdout**). It continues with the use of a single observation to check the features of the GSI analysis. Then, observation usage control, analysis domain partition, fit files, and the optimization process will all be presented from information within the GSI output files (including stdout).

This chapter follows the online case example for 2014061700. This case uses a WRF-ARW NetCDF file as the background and analyzes several observations typical for operations, including most conventional observation data, several radiance data (AMSU-A, HIRS4, and MHS), and GPSRO data. The case was run on a Linux cluster supercomputer, using 4 processors. Users can follow this test to reproduce the following results by visiting:

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

4.1 Understanding Standard Output (*stdout*)

In Section 3.3, we listed the files present in the GSI run directory following a successful GSI analysis and briefly introduced the contents of several important files. Of these, **stdout** is the most useful because critical information about the GSI analysis can be obtained from the file. From **stdout**, users can check if the GSI has successfully completed, if optimal iterations look correct, and if the background and analysis fields are reasonable. Understanding the content of this file can also be very helpful for users to find where and why the GSI failed if it crashes.

The structure of **stdout** follows the typical steps in a meteorological data analysis system:

1. Read in all data and prepare analysis:
    a. Read in configuration (namelist)
    b. Read in background
    c. Read in observations
    d. Partition domain and data for parallel analysis
    e. Read in constant fields (fixed files)
2. Optimal iteration (analysis)
3. Save analysis result

In this section, the detailed structure and content of **stdout** are explained using the v3.3 online example case: 2014061700. To keep the output concise and make it more readable, most repeated content was deleted (shown by the blue dotted line). For the same reason, the accuracy of some numbers has been reduced to avoid line breaks in **stdout**.

The following indicates the start of the GSI analysis. It shows the beginning time of this run:

```
* . * . * . * . * . * . * . * . * .  * . * . * . * . * . * . * . * . * . * .
     PROGRAM GSI_ANL HAS BEGUN. COMPILED 1999232.55     ORG: NP23
     STARTING DATE-TIME  JUL 16,2015  10:51:06.534  197  THU   2457220
```

This part shows the content of anavinfo, a list of state and control variables.

```
gsi_metguess_mod*init_:  2D-MET STATE VARIABLES:
 ps
 z
 gsi_metguess_mod*init_:  3D-MET STATE VARIABLES:
 u
 v
 div
 vor
 tv
 q
 oz
 cw
 gsi_metguess_mod*init_: ALL MET STATE VARIABLES:
 u
 v
 div
 vor
 tv
 q
 oz
 cw
 ps
 z
 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
 control_vectors*init_anacv: 2D-CONTROL VARIABLES ARE
 ps
 sst
 control_vectors*init_anacv: 3D-CONTROL VARIABLES ARE
 sf
 vp
 t
 q
```

```
 oz
 cw
 control_vectors*init_anacv: MOTLEY CONTROL VARIABLES
 stl
 sti
 control_vectors*init_anacv: ALL CONTROL VARIABLES
 sf
 vp
 ps
 t
 q
 oz
 sst
 cw
 stl
 sti
 INIT_IO:  reserve units lendian_in=           15  and lendian_out=
           66  for little endian i/o
 INIT_IO:  set IO server task to mype_io=            3
  at 0 in gsimod, use_gfs_stratosphere,nems_nmmb_regional =   F  F
```

Next is the content of all namelist variables used in this analysis. The 1st part shows the 4DVAR setups. Please note that while this version of the GSI includes some 4DVAR interface, it is untested in this release. The general set up for the GSI analysis (3DVAR) is located in the `&SETUP` section of the GSI namelist. Please check Appendix B for definitions and default values of each namelist variable.

```
GSI_4DVAR:  nobs_bins =             1
 SETUP_4DVAR: l4dvar=  F
 SETUP_4DVAR: l4densvar=  F
 SETUP_4DVAR: winlen=    3.000000000000000
 SETUP_4DVAR: winoff=    3.000000000000000
 SETUP_4DVAR: hr_obsbin=    3.000000000000000
 SETUP_4DVAR: nobs_bins=             1
 SETUP_4DVAR: ntlevs_ens=             1
 SETUP_4DVAR: nsubwin,nhr_subwin=           1            3
 SETUP_4DVAR: lsqrtb=  F
 SETUP_4DVAR: lbicg=  F
```

• • • •

```
&SETUP
 GENCODE =    78.00000000000000     ,
 FACTQMIN =    0.000000000000000     ,
 FACTQMAX =    0.000000000000000     ,
 CLIP_SUPERSATURATION =  F,
 FACTV =    1.000000000000000     ,
 FACTL =    1.000000000000000     ,
 FACTP =    1.000000000000000     ,
 FACTG =    1.000000000000000     ,
 FACTW10M =    1.000000000000000     ,
 FACTHOWV =    1.000000000000000     ,
 R_OPTION =  F,
 DELTIM =    1200.000000000000     ,
 DTPHYS =    3600.000000000000     ,
 BIASCOR =   -1.000000000000000     ,
 BCOPTION =           1,
 DIURNALBC =    0.000000000000000     ,
 NITER =            0,
                   10,
                   10,
```

• • • •

```
 /
 &GRIDOPTS
```

```
 &BKGERR
 &ANBKGERR
 &JCOPTS
 &STRONGOPTS
 &OBSQC
 &SUPEROB_RADAR
 &LAG_DATA
 &HYBRID_ENSEMBLE
 &RAPIDREFRESH_CLDSURF
 &CHEM
```

This version of GSI attempts to read multi-time-level backgrounds for FGAT, however we only have provided one in this test case. Therefore, there is error information at the beginning of the reading background portion:

```
CONVERT_NETCDF_MASS:  problem with flnm1 = wrf_inou1, Status =        -1021
```

We can ignore these errors for missing files *wrf_inou1, wrf_inou2*, … …, *wrf_inou9* because we only ran 3DVAR with one background.

Next, the background fields for the analysis are read in and the maximum, minimum and median values of the fields at each vertical level are displayed. Here, only part of the variables `znu` and `T` are shown, and all other variables read by the GSI are listed only as the variable name in the NetCDF file (`rmse_var` = …). The maximum and minimum values are useful for a quick verification that the background fields have been read successfully. From this section, we also know the time (`iy,m,d,h,m,s`) and dimension (`nlon,lat,sig_regional`) of the background field.

```
  dh1  =             3
  iy,m,d,h,m,s=         2014            6           17            0
             0            0
  dh1  =             3
 rmse_var = SMOIS
 ndim1 =            3
 ordering = XYZ
 staggering =  N/A
 start_index =             1            1            1            0
 end_index =           332          215            4            0
 WrfType =          104
 ierr  =            0
  rmse_var = T ndim1 =             3  dh1 =             3
  WrfType =          104 ierr  =            0
  ordering = XYZ staggering =  N/A
  start_index =             1            1            1            0
  end_index =           332          215           50            0
  nlon,lat,sig_regional=          332          215           50
  rmse_var = P_TOP ndim1=            0
```

```
  rmse_var = ZNU ndim1=            1
  WrfType =          104  WRF_REAL=          104 ierr  =            0
  ordering = Z staggering =  N/A
```

```
 start_index =             1           1           1           0
 end_index =              50         215          50           0
 k,znu(k)=             1   0.9990000
 k,znu(k)=             2   0.9960001
 k,znu(k)=             3   0.9905000
 k,znu(k)=            49   7.1999999E-03
 k,znu(k)=            50   2.3500000E-03
 rmse_var = ZNW ndim1=            1
 rmse_var = RDX ndim1=           0
 rmse_var = RDY ndim1=           0
 rmse_var = MAPFAC_M ndim1=           2
 rmse_var = XLAT ndim1=           2
 rmse_var = XLONG ndim1=           2
 rmse_var = MUB ndim1=           2
 rmse_var = MU ndim1=           2
 rmse_var = PHB ndim1=           3
 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 =           332         215          50           0
 k,max,min,mid T=           1    321.6354    280.1795    308.9041
 k,max,min,mid T=           2    321.6452    281.1811    308.9584
 k,max,min,mid T=           3    321.4893    282.7797    309.0582
 k,max,min,mid T=           4    321.4048    283.8192    309.2308
 k,max,min,mid T=           5    321.7208    283.9999    309.3156
 k,max,min,mid T=          48    624.4512    576.9553    596.8545
 k,max,min,mid T=          49    649.7656    613.7888    630.5093
 k,max,min,mid T=          50    680.9388    653.7093    668.4511
 rmse_var = QVAPOR ndim1=           3
 rmse_var = U ndim1=           3
 rmse_var = V ndim1=           3
 rmse_var = XLAND ndim1=           2
 rmse_var = SEAICE ndim1=           2
 rmse_var = SST ndim1=           2
 rmse_var = IVGTYP ndim1=           2
 rmse_var = ISLTYP ndim1=           2
 rmse_var = VEGFRA ndim1=           2
 rmse_var = SNOW ndim1=           2
 rmse_var = U10 ndim1=           2
 rmse_var = V10 ndim1=           2
```

```
 rmse_var = SMOIS ndim1=           3
 rmse_var = TSLB ndim1=           3
 rmse_var = TSK ndim1=           2
 rmse_var = Q2 ndim1=           2
 rmse_var = QCLOUD ndim1=           3
 rmse_var = QRAIN ndim1=           3
 rmse_var = QSNOW ndim1=           3
 rmse_var = QICE ndim1=           3
 rmse_var = QGRAUP ndim1=           3
 rmse_var = QNRAIN ndim1=           3
 rmse_var = RAD_TTEN_DFI ndim1=           3
```

For some variables, the following NETCDF error information might show up when the variables are not in the background fields. These errors don't affect the GSI run so you can ignore them.

```
 rmse_var = QSNOW ndim1=            3
 WrfType =          104  WRF_REAL=          104 ierr  =         -1021
 ordering = XYZ staggering =  N/A
 start_index =            1           1           1           0
 end_index =          332         215          50           0
 NetCDF error: NetCDF: Variable not found
 NetCDF error: NetCDF: Variable not found
 NetCDF error in wrf_io.F90, line         2861  Varname QSNOW
 NetCDF error in wrf_io.F90, line         2861  Varname QSNOW
```

Again, some error information on missing background files shows up. Ignore if you are not doing FGAT:

```
CONVERT_NETCDF_MASS:  problem with flnm1 = wrf_inou4, Status =        -1021
```

Following this is information on the byte order of the binary background files. Because we used a NetCDF file, there is no need to be concerned with byte order. When using a binary format background, byte-order can be a problem. Beginning with the release version v3.2, GSI can automatically check the background byte-order and read it in right order:

```
 in convert_regional_guess, for wrf arw binary input, byte_swap= F
```

Information on setting the grid related variables, and the beginning and ending indices for thread 1:

```
INIT_GRID_VARS:  number of threads            1
INIT_GRID_VARS:  for thread            1  jtstart,jtstop =            1
         168
```

Information on the initial pointer location for each variable in the Jacobian for the use of the satellite radiance data:

```
 Vars in Rad-Jacobian (dims)
 --------------------------
 sst                            0
 u                              1
 v                              2
 tv                             3
 q                             53
 oz                           103
```

Starting *gsisub* and displaying the analysis and background file time (they should be the same):

```
[000]gsisub(): : starting ...
 READ_wrf_mass_FILES:  analysis date,minutes          2014           6
          17           0           0     19175040
 READ_wrf_mass_FILES:  sigma guess file, nming2     0.000000000000000
         2014           6          17           0           0     19175040
 READ_wrf_mass_FILES:  sigma fcst files used in analysis  :              3
    3.000000000000000                 1
 READ_wrf_mass_FILES:  surface fcst files used in analysis:              3
    3.000000000000000                 1
 GESINFO:  Guess    date is             0           6          17
         2014    0.000000000000000
 GESINFO:  Analysis date is          2014           6          17
           0           0   2014061700    3.000000000000000
  using restart file date =          2014           6          17
           0
```

Read in radar station information and generate superobs for radar Level-II radial velocity. This case didn't have radar Level-II velocity data linked. There is warning information about opening the file but this will not impact the rest of the GSI analysis.

```
 RADAR_BUFR_READ_ALL: analysis time is          2014           6          17
           0
 RADAR_BUFR_READ_ALL:  NO RADARS KEPT IN radar_bufr_read_all,
 continue without level 2 data
```

Read in fix file information on scaninfo and pcpinfo.

```
***WARNING file scaninfo not found, use default
 CREATE_PCP_RANDOM:  iseed=   2014061700
 PCPINFO_READ:  no pcpbias file.  set predxp=0.0
```

Read in and show the content of the observation info files (see Section 4.3 for details). Here is part of the stdout shown **convinfo**:

```
READ_CONVINFO: tcp  112    0  1   3.00000     0   0   0   75.0000      5.00000      1.00000
75.0000      0.00000        0  0.00000      0.00000        0  0.00000      0.00000
READ_CONVINFO: ps   120    0  1   3.00000     0   0   0   4.00000      3.00000      1.00000
4.00000     0.300000E-03    0  0.00000      0.00000        0  0.00000      0.00000
```

● ● ● ●

```
READ_CONVINFO: t    120    0  1   3.00000     0   0   0   8.00000      5.60000      1.30000
8.00000     0.100000E-05    0  0.00000      0.00000        0  0.00000      0.00000
READ_CONVINFO: t    126    0 -1   3.00000     0   0   0   8.00000      5.60000      1.30000
8.00000     0.100000E-02    0  0.00000      0.00000        0  0.00000      0.00000
```

● ● ● ●

```
READ_CONVINFO: gps   440    0 -1   3.00000    0   0   0   10.0000      10.0000      1.00000
10.0000      0.00000        0  0.00000      0.00000        0  0.00000      0.00000
READ_CONVINFO: pm2_5 102    0 -1   1.00000    0   0   0   100.000      1.50000     0.750000
10.0000      0.00000        0  0.00000      0.00000        0  0.00000      0.00000
```

Starting *glbsoi* and information on reading in background fields from intermediate binary file *sigf03* and partitioning the whole 2D field into subdomains for parallel analysis:

```
 glbsoi: starting ...
 gsi_metguess_mod*create_: alloc() for met-guess done
 guess_grids*create_chemges_grids: trouble getting number of chem/gases
 READ_WRF_MASS_GUESS:  open lendian_in=           15  to file=sigf03
 READ_WRF_MASS_GUESS:  open lendian_in=           15  to file=sigf03
  at 0 in read_wrf_mass_guess
  at 0.1 in read_wrf_mass_guess
 at 1 in read_wrf_mass_guess, lm            =    50
 at 1 in read_wrf_mass_guess, num_mass_fields=   215
 at 1 in read_wrf_mass_guess, nfldsig       =     1
 at 1 in read_wrf_mass_guess, num_all_fields=   215
 at 1 in read_wrf_mass_guess, npe           =     4
 at 1 in read_wrf_mass_guess, num_loc_groups=    53
 at 1 in read_wrf_mass_guess, num_all_pad   =   216
 at 1 in read_wrf_mass_guess, num_loc_groups=    54
 READ_WRF_MASS_GUESS:  open lendian_in=           15  to file=sigf03
 READ_WRF_MASS_GUESS:  open lendian_in=           15  to file=sigf03
  in read_wrf_mass_guess, num_doubtful_sfct_all =             0
  in read_wrf_mass_guess, num_doubtful_sfct_all =             0
```

Show observer is successfully initialized and inquire about the control vectors.

```
 observer_init: successfully initialized
 control_vectors: length=      5599732
 control_vectors: currently allocated=           0
 control_vectors: maximum allocated=           0
 control_vectors: number of allocates=           0
 control_vectors: number of deallocates=           0
 control_vectors: Estimated max memory used=      0.0 Mb
```

Show the source of observation error used in the analysis (details see Section 4.7.1):

```
 CONVERR:  using observation errors from user provided table
```

The following information is related to observation ingest processes, which are distributed over all the processors with each processor reading in at least one observation type. To speed up reading process, some of the large datasets will use more than one (ntasks) processor to read.

Before reading in the data from BUFR files, GSI resets the file status depending on whether the observation time matches the analysis time and how *offtime_date* is set. This step also checks for consistency between the satellite radiance data types in the BUFR files and the usage setups in the *satinfo* files. The following shows *stdout* information from this step:

```
 read_obs_check: bufr file date is    2014061700 prepbufr q
 read_obs_check: bufr file date is    2014061700 prepbufr ps
 read_obs_check: bufr file uv                    not available satwndbufr
 read_obs_check: bufr file rw                    not available radarbufr
```

```
 read_obs_check: bufr file pcp_tmi   trmm       not available tmirrbufr
 read_obs_check: bufr file hirs3     n17        not available hirs3bufr
 read_obs_check: bufr file goes_img  g11        not available gimgrbufr
 read_obs_check: bufr file date is    2014061700 prepbufr t
 read_obs_check: bufr file date is    2014061700 amsuabufr amsua     n18
 read_obs_check: bufr file amsua     n18        not available amsuabufrears
```

● ● ● ●

```
 read_obs_check: bufr file sndrd3    g15        not available gsnd1bufr
 read_obs_check: bufr file omi       aura       not available omibufr
 read_obs_check: bufr file seviri    m09        not available seviribufr
 read_obs_check: bufr file atms      npp        not available atmsbufr
 read_obs_check: bufr file date is    2014061700 prepbufr sst
 read_obs_check: bufr file sbuv2     n16        not available sbuvbufr
 read_obs_check: bufr file date is    2014061700 hirs4bufr hirs4     metop-a
 read_obs_check: bufr file goes_img  g12        not available gimgrbufr
 read_obs_check: bufr file date is    2014061700 amsuabufr amsua     n19
```

● ● ● ●

```
 read_obs_check: bufr file gome      metop-b    not available gomebufr
 read_obs_check: bufr file uv                   not available oscatbufr
 data type mta_cld             not used in info file -- do not read file
 prepbufr
 data type gos_ctp             not used in info file -- do not read file
 prepbufr
 data type rad_ref             not used in info file -- do not read file
 refInGSI
 data type lghtn               not used in info file -- do not read file
 lghtInGSI
 data type larccld             not used in info file -- do not read file
 larcInGSI
```

The list of observation types that will be read in and processors used to read them:

```
  number of extra processors            1
 READ_OBS:  read   19 hirs4      hirs4_metop-a        using ntasks=   2   0   2
 READ_OBS:  read   33 mhs        mhs_n18              using ntasks=   2   2   2
 READ_OBS:  read   34 mhs        mhs_n19              using ntasks=   2   0   2
 READ_OBS:  read   35 mhs        mhs_metop-a          using ntasks=   2   2   2
 READ_OBS:  read   36 mhs        mhs_metop-b          using ntasks=   2   0   2
 READ_OBS:  read    1 ps         ps                   using ntasks=   1   2   1
 READ_OBS:  read    2 t          t                    using ntasks=   1   3   1
 READ_OBS:  read    3 q          q                    using ntasks=   1   0   1
 READ_OBS:  read    4 pw         pw                   using ntasks=   1   1   1
 READ_OBS:  read    6 uv         uv                   using ntasks=   1   2   1
 READ_OBS:  read   10 sst        sst                  using ntasks=   1   3   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_OBS:  read   22 hirs4      hirs4_metop-b        using ntasks=   1   2   1
 READ_OBS:  read   26 amsua      amsua_n15            using ntasks=   1   3   1
 READ_OBS:  read   27 amsua      amsua_n18            using ntasks=   1   0   1
 READ_OBS:  read   28 amsua      amsua_n19            using ntasks=   1   1   1
 READ_OBS:  read   29 amsua      amsua_metop-a        using ntasks=   1   2   1
 READ_OBS:  read   30 amsua      amsua_metop-b        using ntasks=   1   3   1
```

Display basic statistics for full horizontal surface fields (If radiance BUFR files are not linked, this section will not be in the ***stdout*** file):

```
 GETSFC:  enter with nlat_sfc,nlon_sfc=            0           0
  and nlat,nlon=          215          332
 GETSFC: set nlat_sfc,nlon_sfc=          215          332
================================================================================
Status   Var          Mean                Min                 Max
sfcges2 FC10   1.000000000000E+00   1.000000000000E+00   1.000000000000E+00
```

```
sfcges2 SNOW   2.066276863261E-02   0.000000000000E+00   1.026125106812E+02
sfcges2 VFRC   3.682271878310E-01   0.000000000000E+00   9.899999237061E-01
sfcges2 SRGH   5.000000000000E-02   5.000000000000E-02   5.000000000000E-02
sfcges2 STMP   2.954282288089E+02   2.715000000000E+02   3.243763732910E+02
sfcges2 SMST   5.052902985158E-01   5.994479730725E-02   1.000000000000E+00
sfcges2 SST    2.971105685225E+02   2.715000000000E+02   3.256103210449E+02
sfcges2 VTYP   1.122873353881E+01   1.000000000000E+00   2.100000000000E+01
sfcges2 ISLI   6.631969739423E-01   0.000000000000E+00   1.000000000000E+00
sfcges2 STYP   8.127178481367E+00   1.000000000000E+00   1.600000000000E+01
================================================================================
```

Loop over all data files to read in observations, also reads in rejection list for surface observations and show GPS observations outside the time window:

```
  READ_BUFRTOVS          : file=hirs4bufr       type=hirs4      sis=hirs4_metop-a
nread=     13832 ithin= 2 rmesh=  60.000000 isfcalc= 0 ndata=      7315 ntask=  2
 READ_BUFRTOVS          : file=mhsbufr         type=mhs        sis=mhs_n18
nread=     91065 ithin= 2 rmesh=  60.000000 isfcalc= 0 ndata=     11985 ntask=  2
 READ_BUFRTOVS          : file=mhsbufr         type=mhs        sis=mhs_n19
nread=         0 ithin= 2 rmesh=  60.000000 isfcalc= 0 ndata=         0 ntask=  2
 READ_BUFRTOVS          : file=mhsbufr         type=mhs        sis=mhs_metop-a
nread=     14465 ithin= 2 rmesh=  60.000000 isfcalc= 0 ndata=      2135 ntask=  2
 READ_BUFRTOVS          : file=mhsbufr         type=mhs        sis=mhs_metop-b
nread=      2465 ithin= 2 rmesh=  60.000000 isfcalc= 0 ndata=       560 ntask=  2
 READ_PREPBUFR: messages/reports =          3229 /       389062  ntread =
           1
 READ_PREPBUFR: time offset is     3.000000000000000       hours.
 new vad flag::  F
 READ_PREPBUFR: messages/reports =          3229 /       389062  ntread =
           1
 new vad flag::  F
 READ_PREPBUFR: messages/reports =          3229 /       389062  ntread =
           1
 new vad flag::  F
 READ_PREPBUFR: messages/reports =          3229 /       389062  ntread =
           1
```

● ● ● ●

```
 mesonetuselist: listexist,nprov=  F           0
 w_rejectlist: wlistexist,nwrjs=  F           0
```

● ● ● ●

```
 READ_PREPBUFR          : file=prepbufr        type=sst        sis=sst
nread=         0 ithin= 0 rmesh= 120.000000 isfcalc= 0 ndata=         0 ntask=  1
 READ_GPS:  time outside window    -3.000000000000000   skip this report
 READ_GPS:   time outside window    -2.966666666666667   skip this report
```

● ● ● ●

```
 READ_GPS               : file=gpsrobufr       type=gps_ref    sis=gps
nread=      4327 ithin= 0 rmesh= 120.000000 isfcalc= 0 ndata=      1083 ntask=  1
 READ_BUFRTOVS          : file=amsuabufr       type=amsua      sis=amsua_metop-b
nread=       810 ithin= 2 rmesh=  60.000000 isfcalc= 0 ndata=       718 ntask=  1
 READ_BUFRTOVS          : file=amsuabufr       type=amsua      sis=amsua_n18
nread=     30690 ithin= 2 rmesh=  60.000000 isfcalc= 0 ndata=     25575 ntask=  1
```

● ● ● ●

Using the above output information, many details on the observations can be obtained. For example, the last line (bold) indicates that subroutine "*READ_BUFRTOVS*" was called to read in NOAA-18 AMSU-A (`sis=amsua_n18`) from the BUFR file "*amsuabufr*" (`file=amsuabufr`). Furthermore, this kind of data has 30690 observations in the file (`nread=30690`) and 25575 in analysis domain and time-window (`ndata=25575`). The data was thinned on a 60 km coarse grid (`rmesh=60.000000`).

The next step partitions observations into subdomains. The observation distribution is summarized below by listing the number of observations for each observation variable in each subdomain (see Section 4.4 for more information):

```
OBS_PARA: ps                      1381      2448      4579      6807
OBS_PARA: t                       2152      4009      6886     10342
OBS_PARA: q                       1907      3491      5798      7340
OBS_PARA: pw                       110       152       128        79
OBS_PARA: uv                      2871      5393      8289     10802
OBS_PARA: gps_ref                    0        61       496       526
OBS_PARA: hirs4     metop-a          0         0       108       280
OBS_PARA: hirs4     metop-b          0         0         0        34
OBS_PARA: amsua     n15            771       937       434       257
OBS_PARA: amsua     n18            766       838        84        31
OBS_PARA: amsua     metop-a          0         0        56       225
OBS_PARA: amsua     metop-b          0         0         0        48
OBS_PARA: mhs       n18           1058      1102       185        81
OBS_PARA: mhs       metop-a          0         0       120       311
OBS_PARA: mhs       metop-b          0         0         0       112
```

Information on ingesting background error statistics:

```
 m_berror_stats_reg::berror_read_bal_reg(PREBAL_REG):  get balance variables"
 berror_stats".  mype,nsigstat,nlatstat =          0          60           93
 m_berror_stats_reg::berror_read_wgt_reg(PREWGT_REG):  read error amplitudes "
 berror_stats".  mype,nsigstat,nlatstat =          0          60           93
 Assigned default statistics to variable
 oz
 Assigned default statistics to variable
 cw
```

From this point forward in the *stdout*, the output shows many repeated entries. This is because the information is written from inside the outer loop. Typically the outer loop is iterated twice.

For each outer loop, the work begins with the calculation of the observation innovation. This calculation is done by the subroutine *setuprhsall*, which sets up the right hand side (rhs) of the analysis equation. This information is contained within the ***stdout*** file, which is shown in the following sections:

Start the first outer analysis loop:

```
GLBSOI: jiter,jiterstart,jiterlast,jiterend=           1           1
           2           1
```

Calculate observation innovation for each data type in the first outer loop:

```
 SETUPALL:,obstype,isis,nreal,nchanl=  ps        ps                      19      0
 SETUPALL:,obstype,isis,nreal,nchanl=  t         t                       25      0
 ● ● ● ●
 SETUPALL:,obstype,isis,nreal,nchanl=  uv        uv                      24      0
 SETUPALL:,obstype,isis,nreal,nchanl=  amsua     amsua_n15               33     15
 crtm_interface*init_crtm: crtm_init() on path "./"
 ACCoeff_ReadFile(Binary)(INFORMATION) : FILE: ./amsua_n15.SpcCoeff.bin; ^M
 ACCoeff RELEASE.VERSION:  1.04^M
   N_FOVS=30  N_CHANNELS=15
 SpcCoeff_ReadFile(Binary)(INFORMATION) : FILE: ./amsua_n15.SpcCoeff.bin; ^M
```

```
 SpcCoeff RELEASE.VERSION:  8.01^M
   N_CHANNELS=15
 Read_ODPS_Binary(INFORMATION) : FILE: ./amsua_n15.TauCoeff.bin; ^M
   ODPS RELEASE.VERSION:  2.01  N_LAYERS=100  N_COMPONENTS=2  N_ABSORBERS=1  N_CHANNELS=15
N_COEFFS=21600
 SETUPALL:,obstype,isis,nreal,nchanl=  gps_ref   gps                      16      0
```

● ● ● ●

In the above section, when computing the radiance observation innovation, information on reading in CRTM coefficients follows SETUPALL information. In ***stdout***, only information related to available radiance data are printed. The complete innovation can be found in the diagnostic files for each observation (for details see Appendix A.2):

● ● ● ●

```
 MWwaterCoeff_ReadFile(INFORMATION) : FILE: ./FASTEM5.MWwater.EmisCoeff.bin; ^M
 MWwaterCoeff RELEASE.VERSION: 1.5
 SETUPRAD:  write header record for mhs_n18                        7
          30            8            0            0           17            0
       30303  to file pe0000.mhs_n18_01    2014061700
```

The inner iteration of the first outer loop is discussed in the example below. In this example, the maximum number of iterations is 10.

Print cost function values for each inner iteration (see section 4.6 for more details):

```
GLBSOI:  START pcgsoi jiter=           1
pcgsoi: gnorm(1:2),b=  9.869857497554413276E+05  9.869857497554413276E+05
0.00000000000000000E+00
Initial cost function =  3.91593070716583970 4E+04
Initial gradient norm =  9.934715646436194447E+02
cost,grad,step,b,step? =   1   0  3.91593070716583970 4E+04  9.934715646436194447E+02
4.821051367939106942E-03  0.00000000000000000E+00  good
pcgsoi: gnorm(1:2),b=  4.101618851769856410E+05  4.101618851769853500E+05
4.155702200144395508E-01
cost,grad,step,b,step? =   1   1  3.440099807266351127E+04  6.404388223530688720E+02
5.837551195665082078E-03  4.155702200144395508E-01  good
pcgsoi: gnorm(1:2),b=  4.679752995326447999E+05  4.679752995326457312E+05
1.140952673675940110E+00
cost,grad,step,b,step? =   1   2  3.200665706943234181E+04  6.840872017021256397E+02
3.727016244273905453E-03  1.140952673675940110E+00  good
pcgsoi: gnorm(1:2),b=  2.579351145588153449E+05  2.579351145588165091E+05
5.511724973869557287E-01
cost,grad,step,b,step? =   1   3  3.026250552615522611E+04  5.078731284078882595E+02
4.372546275838497601E-03  5.511724973869557287E-01  good
pcgsoi: gnorm(1:2),b=  1.593388126627019956E+05  1.593388126627000165E+05
6.177476569455880862E-01
```

● ● ● ●

```
cost,grad,step,b,step? =   1   8  2.569146761032943687E+04  2.527962185369054851E+02
8.469805468189874412E-03  8.691587878189316330E-01  good
pcgsoi: gnorm(1:2),b=  6.309379012305534707E+04  6.309379012305517244E+04
9.872916643640710088E-01
cost,grad,step,b,step? =   1   9  2.515019683100273687E+04  2.511847728725914237E+02
5.552768045789822915E-03  9.872916643640710088E-01  good
pcgsoi: gnorm(1:2),b=  5.868194961445817898E+04  5.868194961445837544E+04
9.300748853414525508E-01
cost,grad,step,b,step? =   1  10  2.479985164931967302E+04  2.422435749704379191E+02
5.481216572289848883E-03  9.300748853414525508E-01  good
 update_guess: successfully complete
```

At the end of the 1st outer loop, print some diagnostics about the guess fields after adding the analysis increment to the guess and diagnostics about the analysis increment:

```
================================================================================
Status    Var           Mean                Min                 Max
analysis U      6.040957815192E+00  -1.828465093567E+01   5.595873981948E+01
analysis V      6.835165504865E-01  -6.569889456888E+01   6.086999970875E+01
analysis TV     2.466335068766E+02   1.953294535496E+02   3.159690767006E+02
analysis Q      2.510745886076E-03   1.000000000000E-07   2.137398860200E-02
analysis TSEN   2.461907415511E+02   1.953289096784E+02   3.154048262968E+02
analysis OZ     1.000000000007E-15   1.000000000000E-15   1.000000000000E-15
analysis CW     0.000000000000E+00   0.000000000000E+00   0.000000000000E+00
analysis DIV    0.000000000000E+00   0.000000000000E+00   0.000000000000E+00
analysis VOR    0.000000000000E+00   0.000000000000E+00   0.000000000000E+00
analysis PRSL   4.013978372499E+01   2.149497537431E+00   1.021332138217E+02
analysis PS     9.566205729377E+01   6.561118046179E+01   1.022389554483E+02
analysis SST    2.971105685225E+02   2.715000000000E+02   3.256103210449E+02
analysis radb   1.003633515395E-04  -2.202690500000E+01   2.936838452385E+01
analysis pcpb   0.000000000000E+00   0.000000000000E+00   0.000000000000E+00
analysis aftb   0.000000000000E+00   0.000000000000E+00   0.000000000000E+00
================================================================================
increment u    -2.626278681200E-03  -9.077837823593E+00   6.188598900645E+00
increment v    -4.995984356834E-04  -7.909683120587E+00   5.537231605449E+00
increment tv    2.498565835242E-02  -2.105477862842E+00   3.773668379528E+00
increment tsen  3.752368287894E-02  -2.104470453387E+00   3.987288657004E+00
increment q    -6.936457638776E-05  -4.329778521390E-03   3.071138926345E-03
increment oz    0.000000000000E+00   0.000000000000E+00   0.000000000000E+00
increment cw    0.000000000000E+00   0.000000000000E+00   0.000000000000E+00
increment prse  6.365609109435E-03  -5.520578941680E-02   1.221782252937E-01
increment ps    1.556033036020E-02  -5.520578941680E-02   1.221782252937E-01
increment sst  -5.271593752868E-02  -4.435488608468E-01   2.530089177442E-01
```

Start the second outer loop.

```
GLBSOI: jiter,jiterstart,jiterlast,jiterend=           2           1
           2           1
```

Calculate observation innovations for each data type in the second outer loop:

```
 SETUPALL:,obstype,isis,nreal,nchanl=  ps        ps                    19      0
 SETUPALL:,obstype,isis,nreal,nchanl=  t         t                     25      0
```

● ● ● ●

When calculating the radiance data innovation, there is no need to read in CRTM coefficients again because they were already read in the first outer loop:

```
 SETUPALL:,obstype,isis,nreal,nchanl=  ps        ps                    19      0
 SETUPALL:,obstype,isis,nreal,nchanl=  t         t                     25      0
 SETUPALL:,obstype,isis,nreal,nchanl=  q         q                     22      0
 SETUPALL:,obstype,isis,nreal,nchanl=  pw        pw                    16      0
 SETUPALL:,obstype,isis,nreal,nchanl=  uv        uv                    24      0
 SETUPALL:,obstype,isis,nreal,nchanl=  amsua     amsua_n15             33     15
 SETUPALL:,obstype,isis,nreal,nchanl=  gps_ref   gps                   16      0
 SETUPALL:,obstype,isis,nreal,nchanl=  hirs4     hirs4_metop-a         33     19
 SETUPALL:,obstype,isis,nreal,nchanl=  hirs4     hirs4_metop-b         33     19
 SETUPALL:,obstype,isis,nreal,nchanl=  amsua     amsua_n18             33     15
 SETUPALL:,obstype,isis,nreal,nchanl=  amsua     amsua_metop-a         33     15
 SETUPALL:,obstype,isis,nreal,nchanl=  mhs       mhs_metop-a           33      5
 SETUPALL:,obstype,isis,nreal,nchanl=  amsua     amsua_metop-b         33     15
 SETUPALL:,obstype,isis,nreal,nchanl=  mhs       mhs_metop-b           33      5
 SETUPALL:,obstype,isis,nreal,nchanl=  mhs       mhs_n18               33      5
```

The output from the inner iterations in the second outer loop is shown below. In this example, the maximum number of iterations is 10.

Print cost function values for each inner iteration (see section 4.6 for more details):

```
GLBSOI:  START pcgsoi jiter=           2
pcgsoi: gnorm(1:2),b=  1.79892192768119915 9E+05  1.79892192768119945 0E+05
0.000000000000000000E+00
Initial cost function =  2.7929197827499319 83E+04
Initial gradient norm =  4.2413699764123373 37E+02
cost,grad,step,b,step? =   2   0  2.7929197827499319 83E+04  4.2413699764123373 37E+02
4.3012695270614924 66E-03  0.000000000000000000E+00  good
pcgsoi: gnorm(1:2),b=  1.6417999665988184 28E+05  1.6417999665988137 71E+05
9.1265770978459592 74E-01
cost,grad,step,b,step? =   2   1  2.7155433020589534 86E+04  4.0519130871710689 23E+02
3.9650377706833645 97E-03  9.1265770978459592 74E-01  good
pcgsoi: gnorm(1:2),b=  7.3699586998810758 81E+04  7.3699586998811428 20E+04
4.4889504506135896 60E-01
cost,grad,step,b,step? =   2   2  2.6504453132642443 96E+04  2.7147667855418217 19E+02
6.0658782253718620 40E-03  4.4889504506135896 60E-01  good
pcgsoi: gnorm(1:2),b=  5.0101172457255263 00E+04  5.0101172457255204 79E+04
6.7980262166277338 75E-01
cost,grad,step,b,step? =   2   3  2.6057400412647471 97E+04  2.2383291191702630 84E+02
5.2330568509557544 65E-03  6.7980262166277338 75E-01  good
pcgsoi: gnorm(1:2),b=  4.0367975170950361 64E+04  4.0367975170950470 78E+04
8.0572915145630874 53E-01
```

• • • •

```
cost,grad,step,b,step? =   2   8  2.5034504860600896 79E+04  1.5283772154045314 55E+02
6.4176950373118725 12E-03  9.4801601378828959 10E-01  good
pcgsoi: gnorm(1:2),b=  2.2244313738363769 52E+04  2.2244313738363562 15E+04
9.5226517542856736 75E-01
cost,grad,step,b,step? =   2   9  2.4884591553288304 57E+04  1.4914527729151791 39E+02
6.3493768707554446 36E-03  9.5226517542856736 75E-01  good
pcgsoi: gnorm(1:2),b=  2.3502188299896388 34E+04  2.3502188299896817 63E+04
1.0565481397326117 46E+00
cost,grad,step,b,step? =   2  10  2.4743354022132094 54E+04  1.5330423444868176 83E+02
5.3048802181204137 57E-03  1.0565481397326117 46E+00  good
 update_guess: successfully complete
```

Diagnostics of the analysis results after adding the analysis increment to the guess and diagnostics about the analysis increment:

```
================================================================================
Status   Var          Mean                Min                 Max
analysis U      6.029185523039E+00  -1.833812625817E+01   5.611133032500E+01
analysis V      6.919134224301E-01  -6.592126368670E+01   6.153003826608E+01
analysis TV     2.466579426905E+02   1.951513166597E+02   3.160402684608E+02
analysis Q      2.512044912483E-03   1.000000000000E-07   2.172157790530E-02
analysis TSEN   2.462150351963E+02   1.951507590181E+02   3.155030414357E+02
analysis OZ     1.000000000007E-15   1.000000000000E-15   1.000000000000E-15
analysis CW     0.000000000000E+00   0.000000000000E+00   0.000000000000E+00
analysis DIV    0.000000000000E+00   0.000000000000E+00   0.000000000000E+00
analysis VOR    0.000000000000E+00   0.000000000000E+00   0.000000000000E+00
analysis PRSL   4.014612104303E+01   2.149486271929E+00   1.021387177834E+02
analysis PS     9.565956233603E+01   6.561840959883E+01   1.022346118144E+02
analysis SST    2.971105685225E+02   2.715000000000E+02   3.256103210449E+02
analysis radb  -3.135709649226E-05  -2.202690500000E+01   2.869157618807E+01
analysis pcpb   0.000000000000E+00   0.000000000000E+00   0.000000000000E+00
analysis aftb   0.000000000000E+00   0.000000000000E+00   0.000000000000E+00
increment u    -1.177229215321E-02  -1.990912132779E+00   1.781343510042E+00
increment v     8.396871943500E-03  -1.662604259770E+00   2.385299833914E+00
increment tv    2.443581394505E-02  -7.125013342967E-01   1.286937333545E+00
increment tsen  2.429237396636E-02  -7.186716504770E-01   1.417025779678E+00
increment q     1.299013546380E-06  -1.555485299435E-03   1.137697150017E-03
```

```
increment oz    0.000000000000E+00   0.000000000000E+00   0.000000000000E+00
increment cw    0.000000000000E+00   0.000000000000E+00   0.000000000000E+00
increment prse -1.020667642535E-03  -2.293656161864E-02   2.049371524411E-02
increment ps   -2.494957738179E-03  -2.293656161864E-02   2.049371524411E-02
increment sst  -4.778330400957E-02  -4.305443971837E-01   3.142760467828E-01
```

Because the outer loop is set to 2, the completion of the 2nd outer loop is the end of the analysis. The next step is to save the analysis results. Again, only a portion of variable `T` is shown and all other variables are listed according to variable name in the NetCDF file (`rmse_var` = …). The maximum and minimum values are useful information for a quick check of the reasonableness of the analysis:

```
 at 2 in wrwrfmassa
 update sigf03
  at 3 in wrwrfmassa
  at 6 in wrwrfmassa
  at 10.11 in wrwrfmassa,max,min(temp1)=   2.1961091E-02   1.3487200E-03
  at 10.12 in wrwrfmassa,max,min(tempa)=    0.000000        0.000000
  at 10.13 in wrwrfmassa,max,min(tempa)=    0.000000       -2.1961091E-02
  at 10.14 in wrwrfmassa,max,min(temp1)=    0.000000        0.000000
  iy,m,d,h,m,s=         2014            6           17            0
            0           0
  nlon,lat,sig_regional=          332          215           50
  rmse_var=P_TOP
  ordering=0
  WrfType,WRF_REAL=          104          104
  ndim1=            0
  staggering= N/A
  start_index=            1            1            1            0
  end_index1=          332          215           50            0
  p_top=    2000.000
  rmse_var=MUB
  ordering=XY
  WrfType,WRF_REAL=          104          104
  ndim1=            2
  staggering= N/A
  start_index=            1            1            1            0
  end_index1=          332          215           50            0
  max,min MUB=    99101.85        62163.48
  max,min psfc=    102021.6        65586.03
  max,min MU=    2021.641       -400.4141
  rmse_var=MU
  ordering=XY
  WrfType,WRF_REAL=          104          104
  ndim1=            2
  staggering= N/A
  start_index=            1            1            1            0
  end_index1=          332          215           50            0
  k,max,min,mid T=           1    321.6622        279.6388        309.0912
  k,max,min,mid T=           2    321.6799        280.6721        309.1487

• • • •

  k,max,min,mid T=          49    653.1034        614.2672        634.6012
  k,max,min,mid T=          50    684.9056        655.0489        672.6349

  rmse_var=T

• • • •

  rmse_var=QVAPOR

• • • •

  rmse_var=U

• • • •

  rmse_var=V

• • • •

  rmse_var=SEAICE

• • • •
```

```
 rmse_var=SST
 rmse_var=TSK
 rmse_var=Q2
```

After completion of the analysis, the subroutine *setuprhsall* is called again if `write_diag(3)=.true.`, to calculate analysis O-A information (the third time seeing this information):

```
 SETUPALL:,obstype,isis,nreal,nchanl=  ps        ps                      19      0
 SETUPALL:,obstype,isis,nreal,nchanl=  t         t                       25      0
```

```
 SETUPALL:,obstype,isis,nreal,nchanl=  mhs       mhs_metop-b             33      5
 SETUPRAD:  write header record for mhs_metop-b                    7
          30            8            0           0          17           0
       30303  to file pe0003.mhs_metop-b_03    2014061700
```

Deallocate the data arrays and finalize the GSI run:

```
 gsi_metguess_mod*destroy_: dealloc() for met-guess done
 observer_final: successfully finalized
 glbsoi: complete
[000]gsisub(): : complete.
```

The end of the GSI analysis (a successful analysis must reach this end, but to reach this end is not necessarily a successful analysis), which shows the time of ending this GSI run and some additional resource statistics:

```
ENDING DATE-TIME    JUL 16,2015  10:53:24.640  197  THU   2457220
     PROGRAM GSI_ANL HAS ENDED.
* . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * .
****************RESOURCE STATISTICS*******************************
The total amount of wall time                       = 138.108109
The total amount of time in user mode               = 133.766664
The total amount of time in sys mode                = 2.703588
The maximum resident set size (KB)                  = 1930848
Number of page faults without I/O activity          = 111372
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                = 6448
Number of InVoluntary Context Switches              = 1057
*****************END OF RESOURCE STATISTICS*************************
```

Different GSI applications may write out slightly different stdout information but the major flow and information are the same. A good knowledge of the **stdout** file gives users a clear picture how GSI runs through each part and the key information on a GSI run like data distribution and inner iterations.

4.2 Single Observation Test

A single observation test is a GSI run with only one (pseudo) observation at a specific location of the analysis domain. By examining the analysis increments from a single observation test, one can visualize the important features of the analysis, such as the ratio of background error and observation error variance and the pattern of the background error covariance. Therefore, the single observation test is the first check that users should do after successfully installing the GSI.

4.2.1 Setup a single observation test:

To perform the single observation test with the GSI, the following GSI namelist variables need to be set, which should be done through editing the script *run/comgsi_namelist.sh*:

Under the `&SETUP` section, turn on the single observation test:

```
oneobtest=.true.,
```

under the `&SINGLEOB_TEST` section, set up single observation features like:

```
maginnov=1.0,
magoberr=0.8,
oneob_type='t',
oblat=38.,
oblon=262.,
obpres=500.,
obdattim= 2014061700,
obhourset=0.,
```

Note:

- Please check Appendix C in this User's Guide for the explanation of each parameter. From these parameters, we can see that a useful observation in the analysis should include information like the observation type (`oneob_type`), value (`maginnov`), error (`magoberr`), location (`oblat, oblong, obpres`) and time (`obdattim, obhourset`). Users can dump out (use *ncdump*) the global attributes from the NetCDF background file and set `oblat`=*CEN_LAT*, `oblong`=*360-CEN_LON* to have the observation at the center of the domain.
- In the analysis, the GSI first generates a prepbufr file including only one observation based on the information given in the namelist &`SINGLEOB_TEST` section. To generate this prepbufr file, the GSI needs to read in a PrepBUFR table, which is not needed when running a GSI analysis with real observations. The BUFR table is in the *fix/* directory and needs to be copied to the run directory. We have put the following lines in the GSI run script for the single observation test:

```
bufrtable=${FIX_ROOT}/prepobs_prep.bufrtable
cp $bufrtable ./prepobs_prep.bufrtable
```

4.2.2. Examples of single observation tests for GSI

Figure 4.1 is a single observation test that has a temperature observation (`oneob_type='t'`) with a 1 degree innovation (`maginnov=1.0`) and a 0.8 degree observation error (`magoberr=0.8`). The background error covariance from global was picked for better illustration.

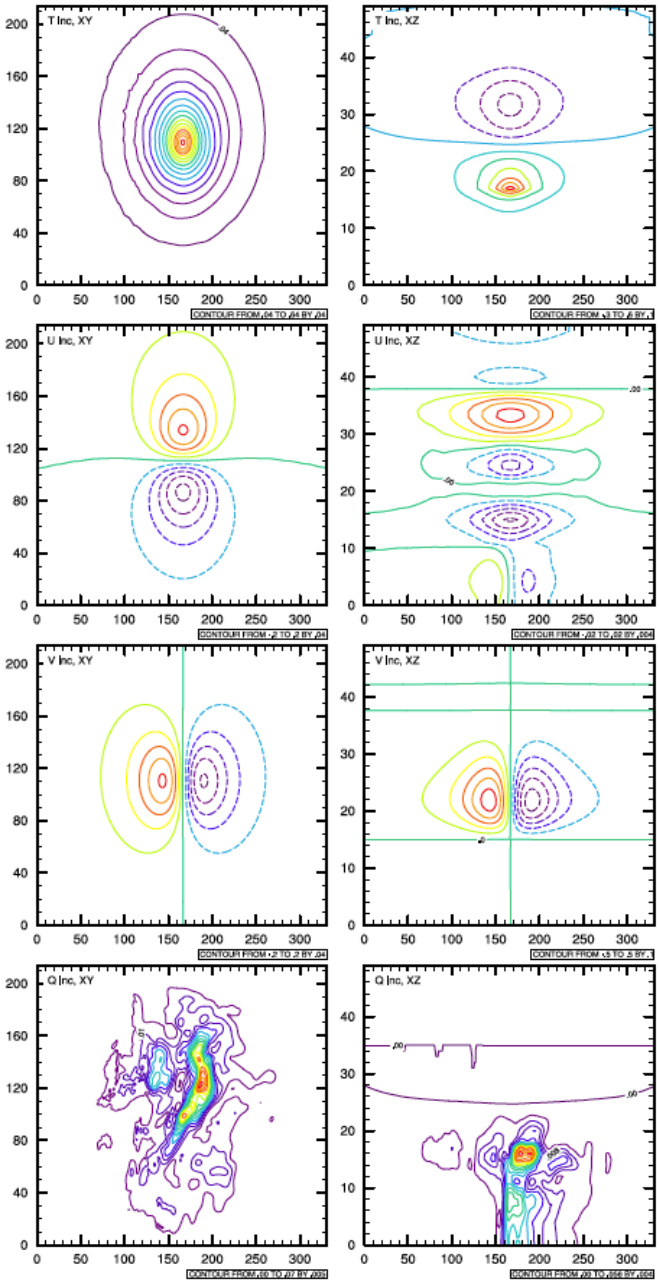

Fig. 4.1. Horizontal cross sections (left column) and vertical cross sections (right column) of analysis increment of T, U, V, and Q from a single T observation

This single observation was located at the center of the domain. The results are shown with figures of the horizontal and vertical cross sections through the point of maximum analysis increment. The above figure was generated using NCL scripts, which can be found in the *util/Analysis_Utilities/plots_ncl* directory, introduced in Section A.4.

4.3 Control Data Usage

Observation data used in the GSI analysis can be controlled through three parts of the GSI system:

1. In GSI run script, through linking observation BUFR files to working directory
2. In GSI namelist (inside *comgsi_namelist.sh*), through section &OBS_INPUT
3. Through parameters in info files (e.g.: convinfo, satinfo, etc)

Each part gives different levels of control to the data usage in the GSI, which is introduced below:

1. Link observation BUFR files to working directory in GSI run script:

All BUFR/PrepBUFR observation files need to be linked to the working directory with GSI recognized names before can be used by GSI analysis. The run script (*run_gsi_regional.ksh*) makes these links after locating the working directory. Turning on or off these links can control the use of all the data contained in the BUFR files. Table 3.1 provides a list of all default observation file names recognized by GSI and the corresponding examples of the observation BUFR files from NCEP. The following is the first 3 rows of the table as an example:

GSI Name	Content	Example file names
prepbufr	Conventional observations, including ps, t, q, pw, uv, spd, dw, sst, from observation platforms such as METAR, sounding, et al.	*gdas1.t12z.prepbufr*
satwndbufr	satellite winds	*gdas1.t12z.satwnd.tm00.bufr_d*
amsuabufr	AMSU-A 1b radiance (brightness temperatures) from satellites NOAA-15, 16, 17,18, 19 and METOP-A/B	*gdas1.t12z.1bamua.tm00.bufr_d*

The left column is the GSI recognized name (bold) and the right column are names of BUFR files from NCEP (italic). In the run script, the following lines are used to link the BUFR files in the right column to the working directory using the GSI recognized names shown in the left column:

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

# Link to the radiance data
 ln -s ${OBS_ROOT}/gdas1.t12z.1bamua.tm00.bufr_d amsuabufr
```

The GSI recognized default observation filenames are set up in the namelist section &OBS_INPUT, which certainly can be changed based on application needs (details see below).

2. In GSI namelist (inside *comgsi_namelist.sh*), section &OBS_INPUT:

In this namelist section, observation files (column of **dfile**) are tied to the observation variables used inside the GSI code (column of **dsis**), for example, part of section OBS_INPUT shows:

```
&OBS_INPUT
   dmesh(1)=120.0,dmesh(2)=60.0,dmesh(3)=30,time_window_max=1.5,ext_sonde=.true.,
 /
OBS_INPUT::
!  dfile          dtype       dplat     dsis                dval    dthin dsfcalc
   prepbufr       ps          null      ps                  1.0     0     0
   prepbufr       t           null      t                   1.0     0     0
   prepbufr       q           null      q                   1.0     0     0
   prepbufr       pw          null      pw                  1.0     0     0
   satwndbufr     uv          null      uv                  1.0     0     0
   prepbufr       uv          null      uv                  1.0     0     0
   prepbufr       spd         null      spd                 1.0     0     0
   prepbufr       dw          null      dw                  1.0     0     0
   radarbufr      rw          null      rw                  1.0     0     0
   prepbufr       sst         null      sst                 1.0     0     0
   gpsrobufr      gps_ref     null      gps                 1.0     0     0
   ssmirrbufr     pcp_ssmi    dmsp      pcp_ssmi            1.0    -1     0
   ● ● ● ●
   amsuabufr      amsua       n15       amsua_n15          10.0     2     0
   amsuabufr      amsua       n18       amsua_n18          10.0     2     0
   ● ● ● ●
```

This setup tells GSI that conventional observation variables "`ps`", "`t`", and "`q`" should be read in from the file *prepbufr* and AMSU-A radiances from NOAA-15 and -18 satellites should be read in from the file *amsuabufr*. Deleting a particular line in `&OBS_INPUT` will turn off the use of the observation variable presented by the line in the GSI analysis but other variables under the same type still can be used. For example, if we delete:

```
amsuabufr      amsua       n15       amsua_n15          10.0     2     0
```

Then, the AMSU-A observation from NOAA-15 will not be used in the analysis but the AMSU-A observations from NOAA-18 will still be used.

The observation filename in *dfile* can be different from the sample script (*comgsi_namelist.ksh*). If the filename in *dfile* has been changed, the link from the BUFR files to the GSI recognized name in the run script also needs to be changed correspondingly. For example, if we change the *dfile* for `amsuabufr` file for NOAA-15 to be `amsuabufr_n15`

```
amsuabufr_n15  amsua       n15       amsua_n15          10.0     2     0
amsuabufr      amsua       n18       amsua_n18          10.0     2     0
```

Then a new link needs to be added in the run script:

```
# Link to the radiance data
ln -s ${OBS_ROOT}/le_gdas1.t00z.1bamua.tm00.bufr_d amsuabufr
ln -s ${OBS_ROOT}/le_gdas1.t00z.1bamua.tm00.bufr_d amsuabufr_n15
```

The GSI will read NOAA-18 AMSU-A observations from file `amsuabufr` and NOAA-15 AMSU-A observations from file `amsuabufr_n15` based on the above changes to the run scripts and namelist. In this example, both `amsuabufr` and `amsuabufr_15` are linked to the same BUFR file and NOAA-15 AMSU-A and NOAA-18 AMSU-A observations are still read in from the same BUFR file. If `amsuabufr` and `amsuabufr_15` link to different BUFR files, then NOAA-15 AMSU-A and NOAA-18 AMSU-A will be read in from different BUFR files. Clearly, the changeable filename in *dfile* gives GSI more capability to handle multiple data resources.

3. Use info files to control data usage

For each variable, observations can come from multiple platforms (data types or observation instruments). For example, surface pressure (ps) can come from METAR observation stations (data type 187) and Rawinsonde (data type 120). There are several files named *info in the GSI system (located in *./fix*) to control the usage of observations based on the observation platform. Table 4.1 is a list of info files and their function:

Table 4.1 The content of info files

File name in GSI	Function and Content
convinfo	Control the usage of conventional data, including *tcp, ps, t, q, pw, sst, uv, spd, dw,* radial wind (Level 2 *rw* and 2.5 *srw*), *gps, pm2_5*
satinfo	Control the usage of satellite data. Instruments include AMSU-A/B, HIRS3/4, MHS, ssmi, ssmis, iasi, airs, sndr, cris, amsre, imgr, seviri, atms, avhrr3, etc. and satellites include NOAA 15, 17, 18, 19, aqua, GOES 11, 12, 13, METOP-A/B, NPP, DMSP 15,16,17,18,19,20, M08, M09, M10, etc.
ozinfo	Control the usage of ozone data, including sbuv6, 8 from NOAA 14, 16, 17, 18, 19. omi_aura, gome_metop-a, mls_aura
pcpinfo	Control the usage of precipitation data, including pcp_ssmi, pcp_tmi
aeroinfo	Control the usage of aerosol data, including modis_aqua and modis_terra

The header of each info file includes an explanation of the content of the file. Here we discuss the most commonly used two info files:

- **convinfo**

  The convinfo is to control the usage of conventional data. The following is the part of the content of *convinfo*:

```
 !otype   type  sub iuse twindow numgrp ngroup nmiter gross ermax ermin var_b    var_pg
ithin rmesh  pmesh  npred  pmot  ptime
  tcp      112    0    1     3.0      0      0      0  75.0   5.0   1.0  75.0  0.000000
0    0.     0.      0    0.     0.
  ps       120    0    1     3.0      0      0      0   4.0   3.0   1.0   4.0  0.000300
0    0.     0.      0    0.     0.
```

```
  ps       132    0   -1     3.0      0       0       0   4.0   3.0   1.0   4.0  0.000300
0    0.     0.      0    0.     0.
  ps       180    0    1     3.0      0       0       0   4.0   3.0   1.0   4.0  0.000300
0    0.     0.      0    0.     0.
  ps       181    0    1     3.0      0       0       0   3.6   3.0   1.0   3.6  0.000300
0    0.     0.      0    0.     0.
  ps       182    0    1     3.0      0       0       0   4.0   3.0   1.0   4.0  0.000300
0    0.     0.      0    0.     0.
  ps       183    0   -1     3.0      0       0       0   4.0   3.0   1.0   4.0  0.000300
0    0.     0.      0    0.     0.
  ps       187    0    1     3.0      0       0       0   4.0   3.0   1.0   4.0  0.000300
0    0.     0.      0    0.     0.
  t        120    0    1     3.0      0       0       0   8.0   5.6   1.3   8.0  0.000001
0    0.     0.      0    0.     0.
  t        126    0   -1     3.0      0       0       0   8.0   5.6   1.3   8.0  0.001000
0    0.     0.      0    0.     0.
  t        130    0    1     3.0      0       0       0   7.0   5.6   1.3   7.0  0.001000
0    0.     0.      0    0.     0.
  t        131    0    1     3.0      0       0       0   7.0   5.6   1.3   7.0  0.001000
0    0.     0.      0    0.     0.
  t        132    0    1     3.0      0       0       0   7.0   5.6   1.3   7.0  0.001000
0    0.     0.      0    0.     0.
  t        133    0    1     3.0      0       0       0   7.0   5.6   1.3   7.0  0.004000
0    0.     0.      0    0.     0.
  t        134    0   -1     3.0      0       0       0   7.0   5.6   1.3   7.0  0.004000
0    0.     0.      0    0.     0.
  t        135    0   -1     3.0      0       0       0   7.0   5.6   1.3   7.0  0.004000
0    0.     0.      0    0.     0.
  t        180    0    1     3.0      0       0       0   7.0   5.6   1.3   7.0  0.004000
0    0.     0.      0    0.     0.
  t        181    0   -1     3.0      0       0       0   7.0   5.6   1.3   7.0  0.004000
0    0.     0.      0    0.     0.
  t        182    0    1     3.0      0       0       0   7.0   5.6   1.3   7.0  0.004000
0    0.     0.      0    0.     0.
```


The meaning of each column is explained in the header of the file and is listed in Table 4.2.

Table 4.2 list of the content for each *convinfo* column

Column Name	Content of the column
`otype`	observation variables (t, uv, q, etc.)
`type`	prepbufr observation type (if available)
`sub`	prepbufr subtype (not yet available)
`iuse`	flag if to use/not use / monitor data =1, use data, the data type will be read and used in the analysis after quality control =0, read in and process data, use for quality control, but do NOT assimilate =-1, monitor data. This data type will be read in and monitored but not be used in the GSI analysis
`twindow`	time window (+/- hours) for data used in the analysis
`numgrp`	cross validation parameter - number of groups
`ngroup`	cross validation parameter - group to remove from data use
`nmiter`	cross validation parameter - external iteration to introduce removed data
`gross`	gross error parameter - gross error
`ermax`	gross error parameter – maximum error
`ermin`	gross error parameter – minimum error
`var_b`	variational quality control parameter - b parameter
`var_pg`	variational quality control parameter - pg parameter
`ithin`	Flag to turn on thinning (0, no thinning, 1 - thinning)

rmesh	size of horizontal thinning mesh (in kilometers)
pmesh	size of vertical thinning mesh
npred	Number of bias correction predictors
pmot	the option to keep thinned data as monitored, 0: not to keep, other values: to keep
ptime	time interval for thinning, 0, no temporal thinning, other values define time interval (less than 6)

From this table, we can see that parameter `iuse` is used to control the usage of data and parameter `twindow` is to control the time window of data usage. Parameters `gross`, `ermax`, and `ermin` are for gross quality control. Through these parameters, GSI can control how to use certain types of the data in the analysis.

- **satinfo:**

The *satinfo* file contains information about the channels, sensors, and satellites. It specifies observation error (cloudy or clear) for each channel, how to use the channels (assimilate, monitor, etc), and other useful information. The following is part of the content of *satinfo*. The meaning of each column is explained in Table 4.3.

```
!sensor/instr/sat      chan iuse  error  error_cld  ermax   var_b    var_pg  cld_det
 amsua_n15               1   1    3.000    9.100    4.500   10.000    0.000     -2
 amsua_n15               2   1    2.000   13.500    4.500   10.000    0.000     -2
 amsua_n15               3   1    2.000    7.100    4.500   10.000    0.000     -2
 amsua_n15               4   1    0.600    1.300    2.500   10.000    0.000     -2
 ● ● ● ●
 amsua_n15              14  -1    2.000    1.400    4.500   10.000    0.000     -2
 amsua_n15              15   1    3.000   10.000    4.500   10.000    0.000     -2
 hirs3_n17               1  -1    2.000    0.000    4.500   10.000    0.000     -1
 hirs3_n17               2  -1    0.600    0.000    2.500   10.000    0.000      1
 hirs3_n17               3  -1    0.530    0.000    2.500   10.000    0.000      1
 ● ● ● ●
```

Table 4.3: list of the content for each *satinfo* column

Column Name	Content of the column
sensor/instr/sat	Sensor, instrument, and satellite name
chan	Channel number for certain sensor
iuse	= 1, use this channel data =-1, don't use this channel data
error	Variance for each satellite channel
error_cld	Variance for each satellite channel if it is cloudy
ermax	Error maximum for gross check to observations
var_b	Possible range of variable for gross errors
var_pg	Probability of gross error
icld_det	Use this channel in cloud detection if > 0

4.4 Domain Partition for Parallelization and Observation Distribution

The standard output file (*stdout*) has an information block that shows the distribution of different kinds of observations in each sub-domain. This block follows the observation

input section. The following is the observation distribution of the case shown in Section 4.1. From the case introduction, we know the prepbufr (conventional data), radiance BUFR files, and GPS BUFR files were used. In this list, the conventional observations (`ps, t, q, pw, and uv`), GPSRO (`gps_ref`), and radiance data (`amusa, hirs4,` and `mhs` from `Metop-a, Metop-b, NOAA 15 and 18`) were distributed among 4 sub-domains:

```
OBS_PARA: ps                      1381      2448      4579      6807
OBS_PARA: t                       2152      4009      6886     10342
OBS_PARA: q                       1907      3491      5798      7340
OBS_PARA: pw                       110       152       128        79
OBS_PARA: uv                      2871      5393      8289     10802
OBS_PARA: gps_ref                    0        61       496       526
OBS_PARA: hirs4     metop-a          0         0       108       280
OBS_PARA: hirs4     metop-b          0         0         0        34
OBS_PARA: amsua     n15            771       937       434       257
OBS_PARA: amsua     n18            766       838        84        31
OBS_PARA: amsua     metop-a          0         0        56       225
OBS_PARA: amsua     metop-b          0         0         0        48
OBS_PARA: mhs       n18           1058      1102       185        81
OBS_PARA: mhs       metop-a          0         0       120       311
OBS_PARA: mhs       metop-b          0         0         0       112
```

This list is a good way to quickly check which kinds of data are used in the analysis and how they are distributed in the analysis domain.

4.5 Observation Innovation Statistics

The GSI analysis gives a group of files named *fort.2** to summarize observations fitting to the current solution in each outer loop (except for *fort.220*, see explanation on *fort.220* in next section). The content of some of these files is listed in Table 4.4:

Table 4.4 List of the content and units for each fort files

File name	Variables in file	Ranges/units
fort.201 or fit_p1.analysis_time	fit of surface pressure data	mb
fort.202 or fit_w1.analysis_time	fit of u, v wind data	m/s
fort.203 or fit_t1.analysis_time	fit of temperature data	K
fort.204 or fit_q1.analysis_time	fit of moisture data	percent of guess $q_{saturation}$
fort.205	fit of precipitation water data	mm
fort.206	fit of ozone observations from sbuv6_n14 (, _n16, _n17, _n18), sbuv8_n16 (, _n17, _n18, _n19), omi_aura, gome_metop-a/b, mls_aura	
fort.207 or fit_rad1.analysis_time	fit of satellite radiance data, such as: amsua_n15(, n16, n17, n18, metop-a, aqua, n19), amsub_n17, hirs3_n17, hirs4_n19 (, metop-a), etc	
fort.208	fit of prepcipitation rate (pcp_ssmi, pcp_tmi)	
fort.209	fit of radar radial wind (rw)	
fort.210	fit of lidar wind (dw)	

fort.211	fit of radar superob wind data (srw)	
fort.212	fit of GPS data (refractivity or bending angle)	fractional difference
fort.213	fit of conventional sst data	C
fort.214	Tropical cyclone central pressure	
fort.215	Lagrangian tracer data	
Fort.217	Fit of aerosol product (aod)	
Fort.218	Fit of wind gust	
Fort.219	Fit of visibility	

To help users understand the information inside these files, some examples from these files are given in the following sub-sections with corresponding explanations.

4.5.1 Conventional observations

Example of files including single level data (*fort.201, fort.205, fort.213*)

```
current fit of surface pressure data, ranges in mb
-------------------------------------------------
 pressure levels (hPa)=   0.0 2000.0
     it     obs    type stype    count      bias       rms      cpen     qcpen
 o-g 01      ps     120 0000        73    0.1795    0.6492    0.5876    0.5875
 o-g 01      ps     180 0000      2087    0.2765    0.8542    0.3952    0.3743
 o-g 01      ps     181 0000       260    0.1242    0.8486    0.7423    0.7291
 o-g 01      ps     182 0000         1    0.9171    0.9171    2.8963    2.8957
 o-g 01      ps     187 0000     10828    0.2040    0.7680    0.3084    0.2922
 o-g 01             all          13249    0.2138    0.7833    0.3323    0.3155
 o-g 01      ps rej 180 0000         3   -1.7385    4.6528    0.0000    0.0000
 o-g 01      ps rej 181 0000        35 -100.8366  160.6771    0.0000    0.0000
 o-g 01      ps rej 187 0000        45    4.1463    6.7348    0.0000    0.0000
 o-g 01         rej all             83  -40.3363  104.4611    0.0000    0.0000
 o-g 01      ps mon 132 0000         1    0.9171    0.9171    0.2103    0.2103
 o-g 01      ps mon 180 0000       143   -0.0110    0.7166    1.7622    1.4127
 o-g 01      ps mon 181 0000       242    0.1558    1.2462    2.2266    2.1942
 o-g 01      ps mon 183 0000      1165    0.3518    1.1231    0.0000    0.0000
 o-g 01      ps mon 187 0000       222   -0.2022    1.6370    8.4921    5.5832
 o-g 01         mon all           1773    0.2267    1.1909    1.5095    1.1126
```

Example of files including multiple level data (*fort.202, fort.203, fort.204*)

```
                          ptop  1000.0  900.0  800.0  600.0  100.0   50.0     0.0
    it     obs    type styp  pbot  1200.0 1000.0  900.0  800.0  150.0  100.0  2000.0
-------------------------------------------------------------------------------------
 o-g 01      uv     220 0000 count      44    223    223    478    437    519    4231
 o-g 01      uv     220 0000  bias    0.26   0.51   0.04   0.35   0.99   0.97    0.59
 o-g 01      uv     220 0000   rms    2.21   2.48   2.51   2.84   5.88   4.79    4.18
 o-g 01      uv     220 0000  cpen    0.31   0.38   0.44   0.61   1.64   1.18    0.90
 o-g 01      uv     220 0000 qcpen    0.31   0.38   0.44   0.60   1.63   1.18    0.89
 o-g 01      uv     223 0000 count       0      6     16     88     32      8     331
 o-g 01      uv     223 0000  bias    0.00   0.23  -0.29   0.85  -0.76  -4.89    0.27
 o-g 01      uv     223 0000   rms    0.00   1.28   1.32   2.86   5.21   6.63    3.76
 o-g 01      uv     223 0000  cpen    0.00   0.06   0.07   0.45   0.88   1.31    0.73
```

● ● ● ●

```
 o-g 01             all      count    1799   1726   2001   3050    469    527   13124
 o-g 01             all       bias    0.05   0.91   0.78   0.63   0.87   0.88    0.62
 o-g 01             all        rms    2.46   2.58   2.65   3.21   5.83   4.82    3.54
 o-g 01             all       cpen    0.24   0.24   0.30   0.45   1.58   1.19    0.52
 o-g 01             all      qcpen    0.23   0.24   0.30   0.44   1.58   1.19    0.52
```

```
 o-g 01      uv rej 220 0000 count       0      0      0      0      0      0    800
 o-g 01      uv rej 220 0000  bias    0.00   0.00   0.00   0.00   0.00   0.00   3.11
 o-g 01      uv rej 220 0000   rms    0.00   0.00   0.00   0.00   0.00   0.00   6.41
```

● ● ● ●

```
 o-g 01         rej all      count      23    108     21     41      2      0   1008
 o-g 01         rej all       bias   44.72   6.40  -0.30  -7.38  11.46   0.00   3.84
 o-g 01         rej all        rms   56.08  16.60   6.87  10.47  43.15   0.00  12.28
 o-g 01         rej all       cpen    0.00   0.00   0.00   0.00   0.00   0.00   0.00
 o-g 01         rej all      qcpen    0.00   0.00   0.00   0.00   0.00   0.00   0.00
 o-g 01      uv mon 220 0000 count       0      4      0      1      8      9     29
 o-g 01      uv mon 220 0000  bias    0.00   1.07   0.00   6.04  -1.03   0.23  -2.21
 o-g 01      uv mon 220 0000   rms    0.00  20.49   0.00  17.78   9.06   8.51  12.52
 o-g 01      uv mon 220 0000  cpen    0.00   0.00   0.00   0.00   0.00   0.00   0.00
 o-g 01      uv mon 220 0000 qcpen    0.00   0.00   0.00   0.00   0.00   0.00   0.00
```

● ● ● ●

```
 o-g 01         mon all      count    3573   7736   1004    563      8      9  13052
 o-g 01         mon all       bias   -0.01   0.40   0.05  -0.30  -1.03   0.23   0.23
 o-g 01         mon all        rms    2.55   3.25   4.67   5.98   9.06   8.51   3.60
 o-g 01         mon all       cpen    0.81   1.21   1.67   1.28   0.00   0.00   1.12
 o-g 01         mon all      qcpen    0.80   1.15   1.53   1.16   0.00   0.00   1.07
```

● ● ● ●

Please note 5 layers from 600 to 150 hPa have been deleted to make each row fit into one line. Only observation type 220 and 223 are shown as an example.

The following table lists the meaning of each item in file *fort.201-213* except file *fort.207*:

Table 4.5: list of each item in file fort.201-213 (except fort.207)

Name	Explanation
it	outer loop number = 01: observation – background = 02: observation – analysis (after 1st outer loop) = 03: observation – analysis (after 2nd outer loop)
obs	observation variable (such as *uv*, *ps*) and usage of the type, which include: *blank*: used in GSI analysis *mon*: monitored, (read in but not assimilated by GSI). *rej*: rejected because of quality control in GSI
type	prepbufr observation type (see BUFR User's Guide for details)
styp	prepbufr observation subtype (not used now)
ptop	for multiple level data: pressure at the top of the layer
pbot	for multiple level data: pressure at the bottom of the layer
count	The number of observations summarized under observation types and vertical layers
bias	Bias of observation departure for each outer loop (it)
rms	Root Mean Square of observation departure for each outer loop (it)
cpen	Observation part of penalty (cost function)
qcpen	nonlinear qc penalty

The contents of the fit files are calculated based on O-B or O-A for each observation. The detailed departure information about each observation is saved in the diagnostic files. For the content of the diagnostic files, please check the content of the array `rdiagbuf` in one of the setup subroutines for conventional data, for example, *setupt.f90*. We provide a tool in appendix A.2 to help users read in the information from the diagnostic files.

These fit files give lots of useful information on how data are analyzed by the GSI, such as how many observations are used and rejected, what is the bias and rms for certain data types or for all observations, and how analysis results fit to the observation before and after analysis. Again, we use observation type 220 in *fort.202* (*fit_w1.2014061700*) as an example to illustrate how to read this information. The fit information for observation type 220 (sounding observation) is listed below. Like the previous example, 5 layers from 600 to 150 hPa were deleted to make each row fit into one line. All fit information of observation type 220 are shown.

```
                                ptop  1000.0   900.0   800.0   600.0   100.0    50.0     0.0
     it     obs    type styp    pbot  1200.0  1000.0   900.0   800.0   150.0   100.0  2000.0
-------------------------------------------------------------------------------------------
 o-g 01      uv     220 0000 count       44     223     223     478     437     519    4231
 o-g 01      uv     220 0000  bias     0.26    0.51    0.04    0.35    0.99    0.97    0.59
 o-g 01      uv     220 0000   rms     2.21    2.48    2.51    2.84    5.88    4.79    4.18
 o-g 01      uv     220 0000  cpen     0.31    0.38    0.44    0.61    1.64    1.18    0.90
 o-g 01      uv     220 0000 qcpen     0.31    0.38    0.44    0.60    1.63    1.18    0.89
● ● ● ●
 o-g 01      uv rej 220 0000 count        0       0       0       0       0       0     800
 o-g 01      uv rej 220 0000  bias     0.00    0.00    0.00    0.00    0.00    0.00    3.11
 o-g 01      uv rej 220 0000   rms     0.00    0.00    0.00    0.00    0.00    0.00    6.41
 o-g 01      uv rej 220 0000  cpen     0.00    0.00    0.00    0.00    0.00    0.00    0.00
 o-g 01      uv rej 220 0000 qcpen     0.00    0.00    0.00    0.00    0.00    0.00    0.00

● ● ● ●
 o-g 01      uv mon 220 0000 count        0       4       0       1       8       9      29
 o-g 01      uv mon 220 0000  bias     0.00    1.07    0.00    6.04   -1.03    0.23   -2.21
 o-g 01      uv mon 220 0000   rms     0.00   20.49    0.00   17.78    9.06    8.51   12.52
 o-g 01      uv mon 220 0000  cpen     0.00    0.00    0.00    0.00    0.00    0.00    0.00
 o-g 01      uv mon 220 0000 qcpen     0.00    0.00    0.00    0.00    0.00    0.00    0.00



                                ptop  1000.0   900.0   800.0   600.0   100.0    50.0     0.0
     it     obs    type styp    pbot  1200.0  1000.0   900.0   800.0   150.0   100.0  2000.0
-------------------------------------------------------------------------------------------
 o-g 02      uv     220 0000 count       44     223     223     478     437     519    4231
 o-g 02      uv     220 0000  bias     0.21    0.05   -0.43    0.03    0.69    0.96    0.39
 o-g 02      uv     220 0000   rms     2.13    2.26    2.29    2.56    5.19    4.55    3.73
 o-g 02      uv     220 0000  cpen     0.32    0.31    0.37    0.50    1.27    1.07    0.71
 o-g 02      uv     220 0000 qcpen     0.32    0.31    0.37    0.50    1.27    1.07    0.71

● ● ● ●
 o-g 02      uv rej 220 0000 count        0       0       0       0       0       0     800
 o-g 02      uv rej 220 0000  bias     0.00    0.00    0.00    0.00    0.00    0.00    2.96
 o-g 02      uv rej 220 0000   rms     0.00    0.00    0.00    0.00    0.00    0.00    6.33
 o-g 02      uv rej 220 0000  cpen     0.00    0.00    0.00    0.00    0.00    0.00    0.00
 o-g 02      uv rej 220 0000 qcpen     0.00    0.00    0.00    0.00    0.00    0.00    0.00
```

• • • •

```
 o-g 02      uv mon 220 0000 count       0      4      0      1      8      9      29
 o-g 02      uv mon 220 0000  bias    0.00   2.16   0.00   5.80  -1.00   0.23   -0.82
 o-g 02      uv mon 220 0000   rms    0.00  18.44   0.00  12.60  10.31   8.58   10.87
 o-g 02      uv mon 220 0000  cpen    0.00   0.00   0.00   0.00   0.00   0.00    0.00
 o-g 02      uv mon 220 0000 qcpen    0.00   0.00   0.00   0.00   0.00   0.00    0.00
```

```
                              ptop  1000.0  900.0  800.0  600.0  100.0   50.0     0.0
    it     obs    type styp   pbot  1200.0 1000.0  900.0  800.0  150.0  100.0  2000.0
-------------------------------------------------------------------------------------
 o-g 03      uv     220 0000 count      44    223    223    478    437    519     4231
 o-g 03      uv     220 0000  bias    0.28   0.08  -0.32   0.11   0.74   1.00     0.43
 o-g 03      uv     220 0000   rms    2.11   2.17   2.20   2.38   5.00   4.46     3.60
 o-g 03      uv     220 0000  cpen    0.31   0.29   0.34   0.43   1.18   1.03     0.65
 o-g 03      uv     220 0000 qcpen    0.31   0.29   0.34   0.43   1.18   1.03     0.65
```

• • • •

```
 o-g 03      uv rej 220 0000 count       0      0      0      0      0      0      800
 o-g 03      uv rej 220 0000  bias    0.00   0.00   0.00   0.00   0.00   0.00    2.98
 o-g 03      uv rej 220 0000   rms    0.00   0.00   0.00   0.00   0.00   0.00    6.35
 o-g 03      uv rej 220 0000  cpen    0.00   0.00   0.00   0.00   0.00   0.00    0.00
 o-g 03      uv rej 220 0000 qcpen    0.00   0.00   0.00   0.00   0.00   0.00    0.00
```

• • • •

```
 o-g 03      uv mon 220 0000 count       0      4      0      1      8      9      29
 o-g 03      uv mon 220 0000  bias    0.00   1.86   0.00   6.09  -0.98   0.07   -0.94
 o-g 03      uv mon 220 0000   rms    0.00  18.76   0.00  12.59  10.34   8.69   11.01
 o-g 03      uv mon 220 0000  cpen    0.00   0.00   0.00   0.00   0.00   0.00    0.00
 o-g 03      uv mon 220 0000 qcpen    0.00   0.00   0.00   0.00   0.00   0.00    0.00
```

In loop section "*o-g 01*", from "*count*" line, we can see there are 4231 sounding observations used in the analysis. Among them, 44 are within the 1000-1200 hPa layer. Also from the "*count*" lines, in the rejection and monitoring section, there are 800 observations rejected and 29 observations being monitored. In the same loop section, from the "*bias*" line and "*rms*" lines, we can see the total bias and rms of O-B for soundings is 0.59 and 4.18. The bias and rms of each layer for sounding observation can also be found in the file.

When reading bias and rms values from different loops, as shown with the comparison in the following three lines:

```
o-g 01      uv     220 0000   rms    2.21   2.48   2.51   2.84   5.88   4.79    4.18
o-g 02      uv     220 0000   rms    2.13   2.26   2.29   2.56   5.19   4.55    3.73
o-g 03      uv     220 0000   rms    2.11   2.17   2.20   2.38   5.00   4.46    3.60
```

These three lines show that the rms reduced from 4.18 (o-g 01, which is O-B) to 3.73 (o-g 02, which is O-A after 1st outer loop) and then to 3.60 (o-g 03, which is O-A after 2nd outer loop, the final analysis result). The reduction in the rms shows the observation type 220 (sounding) was used in the GSI analysis to modify the background fields to fit to the observations. Please note this example only used 10 iterations for each outer loop.

4.5.2 Satellite radiance

The file *fort.207* is the statistic fit file for radiance data. Its content includes important information about the radiance data analysis.

The first part of the file *fort.207* lists the content that corresponds to those in the file *satinfo*, which is the info file to control the data usage for radiance data.

```
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
   5 amsua_n15           chan=    5 var=   0.300 varch_cld=  0.550 use=
1 ermax=   2.000 b_rad=   10.00 pg_rad=   0.00 icld_det=-2
   6 amsua_n15           chan=    6 var=   0.230 varch_cld=  0.230 use= -
1 ermax=   2.000 b_rad=   10.00 pg_rad=   0.00 icld_det=-2
```

• • • •

This shows there are 2680 channels listed in the *satinfo* file and the 2680 lines following this line include the detailed setups in the *satinfo* file for each channel.

The second part of the file is a list of the coefficients for air mass bias correction, after reading the satbias_angle file:

```
 RADINFO_READ:  read satbias_angle file
 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
   2           amsua_n15   -0.677697    0.382025    1.424922   -0.000061
0.016514    0.000000    0.000000
   3           amsua_n15   -2.631062    0.134578    2.968469   -0.004946
1.213581    0.000000    0.000000
```

• • • •

Each channel has 7 coefficients listed in a line. Therefore, there are 2680 lines of air mass bias correction coefficients for all channels though some of the coefficients are 0.

The 3rd part of the *fort.207* file is similar to other fit files with similar content repeated in 3 sections to give detailed statistic information about the data in stages before the 1st outer loop, between 1st and 2nd outer loop, and after 2nd outer loop. The results before the 1st outer loop are used here as an example to explain the content of the statistic results:

- **Summaries for various statistics as a function of observation type**

```
 sat       type             penalty    nobs   iland isnoice  icoast ireduce   ivarl
nlgross
 metop-a    hirs4         371.46818888    385     143       0      25       0    1452
0
```

```
                              qcpenalty      qc1      qc2      qc3      qc4      qc5      qc6
qc7
                            371.46818888       0        0     1651     3330        0        0
0

  sat       type              penalty     nobs    iland isnoice  icoast ireduce   ivarl
nlgross
 metop-b    hirs4          0.00000000       34       25        0        0        0      139
0
                              qcpenalty      qc1      qc2      qc3      qc4      qc5      qc6
qc7
                              0.00000000       0        0       97      279        0        0
0
```

●●●●[6]

```
rad total   penalty_all=   13986.0933845987129
rad total qcpenalty_all=   13986.0933845987129
rad total failed nonlinqc=                     0
```

The Table 4.6 lists the meaning of each item in the above statistics:

Table 4.6: content of summarizing radiance observation process in fort.207

Name	Explanation
sat	satellite name
type	instrument type
penalty	contribution to cost function from this observation type
nobs	number of good observations used in the assimilation
iland	number of observations over land
isnoice	number of observations over sea ice and snow
icoast	number of observations over coast
ireduce	number of observations that reduce qc bounds in tropics
ivarl	number of observations tossed by gross check
nlgross	number of observation tossed by nonlinear qc
qcpenalty	nonlinear qc penalty from this data type
qc1-7	number of observations whose quality control criteria has been adjusted by each qc method (1-7), details see in the Radiance Chapter of the Advanced User's Guide
rad total penalty all	summary of penalty for all radiance observation types
rad total qcpenalty all	summary of qcpenalty for all radiance observation types
rad total failed nonlinqc	summary of observation tossed by nonlinear qc for all radiance observation types

Note: one radiance observation may include multiple channels, not all channels are used in the analysis.

- **Summaries for various statistics as a function of channel**

```
 1  2     3          4   5     6         7          8         9        10        11
 1  1 amsua_n15   915 247  3.000  0.6623748  1.2673027  0.2414967 2.4209147 2.0627099
 2  2 amsua_n15   902 261  2.000  0.5209886  1.1773765  0.2626527 2.2705374 1.9414234
 3  3 amsua_n15  1114  48  2.000  1.2349467 -1.5090660  0.3421312 1.9416738 1.2218089
```

```
 4   4 amsua_n15 1162    0  0.600 -0.2114506 -0.3195250  0.5286580  0.6271317  0.5396275
 5   5 amsua_n15 1162    3  0.300 -0.0888496 -0.1685199  0.4352839  0.2734131  0.2153040
 6   6 amsua_n15 2093    3 -0.230 -1.8141034 -0.0695503  0.8415303  0.2646033  0.2552992
 7   7 amsua_n15 2342   28  0.250 -0.0830293  0.0453835  0.6141132  0.2522587  0.2481427
 8   8 amsua_n15 2311   59  0.275  0.0201868  0.1758359  0.7782391  0.3173862  0.2642267
 9   9 amsua_n15 2124 246  0.340  0.1262987  0.5008846  1.4907547  0.5544190  0.2376868
10  10 amsua_n15   82 2288 0.400  0.6700157  0.8263504  2.0387089  0.8304076  0.0819868
15  15 amsua_n15  862  300 3.000  0.8244923 -2.2960147  0.3769625  2.7445692  1.5036544
```

● ● ● ●

```
63   4 hirs4_metop-a  11 217 0.400  0.9717384  0.9536444  1.8583033  0.9570578  0.0807581
64   5 hirs4_metop-a  81   4 0.360  0.1655806 -0.2231640  0.3231878  0.3433454  0.2609289
65   6 hirs4_metop-a  25  27 0.460 -0.8415009 -1.1454801  2.6082742  1.1558578  0.1545404
66   7 hirs4_metop-a  20   3 0.570 -0.9563649 -1.1287970  1.4138248  1.1699278  0.3074870
67   8 hirs4_metop-a  23   0 1.000  1.3954294  0.6716651  0.1668694  0.9134028  0.6190078
```

● ● ● ●

The Table 4.7 lists the meaning of each column in above statistics:

Table 4.7 content of fit statistic for each channel in fort.207

Column #	Content
1	series number of the channel in satinfo file
2	channel number for certain radiance observation type
3	radiance observation type (for example: amsua_n15)
4	number of observations (nobs) used in GSI analysis within this channel
5	number of observations (nobs) tossed by gross check within this channel
6	variance for each satellite channel
7	bias (observation-guess before bias correction)
8	bias (observation-guess after bias correction)
9	penalty contribution from this channel
10	sqrt of (observation-guess with bias correction)**2
11	standard deviation

- **Final summary for each observation type**

```
it      satellite instrument # read  # keep  # assim  penalty   qcpnlty    cpen   qccpen
o-g 01 rad  n16      hirs3         0       0        0   0.0000    0.0000  0.0000   0.0000
o-g 01 rad  n17      hirs3         0       0        0   0.0000    0.0000  0.0000   0.0000
o-g 01 rad  metop-a  hirs4     13832    7315      651 371.47    371.47    0.57061  0.57061
o-g 01 rad  n18      hirs4         0       0        0   0.0000    0.0000  0.0000   0.0000
o-g 01 rad  n19      hirs4         0       0        0   0.0000    0.0000  0.0000   0.0000
o-g 01 rad  metop-b  hirs4      2527     646        0   0.0000    0.0000  0.0000   0.0000
o-g 01 rad  g11     goes_img       0       0        0   0.0000    0.0000  0.0000   0.0000
o-g 01 rad  g12     goes_img       0       0        0   0.0000    0.0000  0.0000   0.0000
o-g 01 rad  aqua      airs         0       0        0   0.0000    0.0000  0.0000   0.0000
o-g 01 rad  n15      amsua     43515   30810    12976   8854.4    8854.4  0.68236  0.68236
o-g 01 rad  n18      amsua     30690   25575     8203   3642.7    3642.7  0.44407  0.44407
o-g 01 rad  n19      amsua         0       0        0   0.0000    0.0000  0.0000   0.0000
o-g 01 rad  metop-a  amsua      4890    3878     1466   652.20    652.20  0.44489  0.44489
o-g 01 rad  metop-b  amsua       810     718      294   201.79    201.79  0.68636  0.68636
o-g 01 rad  aqua     amsua         0       0        0   0.0000    0.0000  0.0000   0.0000
o-g 01 rad  n17      amsub         0       0        0   0.0000    0.0000  0.0000   0.0000
```

The following table lists the meaning of each column in the above statistics:

Table 4.8 content of final summary section in fort.207

Name	Explanation
it	stage (o-g 01 rad = before 1st outer loop for radiance data)
satellite	satellite name (n16=NOAA-16)
instrument	instrument name (HIRS-3)
# read	number of data (channels) read in within analysis time window and domain
# keep	number of data (channels) kept after data thinning
# assim	number of data (channels) used in analysis (passed all qc process)
penalty	contribution from this observation type to cost function
qcpnlty	nonlinear qc penalty from this data type
cpen	penalty divided by (the number of data assimilated)
qccpen	qcpnlty divided by (the number of data assimilated)

Similar to other fit files, a comparison between results from different outer loops can give us very useful information on how much impact each channel and data type has in the GSI.

4.6 Convergence Information

There are two ways to check the convergence information for each iteration of the GSI:

1. Standard output file (*stdout*):

The value of the cost function and norm of the gradient for each iteration are listed in the file *stdout*.

The following is an example showing the iterations from the first outer loop:

```
  GLBSOI:  START pcgsoi jiter=           1
  pcgsoi: gnorm(1:2),b=  9.869857497554413276E+05  9.869857497554413276E+05
0.000000000000000000E+00
  Initial cost function =  3.915930707165839704E+04
  Initial gradient norm =  9.934715646436194447E+02
  cost,grad,step,b,step? =   1   0  3.915930707165839704E+04  9.934715646436194447E+02
4.821051367939106942E-03  0.000000000000000000E+00  good
  pcgsoi: gnorm(1:2),b=  4.101618851769856410E+05  4.101618851769853500E+05
4.155702200144395508E-01
  cost,grad,step,b,step? =   1   1  3.440099807266351127E+04  6.404388223530688720E+02
5.837551195665082078E-03  4.155702200144395508E-01  good
  pcgsoi: gnorm(1:2),b=  4.679752995326447999E+05  4.679752995326457312E+05
1.140952673675940110E+00
  cost,grad,step,b,step? =   1   2  3.200665706943234181E+04  6.840872017021256397E+02
3.727016244273905453E-03  1.140952673675940110E+00  good
  pcgsoi: gnorm(1:2),b=  2.579351145588153449E+05  2.579351145588165091E+05
5.511724973869557287E-01
```

● ● ● ●

```
  cost,grad,step,b,step? =   1   9  2.515019683100273687E+04  2.511847728725914237E+02
5.552768045789822915E-03  9.872916643640710088E-01  good
  pcgsoi: gnorm(1:2),b=  5.868194961445817898E+04  5.868194961445837544E+04
9.300748853414525508E-01
  cost,grad,step,b,step? =   1  10  2.479985164931967302E+04  2.422435749704379191E+02
5.481216572289848883E-03  9.300748853414525508E-01  good
```

The following are the iterations from the second outer loop:

```
Initial cost function =  2.792919782749931983E+04
Initial gradient norm =  4.241369976412337337E+02
cost,grad,step,b,step? =   2   0  2.792919782749931983E+04  4.241369976412337337E+02
4.301269527061492466E-03  0.000000000000000000E+00  good
pcgsoi: gnorm(1:2),b=  1.641799966598818428E+05  1.641799966598813771E+05
9.126577097845959274E-01
cost,grad,step,b,step? =   2   1  2.715543302058953486E+04  4.051913087171068923E+02
3.965037770683364597E-03  9.126577097845959274E-01  good
pcgsoi: gnorm(1:2),b=  7.369958699881075881E+04  7.369958699881142820E+04
4.488950450613589660E-01
cost,grad,step,b,step? =   2   2  2.650445313264244396E+04  2.714766785541821719E+02
6.065878225371862040E-03  4.488950450613589660E-01  good
pcgsoi: gnorm(1:2),b=  5.010117245725526300E+04  5.010117245725520479E+04
6.798026216627733875E-01
```

● ● ● ●

```
cost,grad,step,b,step? =   2   9  2.488459155328830457E+04  1.491452772915179139E+02
6.349376870755444636E-03  9.522651754285673675E-01  good
pcgsoi: gnorm(1:2),b=  2.350218829989638834E+04  2.350218829989681763E+04
1.056548139732611746E+00
cost,grad,step,b,step? =   2  10  2.474335402213209454E+04  1.533042344486817683E+02
5.304880218120413757E-03  1.056548139732611746E+00  good
```

We can see clearly the number of outer loops and the inner loops (Minimization iteration). The meaning of the names (bold) used in *stdout* are explained in the following:

`cost`: the values of cost function, (=J)
`grad`: inner product of gradients (norm of the gradient (Y*X))
`step`: stepsize (α)
`b`: parameter to estimate the new search direction

As a quick check, the cost function reduced from `3.915930707165839704E+04` to `2.479985164931967302E+04` in the1st outerloop and reduced from `2.792919782749931983E+04` to `2.474335402213209454E+04` in the 2nd outer loop.

2. Convergence information in file *fort.220*:

In file *fort.220*, users can find more detailed minimization information about each iteration. A detailed description and example are provided in the Advanced User's Guide.

To evaluate the convergence of the iteration, we usually make plots based on the information from *fort.220*, such as the value of the cost function and the norm of the gradient. The following are example plots showing the evolution of the cost function and the norm of gradient in different outer loops:

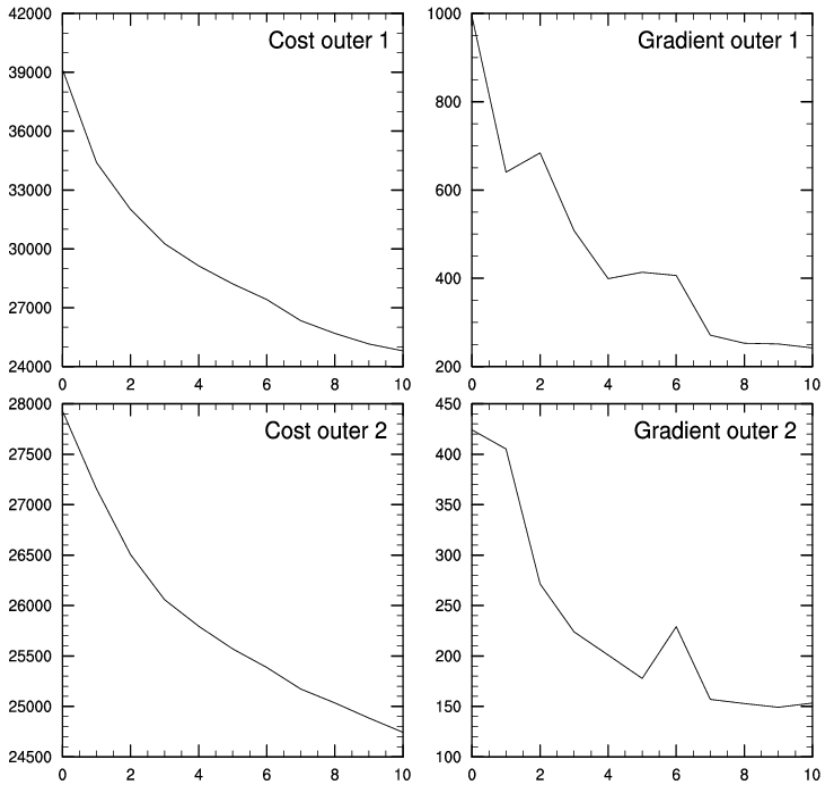

Fig.4.2 Evolution of cost function (left column) and the norm of gradient (right column) in the first outer loop (top raw) and the second outer loop (bottom raw)

Scripts are available in the release code to read convergence information from *fort.220* and produce the above plots. Please see Section A.3 for information on where to locate and how to run these scripts.

4.7 Conventional Observation Errors

Each observation type has its own observation errors. In this section, we introduce several topics related to the conventional observation error processing in GSI. The observation error for satellite radiance and its adjustment is discussed in the Advanced User's Guide.

4.7.1 Getting original observation errors

For the global GSI analysis, when `oberrflg` (a namelist option in section `&obsqc`) is true, observation errors are generated based on an external observation error table according to the types of observations. Otherwise, observation errors are read in from the PrepBUFR file.

For regional GSI runs, GSI forces the use of an external observation error table to get observation errors no matter what the `oberrflg` is set to (`oberrflg` is forced to be true for regional runs in *gsimod.F90*).

The external observation error table file, *errtable*, includes observation errors for all types of conventional observations. It is copied from the ~/*comGSIv3.4_EnKFv1.0/fix* directory by the run script. This release package has three sample external observation error table files, *nam_errtable.r3dv*, *prepobs_errtable.global*, and *rtma*/*new_rtma_nam_errtable.r3dv* in the *./fix* directory. The *nam_errtable.r3dv* is used in the sample run script as a default observation error table. The observation error file is a text file that can be easily edited to tune the error values. The following shows a portion of *nam_errtable.r3dv* for rawinsondes and its description of each column in Table 4.9:

```
Column #     1          2          3          4          5          6
      120 OBSERVATION TYPE
      0.11000E+04 0.12696E+01 0.60737E+00 0.10000E+10 0.76322E+00 0.10000E+10
      0.10500E+04 0.13282E+01 0.66294E+00 0.10000E+10 0.76322E+00 0.10000E+10
      0.10000E+04 0.13932E+01 0.74223E+00 0.10000E+10 0.76322E+00 0.10000E+10
      0.95000E+03 0.14390E+01 0.83688E+00 0.10000E+10 0.79899E+00 0.10000E+10
      0.90000E+03 0.14354E+01 0.94025E+00 0.10000E+10 0.83561E+00 0.10000E+10
      0.85000E+03 0.13669E+01 0.10439E+01 0.10000E+10 0.87224E+00 0.10000E+10


      220 OBSERVATION TYPE
      0.11000E+04 0.10000E+10 0.10000E+10 0.18521E+01 0.10000E+10 0.10000E+10
      0.10500E+04 0.10000E+10 0.10000E+10 0.20636E+01 0.10000E+10 0.10000E+10
      0.10000E+04 0.10000E+10 0.10000E+10 0.22799E+01 0.10000E+10 0.10000E+10
      0.95000E+03 0.10000E+10 0.10000E+10 0.24211E+01 0.10000E+10 0.10000E+10
      0.90000E+03 0.10000E+10 0.10000E+10 0.24934E+01 0.10000E+10 0.10000E+10
      0.85000E+03 0.10000E+10 0.10000E+10 0.25155E+01 0.10000E+10 0.10000E+10
```

Table 4.9 Description of each column in the observation error table file

Column #	1	2	3	4	5	6
Content	Pressure	T	q	UV	Ps	Pw
Unit	hPa	degree C	percent/10	m/s	mb	kg/m^2(or mm)

For each type of observation, the error table has 6 columns and 33 rows (levels). The 1st column prescribes 33 pressure levels, which cover from 1100 hPa to 0 hPa. The columns 2-6 prescribe the observation errors for temperature (T), moisture (q), horizontal wind component (UV), surface pressure (P_s), and the total column precipitable water (P_w). The missing value is `0.10000E+10`.

The observation error table for each observation type starts with the observation type number defined for the PrepBUFR files, such as:

```
120 OBSERVATION TYPE
220 OBSERVATION TYPE
```

The PrepBUFR data type number 100-199 are for temperature (T), moisture (q), and surface pressure (P_s) observations, while number 200-299 are horizontal wind component

(UV) observations. The detailed explanation of each data type number can be found from the following table in the EMC website:

http://www.emc.ncep.noaa.gov/mmb/data_processing/prepbufr.doc/table_2.htm

For more details on PrepBUFR/BUFR, please check the BUFR/PrepBUFR User's Guide, which is freely available at the DTC BUFR/PrepBUFR website:

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

4.7.2 Observation error gross error check within GSI

The gross error check is an important quality check step to exclude questionable observations that degrade the analysis. Users can adjust the threshold of the gross error check for each data type within the *convinfo* file to make the gross error check tighter or looser for a certain data type. For example, the following is a part of *convinfo* without the last five columns:

```
!otype  type  sub iuse twindow numgrp ngroup nmiter gross ermax ermin var_b    var_pg ithin
ps       183    0   -1     3.0      0      0      0   4.0   3.0   1.0   4.0  0.000300     0
ps       187    0    1     3.0      0      0      0   4.0   3.0   1.0   4.0  0.000300     0
t        120    0    1     3.0      0      0      0   8.0   5.6   1.3   8.0  0.000001     0
t        126    0   -1     3.0      0      0      0   8.0   5.6   1.3   8.0  0.001000     0
```

The gross check for each data type is controlled by ***gross, ermax***, and ***ermin***. If an observation has observation error: *obserror*, then a gross check ratio is calculated:

$$ratio = (Observation\text{-}Background)/max(\mathbf{ermin}, min(\mathbf{ermax}, obserror))$$

If *ratio* > ***gross***, then this observation fails the gross check and will not be used in the analysis. The unused observation is indicated as "rejection" in the fit files.

4.8 Background Error Covariance

The GSI package has several files in *~/comGSIv3.4_EnKFv1.0/fix/* to hold the pre-computed background error statistics for different GSI applications with different grid configurations. Within the *./fix* directory subdirectories *Big_Endian* and *Little_Endian* contain the fix files corresponding to each endianness. Since the GSI code has a build-in mechanism to interpolate the input background error matrix to any desired analysis grid, the following two background error files can be used to specify the **B** matrix for any GSI regional application.

- *nam_nmmstat_na.gcv* : contains the regional background error statistics, computed using forecasts from the NCEP's NAM model covering North America. The values of this B matrix cover the northern hemisphere with 93

latitude lines from -2.5 degree to 89.5 degree with 60 vertical sigma levels from 0.9975289 to 0.01364.

- *nam_glb_berror.f77.gcv* : contains the global background errors based on the NCEP's GFS model, a global forecast model. The values of this B matrix covers global with 192 latitude lines from -90 degree to 90 degree and with 42 vertical sigma levels from 0.99597 to 0.013831.

Also included in this release package is the background error matrix for RTMA GSI:

- new_rtma_regional_nmm_berror.f77.gcv

These background error matrix files listed above are Big Endian binary files (therefore located in the *Big_Endian* directory). In the *Little_Endian* directory, *nam_nmmstat_na.gcv* and *nam_glb_berror.f77.gcv* are their Little Endian versions for certain computer platforms that cannot compile GSI with the Big Endian option. In this release version, GSI can be compiled with the Big Endian option with PGI and Intel, but not with gfortran compiler.

4.8.1 Tuning background error covariance through namelist and anavinfo

The final background error covariance matrix used in the GSI analysis are the content from the fixed file "*berror*" multiplied by several factors set by the namelist and the anavinfo.

In GSI namelist, three variables are used for tuning horizontal and vertical impact scales:

`vs`	scale factor for vertical correlation lengths for background error
`hzscl(3)`	scale factor for three scales specified for horizontal smoothing
`hswgt(3)`	weights to apply to each horizontal scales

In the GSI anavinfo files, the column **as/tsfc_sdv** in the *control_vector* section are factors for tuning the variance of each analysis control variable.

These values can be used to tuning the background error covariance used in the GSI analysis. For each background error matrix file, there are recommended values for these parameters listed in table 4.10.

Table 4.10 recommended tuning values for the provided B matrix

	Global	Regional
fixed B matrix	nam_glb_berror.f77.gcv	nam_nmmstat_na.gcv
vs	`0.7`	`1.0`
hzscl	`1.7, 0.8, 0.5`	`0.373,0.746,1.50`
hswgt	`0.45, 0.3, 0.25`	`0.45, 0.3, 0.25`

ss/tsfc_sdv	control_vector:: !var as/tsfc_sdv sf 0.60 vp 0.60 ps 0.75 t 0.75 q 0.75 oz 0.75 sst 1.00 cw 1.00 stl 3.00 sti 3.00	control_vector:: !var as/tsfc_sdv sf 1.00 vp 1.00 ps 0.50 t 0.70 q 0.70 oz 0.50 sst 1.00 cw 1.00 stl 1.00 sti 1.00

4.9 Analysis Increments

Analysis increments are defined as the difference of analysis results minus background. A plot of analysis increments can help users to understand how the analysis procedure modifies the background fields according to observations, background and observation error covariance, and other constraints. You can either calculate *analysis–guess* and plot the difference field or use the tools introduced in Appendix A.4 to make analysis increment figures for different analysis fields.

4.10 Running Time and Memory Usage

In addition to analysis increments, run time and memory usage are other important features of an analysis system, especially for operational code like the GSI.

The GSI standard output file (stdout) gives the GSI start time and end time at the top and the end of the file. For example:

```
* . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * .
     PROGRAM GSI_ANL HAS BEGUN. COMPILED 1999232.55     ORG: NP23
     STARTING DATE-TIME  JUL 16,2015  10:51:06.534  197  THU   2457220
```

● ● ● ●

```
ENDING DATE-TIME    JUL 16,2015  10:53:24.640  197  THU   2457220
     PROGRAM GSI_ANL HAS ENDED.
* . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * .
```

This tells us this case started at `10:51:06.534` and ended at `10:53:24.640`. Meaning GSI used 2 minutes and 18.1 seconds to finish

Following the ending date-time, there is a resource statistics section at the end of the *stdout* file, which gives information about run time and memory usage for the analysis:

```
***************RESOURCE STATISTICS*******************************
The total amount of wall time                      = 138.108109
```

```
The total amount of time in user mode               = 133.766664
The total amount of time in sys mode                = 2.703588
The maximum resident set size (KB)                  = 1930848
Number of page faults without I/O activity          = 111372
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                = 6448
Number of InVoluntary Context Switches              = 1057
*****************END OF RESOURCE STATISTICS**************************
```

Chapter 5: GSI Applications for Regional 3DVAR and Hybrid

In this chapter, the knowledge from the previous chapters will be applied to three regional GSI cases with different data sources. These examples are to give users a clear idea on how to set up GSI with various configurations and properly check the run status and analysis results in order to determine if a particular GSI application was successful. Note the examples here only use the WRF ARW system –WRF NMM runs are similar, but require different background and namelist options.

For illustrations of all the cases, it is assumed that the reader has successfully compiled GSI on a local machine. For regional case studies, users should have the following data available:

1. Background file
   - When using WRF, WPS and real.exe will be run to create a WRF input file: wrfinput_<domain>_<yyyy-mm-dd_hh:mm:ss>
2. Conventional data
   - Real time NAM PREPBUFR data can be obtained from the server: ftp://ftpprd.ncep.noaa.gov/pub/data/nccf/com/nam/prod
     Note: NDAS prepbufr data was chosen to increase the amount of data
3. Radiance data and GPS RO data
   - Real time GDAS BUFR files can be obtained from the following server: ftp://ftpprd.ncep.noaa.gov/pub/data/nccf/com/gfs/prod
     Note: GDAS data was chosen to get better coverage for radiance and GPS RO

The following cases will give users an example of a successful GSI run with various data sources. Users are welcome to download these example data from the GSI users' webpage (online case for release version 3.4) or create a new background and get the observation data from the above server. The background and observations used in this case study are as follows:

1. Background files: wrfinput_d01_2014-06-17_00:00:00
   - The horizontal grid spacing is 30-km with 51 vertical sigma levels

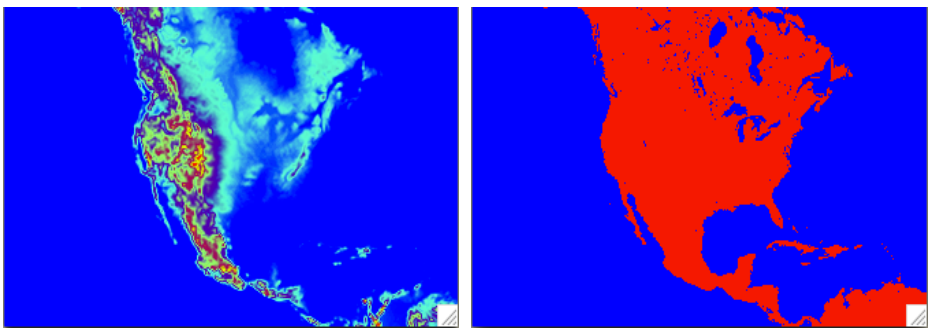

Figure 5.1: The terrain (left) and land mask (right) of the background used in this case study

2. Conventional data: NAM PrepBUFR data from 00UTC 17 June 2014.
   - File: *nam.t00z.prepbufr.tm00.nr*

3. Radiance and GPS RO data: GDAS PREPBUFR data from 00 UTC 17 June 2014
   - Files: *gdas.t00z.1bamua.tm00.bufr_d*
     gdas.t00z.1bhrs4.tm00.bufr_d
     gdas.t00z.gpsro.tm00.bufr_d

This case study was run on a Linux cluster. Starting from version 3.2, the BUFR/PrepBUFR files do not need to be byte-swapped to little endian format. BUFRLIB can automatically handle byte order issues.

Assume the background file is located at:

data/2014061700/arw

all the observations are located at:

data/2014061700/obs

and the GSI release version 3.4 is located at

code/comGSIv3.4_EnKFv1.0

5.1. Assimilating Conventional Observations with Regional GSI:

5.1.1: Run script

With GSI successfully compiled and background and observational data acquired, move to the *./run* directory under *./comGSIv3.4_EnKFv1.0* to run the GSI using the sample script *run_gsi_regional.ksh*. The *run_gsi.ksh* script must be modified in several places before running:

- Set up batch queuing system
  To run GSI with multi-processors, a job queuing head has to be added at the beginning of the *run_gsi_regional.ksh* script. The set up of the job queue is dependent on the machine and the job control system. More examples of the setup are described in section 3.2.2.1. The following example is set up to run on a Linux cluster supercomputer with LSF. The job head is as follows:

```
#BSUB -P ?????????             # project code
#BSUB -W 00:20                 # 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
```

In order to find out how to set up the job head, a good method is to use an existing MPI job script and copy the job head over.

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

```
GSIPROC=4
ARCH='LINUX_PBS'
```

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

  Set up analysis time:

```
ANAL_TIME=2014061700
```

  Set up a working directory, which will hold all the analysis results. This directory must have correct write permissions, as well as enough space to hold the output.

```
WORK_ROOT=gsiprd_${ANAL_TIME}_prepbufr
```

  Set path to the background file:

```
BK_FILE=data/2014061700/arw/wrfinput_d01_2014-06-17_00:00:00
```

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

```
OBS_ROOT=data/2014061700/obs
PREPBUFR=${OBS_ROOT}/nam.t00z.prepbufr.tm00.nr
```

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

```
CRTM_ROOT=CRTM_REL-2.1.3
FIX_ROOT=code/comGSIv3.4_EnKFv1.0/fix
GSI_EXE=code/comGSIv3.4_EnKFv1.0/run/gsi.exe
```

- 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.

5.1.2: Run GSI and check the run status

Once the run script is set up properly for the case and machine, and the `anavinfo` file has been updated with the same number of vertical levels as the background (please see section 3.1 for more details), GSI can be run through the run script. On our test machine, the GSI run is submitted as follows:

```
$ 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=gsiprd_${ANAL_TIME}_prepbufr
```

Go to directory */scratch1* to check the GSI run directory.

A directory named *gsiprd_2014061700_prepbufr* should have been created. This directory is the run directory for this GSI case study. While GSI is still running, the contents of this directory should include files such as:

```
imgr_g12.TauCoeff.bin          ssmi_f15.SpcCoeff.bin
imgr_g13.SpcCoeff.bin          ssmi_f15.TauCoeff.bin
imgr_g13.TauCoeff.bin          ssmis_f16.SpcCoeff.bin
```

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

`stdout:`	standard out file
`wrf_inout:`	background file
`gsiparm.anl:`	GSI namelist
`prepbufr:`	PrepBUFR file for conventional observation
`convinfo:`	data usage control for conventional data
`berror_stats:`	background error file
`errtable:`	observation error file

The presence of these files indicates that the GSI run scripts have successfully set up a run environment for GSI and the GSI executable is running. While GSI is still running, checking the content of the standard output file (stdout) can monitor the status of the GSI analysis:

```
$ tail -f stdout

GLBSOI:  START pcgsoi jiter=           1
pcgsoi: gnorm(1:2),b=  1.13152054892331 3618E+05  1.13152054892331 3618E+05
0.00000000000000000E+00
Initial cost function =  3.24958551456716 5956E+04
Initial gradient norm =  3.36380818258609 0538E+02
cost,grad,step,b,step? =   1   0  3.24958551456716 5956E+04  3.36380818258609 0538E+02
2.54855354723164 0140E-02  0.00000000000000000E+00  good
pcgsoi: gnorm(1:2),b=  9.71189065354544 8731E+04  9.71189065354545 4552E+04
8.58304399578664 4453E-01
cost,grad,step,b,step? =   1   1  2.96121144369476 6422E+04  3.11639064520888 4962E+02
2.51452180550157 2851E-02  8.58304399578664 4453E-01  good
pcgsoi: gnorm(1:2),b=  8.59709165907255 6649E+04  8.59709165907252 1725E+04
8.85212979198241 3787E-01
cost,grad,step,b,step? =   1   2  2.71700383548489 6263E+04  2.93207974978044 4899E+02
1.03657516430916 4361E-02  8.85212979198241 3787E-01  good
pcgsoi: gnorm(1:2),b=  5.68351587282540 5677E+04  5.68351587282542 6050E+04
6.61097508112244 1033E-01
cost,grad,step,b,step? =   1   3  2.62788851849405 7644E+04  2.38401255718702 2310E+02
1.94840928411394 5122E-02  6.61097508112244 1033E-01  good
pcgsoi: gnorm(1:2),b=  3.26652742955937 3923E+04  3.26652742955935 5369E+04
5.74737099825409 6586E-01
```

The above output shows that GSI is in the inner iteration stage. It may take several minutes to finish the GSI run. Once GSI has finished running, the number of files in the directory will be greatly reduced from those during the run stage. This is because the run script was set to clean the run directory after a successful run. The important analysis result files and configuration files will remain in the run directory. Please check Section 3.3 for more details on GSI run results. Upon successful completion of GSI, the run directory looks as follows:

```
anavinfo                  fort.202  fort.214  fort.228              satbias_angle
berror_stats              fort.203  fort.215  fort.229              satbias_in
convinfo                  fort.204  fort.217  fort.230              satbias_out
diag_conv_anl.2014061700  fort.205  fort.218  gsi.exe               satinfo
diag_conv_ges.2014061700  fort.206  fort.219  gsiparm.an            stdout
errtable                  fort.207  fort.220  l2rwbufr              stdout.anl.2014061700
fit_p1.2014061700         fort.208  fort.221  list_run_directory    wrfanl.2014061700
fit_q1.2014061700         fort.209  fort.223  ozinfo                wrf_inout
fit_rad1.2014061700       fort.210  fort.224  pcpbias_out
fit_t1.2014061700         fort.211  fort.225  pcpinfo
fit_w1.2014061700         fort.212  fort.226  prepbufr
fort.201                  fort.213  fort.227  prepobs_prep.bufrtable
```

5.1.3: Check for successful GSI completion

It is important to always check for successful completion of the GSI analysis. But, completion of the GSI run without crashing does not guarantee a successful analysis. First, check the *stdout* file in the run directory to make sure GSI completed each step without any obvious problems. The following are several important steps to check:

1. Read in the anavinfo and namelist
   The following lines show GSI started normally and has read in the anavinfo and namelist:

```
 gsi_metguess_mod*init_:  2D-MET STATE VARIABLES:
 ps
 z
 gsi_metguess_mod*init_:  3D-MET STATE VARIABLES:
 u
 v
  ... ...
 state_vectors*init_anasv:  2D-STATE VARIABLES
 ps
 sst
 state_vectors*init_anasv:  3D-STATE VARIABLES u
 v
  ... ...
 GSI_4DVAR:  nobs_bins =             1
 SETUP_4DVAR: l4dvar=  F
 SETUP_4DVAR: l4densvar=  F
 SETUP_4DVAR: winlen=    3.000000000000000
 SETUP_4DVAR: winoff=    3.000000000000000
 SETUP_4DVAR: hr_obsbin=    3.000000000000000
 SETUP_4DVAR: nobs_bins=            1


  ... ...
&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 the range of the max and min values will indicate if certain background fields are normal.

```
 dh1  =             3
  iy,m,d,h,m,s=         2014           6          17           0
            0           0
```

```
 dh1  =             3
rmse_var = SMOIS
ndim1 =             3
ordering = XYZ
staggering =  N/A
start_index =             1            1            1            0
end_index =           332          215            4            0
WrfType =           104
ierr  =             0

.................

 rmse_var = U ndim1=            3
 WrfType =           104  WRF_REAL=          104 ierr  =             0
 ordering = XYZ staggering =  N/A
 start_index =             1            1            1            0
 end_index =           333          215           50            0
 k,max,min,mid U=            1    18.50961      -17.84097      -0.8667576
 k,max,min,mid U=            2    18.68178      -18.39229      -0.8647658
 k,max,min,mid U=            3    19.28049      -19.42709      -0.8610985
 k,max,min,mid U=            4    19.60607      -21.29182      -0.8547171
 k,max,min,mid U=            5    21.58153      -24.50086      -0.8405453
```

3. Read in observational data
   Skipping through a majority of the content towards the middle of the *stdout* file, the following lines will appear:

```
  OBS_PARA: ps                      1429      3190      4655      6774
  OBS_PARA: t                       2564      5200      7057     11128
  OBS_PARA: q                       2346      4626      6148      8128
  OBS_PARA: pw                        65        80        63        49
  OBS_PARA: uv                      3358      6453      8091     11998
```

   This table is an important step to check if the observations have been read in, which types of observations have been read in, and the distribution of observations in each sub domain. At this point, GSI has read in all the data needed for the analysis. Following this table is the inner iteration information.

4. Inner iteration
   The inner iteration step in the stdout file will look as follows:

```
GLBSOI:  START pcgsoi jiter=           1
pcgsoi: gnorm(1:2),b=  1.13152054892331 3618E+05  1.13152054892331 3618E+05
0.00000000000000000E+00
Initial cost function =  3.24958551456716 5956E+04
Initial gradient norm =  3.36380818258609 0538E+02
cost,grad,step,b,step? =   1   0  3.24958551456716 5956E+04  3.36380818258609 0538E+02
2.54855354723164 0140E-02  0.00000000000000000E+00  good
pcgsoi: gnorm(1:2),b=  9.71189065354544 8731E+04  9.71189065354545 4552E+04
8.58304399578664 4453E-01
```

```
cost,grad,step,b,step? =   1   1  2.96121144369476642E+04  3.11639064520888496E+02
2.51452180550157285E-02  8.58304399578664445E-01  good
pcgsoi: gnorm(1:2),b=  8.59709165907255665E+04  8.59709165907252172E+04
8.85212979198241378E-01
cost,grad,step,b,step? =   1   2  2.71700383548489626E+04  2.93207974978044489E+02
1.03657516430916436E-02  8.85212979198241378E-01  good
pcgsoi: gnorm(1:2),b=  5.68351587282540568E+04  5.68351587282542605E+04
6.61097508112244103E-01
... ...
```

Following the namelist set up, similar information will be repeated for each inner loop. In this case, 2 outer loops with 50 inner loops in each outer loop have been set. The last iteration looks like:

```
cost,grad,step,b,step? =   2  43  2.28306638506101844E+04  7.27540284515745036E-03
2.09708965083010741E-02  1.41131849208848048E+00  good
pcgsoi: gnorm(1:2),b=  1.69386888903473762E-05  1.69387151374851113E-05
3.20012080493910833E-01
cost,grad,step,b,step? =   2  44  2.28306638495001971E+04  4.11566384548528626E-03
4.36799433583779026E-02  3.20012080493910833E-01  good
pcgsoi: gnorm(1:2),b=  1.58101615515517240E-05  1.58101157774135482E-05
9.33373053899598348E-01
cost,grad,step,b,step? =   2  45  2.28306638487603050E+04  3.97619938528637196E-03
2.25313861190973123E-02  9.33373053899598348E-01  good
pcgsoi: gnorm(1:2),b=  7.56593578602322126E-06  7.56599549180765260E-06
4.78552699612678600E-01
cost,grad,step,b,step? =   2  46  2.28306638484040886E+04  2.75062461743205179E-03
5.60348131013475095E-02  4.78552699612678600E-01  good
 PCGSOI: WARNING **** Stopping inner iteration ***
 gnorm  0.66865208883943889E-10 less than  0.10000000000000004E-09
 update_guess: successfully complete
```

Clearly, at the 46th iteration GSI met the stop threshold before getting to the maximum iteration number (50). As a quick check of the iteration: the J value should descend with each iteration. Here, J has a value of `3.24958551456716596E+04` at the beginning and a value of `2.28306638484040886E+04` at the final iteration. This means the value has reduced by about one third, which is an expected reduction.

5. Write out analysis results
   The final step of the GSI analysis procedure looks very similar to the portion where the background fields were read in:

```
... ...

max,min psfc=    102799.9        66793.78
max,min MU=    2799.898       -1195.195
rmse_var=MU
ordering=XY
WrfType,WRF_REAL=          104          104
ndim1=            2
```

```
staggering= N/A
start_index=            1           1           1           0
end_index1=           332         215          50           0
k,max,min,mid T=            1    321.6157        270.8151        309.3634
k,max,min,mid T=            2    321.7112        270.9660        309.4070
k,max,min,mid T=            3    321.4324        271.2166        309.3831
k,max,min,mid T=            4    321.2418        271.6100        309.3864
k,max,min,mid T=            5    321.6863        272.2831        309.3308

... ...
```

6. As an indication that GSI has successfully run, several lines will appear at the bottom of the file:

```
ENDING DATE-TIME    JUL 19,2015  11:56:58.853  200  SUN   2457223
          PROGRAM GSI_ANL HAS ENDED.
  * . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * .
```

After carefully investigating each portion of the stdout file, it can be concluded that GSI successfully ran through every step and there were no run issues. A more complete description of the *stdout* file can be found in Section 4.1. However, it cannot be concluded that GSI did a successful analysis until more diagnosis has been completed.

5.1.4: Diagnose GSI analysis results

5.1.4.1: Check analysis fit to observations

The analysis uses observations to correct the background fields to fit the observations closer under certain constraints. The easiest way to confirm the GSI analysis results fit the observations better than the background is to check a set of files with names *fort.2??,* where ?? is a number from 01 to 19. In the run scripts, several fort files have also been renamed as *fit_t1 (q1, p1, rad1, w1).YYYYMMDDHH*. Please check Section 4.5.1 for a detailed explanation of the fit files. Here illustrates how to use these fit files.

- *fit_t1.2014061700 (fort.203)*
  This file shows how the background and analysis fields fit to temperature observations. The contents of this file show five data types were used in the analysis: 120, 130, 132, 180 and 182. Also included are the number of observations, bias and rms of observation minus background (o-g 01) or analysis (o-g 03) on each level for the three data types. The following is a part of the file, only showing data types 120 and 180:

```
                              ptop  1000.0  900.0  800.0  600.0  400.0  300.0  250.0  200.0
150.0  100.0   50.0     0.0
     it     obs    type styp  pbot  1200.0 1000.0  900.0  800.0  600.0  400.0  300.0  250.0
200.0  150.0  100.0  2000.0
-------------------------------------------------------------------------------------------
---------------------------------
 o-g 01       t     120 0000 count     107    350    357    866   1153    719    252    450
551    884    745    7188
 o-g 01       t     120 0000  bias    0.80   0.32  -0.10  -0.12  -0.15  -0.20  -0.24  -0.60
-0.22   0.15  -0.10   -0.07
 o-g 01       t     120 0000   rms    2.06   1.55   0.83   0.77   0.69   0.66   0.73   1.20
1.44   1.65   1.65    1.23
o-g 01       t     180 0000 count    1683     50      0      0      0      0      0      0
0      0      0    1733
 o-g 01       t     180 0000  bias    0.69   2.04   0.00   0.00   0.00   0.00   0.00   0.00
0.00   0.00   0.00    0.73
 o-g 01       t     180 0000   rms    1.99   4.50   0.00   0.00   0.00   0.00   0.00   0.00
0.00   0.00   0.00    2.11
o-g 01             all      count    1792    405    358    871   1172    725    325    800
651    884    745    9482
 o-g 01             all       bias    0.69   0.53  -0.10  -0.12  -0.15  -0.19  -0.09  -0.50
-0.04   0.15  -0.10    0.08
 o-g 01             all        rms    1.99   2.14   0.83   0.77   0.69   0.67   0.84   1.32
1.58   1.65   1.65    1.45
```

```
-------------------------------------------------------------------------------------------
---------------------------------
o-g 03       t     120 0000 count     107    350    357    866   1153    719    252    450
551    884    745    7188
 o-g 03       t     120 0000  bias    0.58   0.29  -0.04  -0.02  -0.04  -0.02   0.01  -0.16
-0.04   0.06   0.04    0.01
 o-g 03       t     120 0000   rms    1.72   1.35   0.70   0.61   0.49   0.43   0.50   0.79
1.14   1.40   1.59    1.05
o-g 03       t     180 0000 count    1683     51      0      0      0      0      0      0
0      0      0    1734
 o-g 03       t     180 0000  bias    0.19   0.76   0.00   0.00   0.00   0.00   0.00   0.00
0.00   0.00   0.00    0.21
 o-g 03       t     180 0000   rms   1.71   2.85   0.00   0.00   0.00   0.00   0.00   0.00
0.00   0.00   0.00    1.76
o-g 03             all      count    1792    406    358    871   1172    725    325    800
651    884    745    9483
 o-g 03             all       bias    0.21   0.34  -0.04  -0.02  -0.04  -0.02   0.04  -0.13
0.06   0.06   0.04    0.05
 o-g 03             all        rms    1.71   1.61   0.69   0.61   0.49   0.43   0.61   0.94
1.26   1.40   1.59    1.22
```

For example: data type 120 has 1153 observations in layer 400.0-600.0 hPa, a bias of -0.15, and a rms of 0.69. The last column shows the statistics for the whole atmosphere. There are several summary lines for all data types, which is indicated by '`all`' in the data types column. For summary O-B (which is "`o-g 01`" in the file), we have 9482 observations total, a bias of 0.08, and a rms of 1.45.

Skipping ahead in the fort file, "`o-g 03`" columns (under 'it') show the observation minus analysis (O-A) information. Under the summary ('`all`') lines, it can be seen that

there were 9483 total observations, a bias of 0.05, and a rms of 1.22. This shows that from the background to the analysis, one more observation data is being used, the bias reduced from 0.08 to 0.05, and the rms reduced from 1.45 to 1.22. This is about a 16% reduction, which is a reasonable value for large-scale analysis.

- *fit_w1.2014061700 (fort.202)*
  This file demonstrates how the background and analysis fields fit to wind observations. This file (as well as *fit_q1*) are formatted the same as the *fort.203*. Therefore, only the summary lines will be shown for O-B and O-A to gain a quick view of the fitting:

```
                              ptop  1000.0  900.0  800.0  600.0  400.0  300.0  250.0  200.0
150.0  100.0   50.0     0.0
     it     obs    type styp  pbot  1200.0 1000.0  900.0  800.0  600.0  400.0  300.0  250.0
200.0  150.0  100.0  2000.0
 ------------------------------------------------------------------------------------------
------------------------------
 o-g 01             all      count    1597   1703   1839   2930   1213    828    290    687
533    694    798   14513
 o-g 01             all       bias    0.27   0.84   0.68   0.61   0.56   0.45   0.67   0.91
0.48   0.83   1.21    0.64
 o-g 01             all        rms    2.50   2.65   2.52   3.11   4.02   3.98   4.37   4.31
5.32   5.41   4.77    3.59
o-g 03             all      count    1608   1695   1843   2931   1212    828    290    687
533    694    798   14520
 o-g 03             all       bias    0.23   0.42   0.26   0.30   0.37   0.33   0.22   0.37
0.32   0.67   1.22    0.39
 o-g 03             all        rms    2.27   2.16   1.94   2.23   2.74   2.82   3.64   3.31
4.22   4.43   4.41    2.90
```

  O-B: 14513 observations in total, bias is 0.64 and rms is 3.59
  O-A: 14520 observations in total, bias is 0.39 and rms is 2.90
  The total bias was reduced from 0.64 to 0.39 and the rms reduced from 3.59 to 2.90 (~20% reduction).

- *fit_q1.2014061700 (fort.204)*
  This file demonstrates how the background and analysis fields fit to moisture observations (relative humidity). The summary lines for O-B and O-A are as follows:

```
                              ptop  1000.0  950.0  900.0  850.0  800.0  700.0  600.0  500.0
400.0  300.0    0.0     0.0
     it     obs    type styp  pbot  1200.0 1000.0  950.0  900.0  850.0  800.0  700.0  600.0
500.0  400.0  300.0  2000.0
 ------------------------------------------------------------------------------------------
------------------------------
 o-g 01             all      count     543    186    182    211    146    457    406    520
621    623      0    3895
 o-g 01             all       bias    1.17  -3.68  -2.47  -1.30  -3.55   0.19   0.64  -1.80
-4.28  -5.55   0.00   -2.05
 o-g 01             all        rms    9.09  10.63   9.03   9.34  12.73  12.30  14.53  15.27
16.45  16.01   0.00   13.66
```

```
 o-g 03             all      count     543    186    182    211    146    457    406    520
621    623      0    3895
 o-g 03             all       bias   -0.39  -0.88  -0.68   0.45  -0.51   0.06   0.13  -0.10
-0.70  -1.90   0.00   -0.53
 o-g 03             all        rms    5.48   5.19   4.37   5.73   8.13   9.31  12.19  13.82
13.01  12.36   0.00   10.64
```

O-B: 3895 observations in total and bias is -2.05 and rms is 13.66

O-A: 3895 observations in total and bias is -0.53 and rms is 10.64

The total bias and rms were reduced.

- *fit_p1.2014061700 (fort.201)*

  This file demonstrates how the background and analysis fields fit to surface pressure observations. Because the surface pressure is two-dimensional field, the table is formatted different than the three-dimensional fields shown above. Once again, the summary lines will be shown for O-B and O-A to gain a quick view of the fitting:

```
    it     obs    type stype    count      bias       rms      cpen     qcpen
 o-g 01             all          13890    0.1912    0.7931    0.4105    0.3835
 o-g 03             all          13916    0.0403    0.6764    0.2921    0.2731
```

O-B: 13890 observations in total and bias is 0.1912 and rms is 0.7931

O-A: 13916 observations in total and bias is 0.0403 and rms is 0.6764

Both the total bias and rms were reduced.

These statistics show that the analysis results fit to the observations closer than the background, which is what the analysis is supposed to do. How close the analysis fit to the observations is based on the ratio of background error variance and observation error.

5.1.4.2: Check the minimization

In addition to the minimization information in the *stdout* file, GSI writes more detailed information into a file called *fort.220*. The content of *fort.220* is explained in the Advanced GSI User's Guide. Below is an example of a quick check of the trend of the cost function and norm of gradient. The value should get smaller with each iteration step.

In the run directory, the cost function and norm of the gradient information can be dumped into an output file by using the command:

```
$ grep 'cost,grad,step,b' fort.220 | sed -e 's/cost,grad,step,b,step? =   //g' | sed -e 's/good//g' > cost_gradient.txt
```

The file *cost_gradient.txt* includes 6 columns, however only the first 4 columns are shown below. The first 5 and last 5 lines read are:

```
1   0  3.249585514567165956E+04  3.363808182586090538E+02
1   1  2.961211443694766422E+04  3.116390645208884962E+02
1   2  2.717003835484896263E+04  2.932079749780444899E+02
1   3  2.627888518494057644E+04  2.384012557187022310E+02
1   4  2.517150367563836699E+04  1.807353709034115354E+02
        … …
2  42  2.283066385194308168E+04  6.124134715091851880E-03
2  43  2.283066385061018445E+04  7.275402845157450359E-03
2  44  2.283066384950019710E+04  4.115663845485286257E-03
2  45  2.283066384876030497E+04  3.976199385286371957E-03
2  46  2.283066384840408864E+04  2.750624617432051791E-03
```

The first column is the outer loop number and the second column is the inner iteration number. The third column is the cost function, and the forth column is the norm of gradient. It can be seen that both the cost function and norm of gradient are descending.

To get a complete picture of the minimization process, the cost function and norm of gradient can be plotted using a provided NCL script located under:

./util/Analysis_Utilities/plot_ncl/GSI_cost_gradient.ncl.

the plot is shown as Fig.5.2:

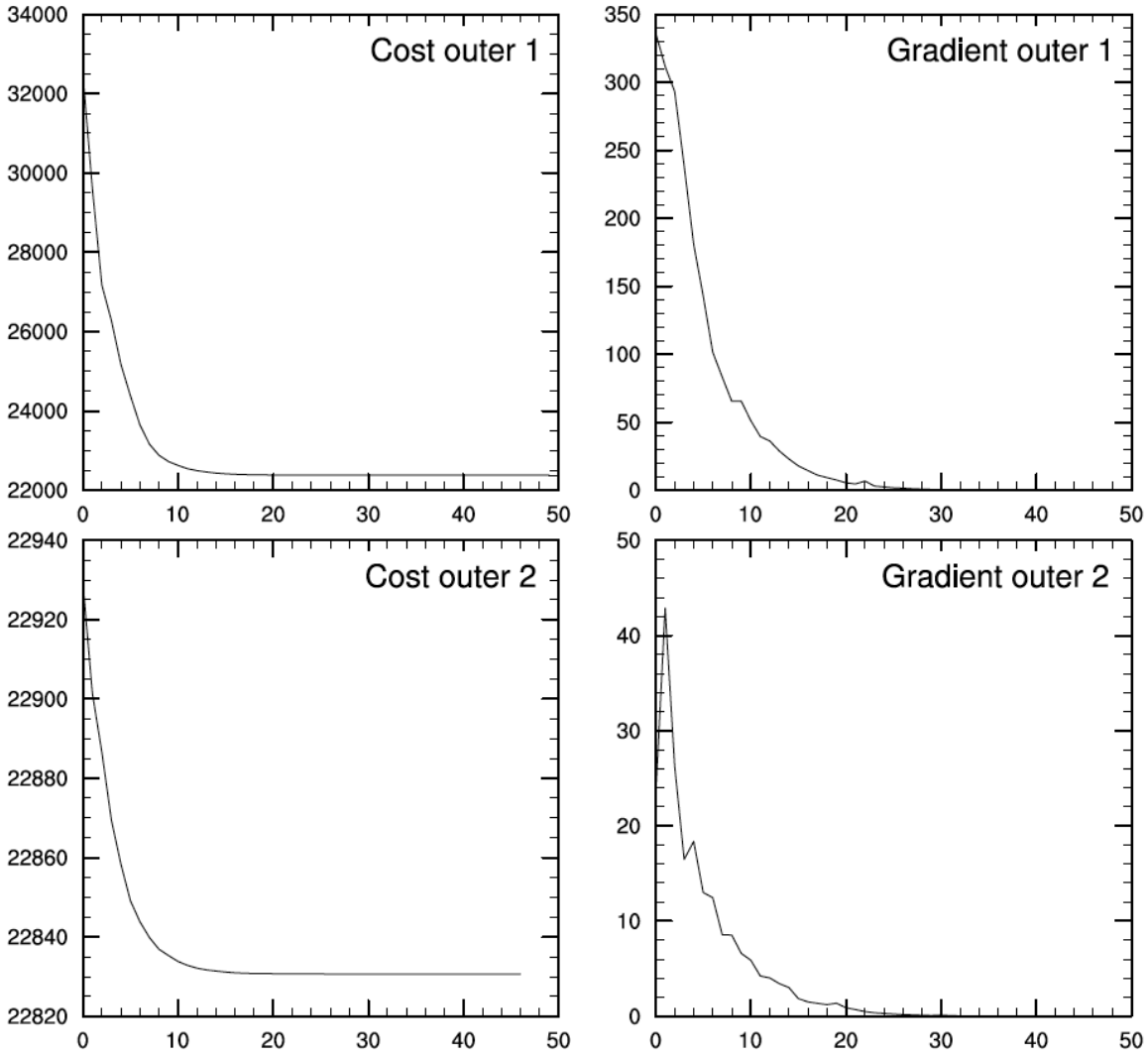

Figure 5.2: Cost function and norm of gradient change with iteration steps

The above plots demonstrate that both the cost function and norm of gradient descend very fast in the first 10 iterations in both outer loops and drop very slowly after the 10th iteration.

5.1.4.3: Check the analysis increment

The analysis increment gives us an idea where and how much the background fields have been modified by the observations through analysis. Another useful graphics tool that can be used to look at the analysis increment is located under:

./util/Analysis_Utilities/plot_ncl/Analysis_increment.ncl.

The graphic below shows the analysis increment at the 15th level. Notice that the scales are different for each of the plots.

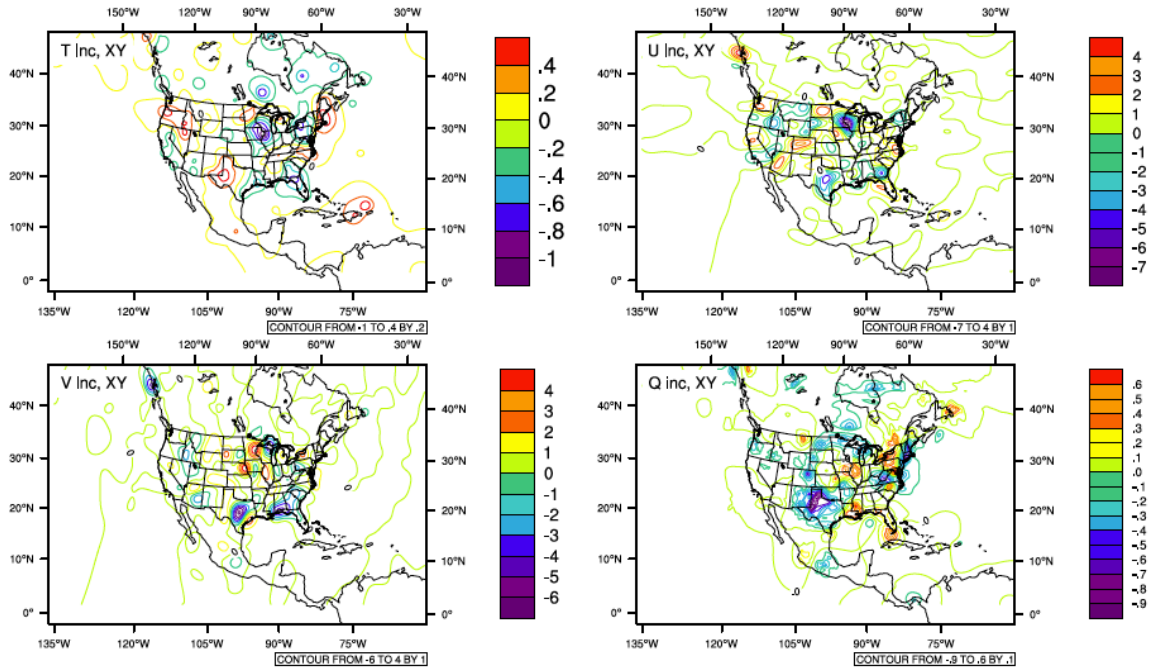

Figure 5.3: Analysis increment at the 15th level

It can be clearly seen that the conventional observations are mostly located in the U.S. CONUS domain and the data availability over the ocean is very sparse.

5.2. Assimilating Radiance Data with Regional GSI

5.2.1: Run script

Adding radiance data into the GSI analysis is very straightforward after a successful run of GSI with conventional data. The same run script from the above section can be used to run GSI with radiance with or without PrepBUFR data. The key step to adding the radiance data is linking the radiance BUFR data files to the GSI run directory with the names listed in the `&OBS_INPUT` section of the GSI namelist. The following example adds the two radiance BUFR files:

AMSU-A: *gdas1.t00z.1bamua.tm00.bufr_d*

HIRS4: *gdas1.t00z.1bhrs4.tm00.bufr_d*

The location of these radiance BUFR files has been previously saved to the scripts variable `OBS_ROOT`, therefore the following three lines can be inserted below the link to the PrepBUFR data in the script *run_gsi_regional.ksh:*

```
ln -s ${OBS_ROOT}/gdas1.t00z.1bamua.tm00.bufr_d amsuabufr
ln -s ${OBS_ROOT}/gdas1.t00z.1bhrs4.tm00.bufr_d hirs4bufr
```

If it is desired to run radiance data in addition to the conventional PrepBUFR data, the following link to the PrepBUFR should be kept as is:

```
ln -s ${PREPBUFR} ./prepbufr
```

Alternatively to analyze radiance data without conventional PrepBUFR data, this line can be commented out in the script *run_gsi.ksh*:

```
## ln -s ${PREPBUFR} ./prepbufr
```

In the following example, the case study will include both radiance and conventional observations.

In order to link the correct name for the radiance BUFR file, the namelist section `&OBS_INPUT` should be referenced. This section has a list of data types and BUFR file names that can be used in GSI. The 1st column '`dfile`' is the file name recognized by GSI. The 2nd column '`dtype`' and 3rd column '`dplat`' are the data type and data platform that are included in the file listed in '`dfile`', respectively. For example, the following line tells us the AMSU-A observation from NOAA-15 should be read from a BUFR file named as '*amsuabufr'*:

```
amsuabufr      amsua       n15       amsua_n15           10.0     2     0
```

With radiance data assimilation, two important setups, data thinning and bias correction, need to be checked carefully. The following is a brief description of these two setups:

- **Radiance data thinning**

  The radiance data thinning is setup in the namelist section `&OBS_INPUT`. The following is a part of namelist in that section:

```
   dmesh(1)=120.0,dmesh(2)=60.0,dmesh(3)=30,time_window_max=1.5,ext_sonde=.true.
!  dfile          dtype     dplat     dsis              dval    dthin  dsfcalc
    amsuabufr      amsua       n15       amsua_n15          10.0     2       0
```

The first line of `&OBS_INPUT` lists multiple mesh grids as elements of the array `dmesh` (three mesh grids in the aboe example). For the line specifying a data type, the 2nd last element of that line is to specify the choice of `dthin`. This selects the mesh grid to be used for thinning. It can be seen that the data thinning option for NOAA-15 AMSU-A observations is 60 km because the value of `dthin is 2,` corresponding to dmesh(2)=60 km. For more information about radiance data thinning, please refer to the Advanced GSI User's Guide.

- **Radiance data bias correction**

  The radiance data bias correction is very important for a successful radiance data analysis. In the sample run scripts, there are two files related to bias correction:

```
SATANGL=${FIX_ROOT}/global_satangbias.txt
cp ${FIX_ROOT}/sample.satbias ./satbias_in
```

  The first file (*global_satangbias.txt*) provides GSI the coefficients for angle bias correction. The angle bias coefficients are calculated off line outside of GSI. The second file (*sample.satbias*) provides GSI the coefficients for mass bias correction. They are usually calculated from within GSI in the previous cycle. In the released version 3.4, most of the mass bias correction values in *sample.satbias* are 0. This means there was not a good estimate base for mass bias correction in this case. The angle bias file *global_satangbias.txt* is also out of date. These two files are provided in *./fix* as an example of the bias correction coefficients. For the best results, it will be necessary for the user to generate their own bias files. The details of the radiance data bias correction are discussed in the Advanced GSI User's Guide.

  For this case, the GDAS bias correction files were downloaded and saved in the observation directory. The run script should contain two lines to link the bias correction coefficient files. In order to do this, change the following lines in run script:

```
SATANGL=${FIX_ROOT}/global_satangbias.txt
cp ${FIX_ROOT}/sample.satbias ./satbias_in
```

  The first line sets the path to the angle bias coefficient file, and the second copies the mass bias coefficient file into the working directory. Once these links are set, we are ready to run the case.

  Also in this release, the angle bias coefficient can also be calculated inside the GSI.

5.2.2: Run GSI and check run status

The process for running GSI is the same as described in section 5.1.2. Once *run_gsi_regional.ksh* has been submitted, move into the run directory to check the GSI analysis results. For our current case, the run directory will look almost as it did for the

conventional data case, the exception being the two links to the radiance BUFR files and new diag files for the radiance data types used. Following the same steps as in section 5.1.2, check the *stdout* file to see if GSI has run through each part of the analysis process successfully. In addition to the information outlined for the conventional run, the radiance BUFR files should have been read in and distributed to each sub domain:

```
OBS_PARA: ps                        1429      3190      4655      6774
OBS_PARA: t                         2564      5200      7057     11128
OBS_PARA: q                         2346      4626      6148      8128
OBS_PARA: pw                          65        80        63        49
OBS_PARA: uv                        3358      6453      8091     11998
OBS_PARA: hirs4     metop-a            0         2      1601      2425
OBS_PARA: hirs4     n19              496      1347         0         0
OBS_PARA: hirs4     metop-b            0         0       154       737
OBS_PARA: amsua     n15             1921      2362       996       391
OBS_PARA: amsua     n18             2697      2789       262        36
OBS_PARA: amsua     n19              326      1137         0         0
OBS_PARA: amsua     metop-a            0         0      1321      1967
OBS_PARA: amsua     metop-b            0         0       386      2238
```

When comparing this output to the content in step 3 of section 5.1.3, it can be seen that there are 8 new radiance data types that have been read in: HIRS4 from METOP-A, METOP-B and NOAA-19, AMSU-A from NOAA-15, NOAA-18, NOAA-19, METOP-A and METOP-B. The table above shows that most of the radiance data read in this case are AMSU-A from NOAA satellite.

5.2.3: Diagnose GSI analysis results

5.2.3.1: Check file *fort.207*

The file *fort.207* contains the statistics for the radiance data, similar to file *fort.203* for temperature. This file contains important details about the radiance data analysis. Section 4.5.2 explains this file in detail. Below are some values from the file *fort.207* to give a quick look at the radiance assimilation for this case study.

The *fort.207* file contains the following lines:

For O-B, the stage before the first outer loop:

```
   it satellite instrument # read  # keep # assim penalty  qcpnlty     cpen   qccpen
o-g 01 rad  n15    amsua    104190   72740   33302   23716.  23716.  0.71216  0.71216
o-g 01 rad  n18    amsua    104715   85857   35576   19844.  19844.  0.55779  0.55779
```

For O-A, the stage after the second outer loop:

```
o-g 03 rad  n15    amsua   104190   72740   40370   6456.7   6456.7    0.15994 0.15994
o-g 03 rad  n18    amsua   104715   85857   41795   11666.    11666.   0.27911 0.27911
```

From the above information, it can be seen that AMSU-A data from NOAA-15 have 104190 observations within the analysis time window and domain. After thinning, 72740 of this data type remained, and only 33302 were used in the analysis. The penalty for this data decreased from 23716 to 6456.7 after 2 outer loops. It is also very interesting to see that the number of AMSU-A observations assimilated in the O-A calculation increase to 40370 from 33302.
The statistics for each channel can be viewed in the *fort.207* file as well. Below channels from AMSU-A NOAA-15 are listed as an example:

For O-B, the stage before the first outer loop:

```
    1    1 amsua_n15          2659    481      3.000   0.7857922   1.0750014   0.2510086
2.2294483   1.9531543
    2    2 amsua_n15          2626    515      2.000   0.1567928   1.1868261   0.2814499
2.0916565   1.7223446
    3    3 amsua_n15          3047     93      2.000   1.3568136  -1.2600380   0.3436821
1.7979629   1.2825657
    4    4 amsua_n15          3140      0      0.600  -0.2667465  -0.3694930   0.5405917
0.5901848   0.4602098
    5    5 amsua_n15          3140      5      0.300  -0.0501683  -0.1470458   0.4435459
0.2552532   0.2086426
    6    6 amsua_n15          4746     81     -0.230  -1.7252094  -0.0636300   1.1587036
0.2995066   0.2926695
    7    7 amsua_n15          5562     28      0.250  -0.0190568   0.0421813   0.7292631
0.2646854   0.2613027
    8    8 amsua_n15          5476    114      0.275   0.0315037   0.1760652   0.9308259
0.3392216   0.2899523
    9    9 amsua_n15          4984    606      0.340   0.1750463   0.4949361   1.5810769
0.5515098   0.2433132
   10   10 amsua_n15           112   5478      0.400   0.6976028   0.8149763   2.4048381
0.8200668   0.0912314
   15   15 amsua_n15          2556    586      3.000   1.3418448  -1.5261734   0.3402708
2.4695874   1.9415604
```

For O-A, the stage after the second outer loop:

```
    1    1 amsua_n15          3039    402      3.000   0.2912526   0.3507109   0.1297502
1.7394024   1.7036791
    2    2 amsua_n15          3012    429      2.000   0.7399610   0.6570999   0.1116286
1.5253063   1.3765098
    3    3 amsua_n15          3342     97      2.000   1.6299453  -0.6625348   0.1591153
1.4080860   1.2424789
    4    4 amsua_n15          3439      0      0.600  -0.0971460   0.0036394   0.1056795
0.3050847   0.3050630
    5    5 amsua_n15          3439      4      0.300   0.0709002  -0.0106937   0.1696502
0.1714584   0.1711246
    6    6 amsua_n15          5167     15     -0.230  -1.6984752  -0.0286474   0.7821360
0.2508623   0.2492212
    7    7 amsua_n15          5579     11      0.250  -0.0553411  -0.0039809   0.1654485
0.1260928   0.1260299
    8    8 amsua_n15          5584      6      0.275  -0.0691681  -0.0030189   0.1463290
0.1306469   0.1306120
    9    9 amsua_n15          5590      0      0.340  -0.1612141   0.0014662   0.1140552
0.1459184   0.1459110
   10   10 amsua_n15          4397   1193      0.400   0.6138809   0.1808972   0.3292628
0.3237209   0.2684612
   15   15 amsua_n15          2949    491      3.000   1.9199387  -0.6627017   0.1431051
1.7816705   1.6538368
```

The second column is channel number for AMSU-A and the last column is the standard deviation for each channel. It can be seen that most of the channels fit better to the observations after the second outer loop.

5.2.3.2: Check the analysis increment

The same methods for checking the optimal minimization as demonstrated in section 5.1.4.2 can be used for radiance assimilation. Similar features to the conventional assimilation should be seen with the minimization. The figures below show detailed information on how the radiance data impact the analysis results on top of the conventional data. Using the same NCL script as in section 5.1.4.3, analysis increment fields are plotted comparing the analysis results with radiance and conventional data to the analysis results with conventional data assimilation only. The Fig 5.5 is for level 49 and the Fig.5.4 is for level 6, which represent the maximum temperature increment level (49) and maximum moisture increment level (6).

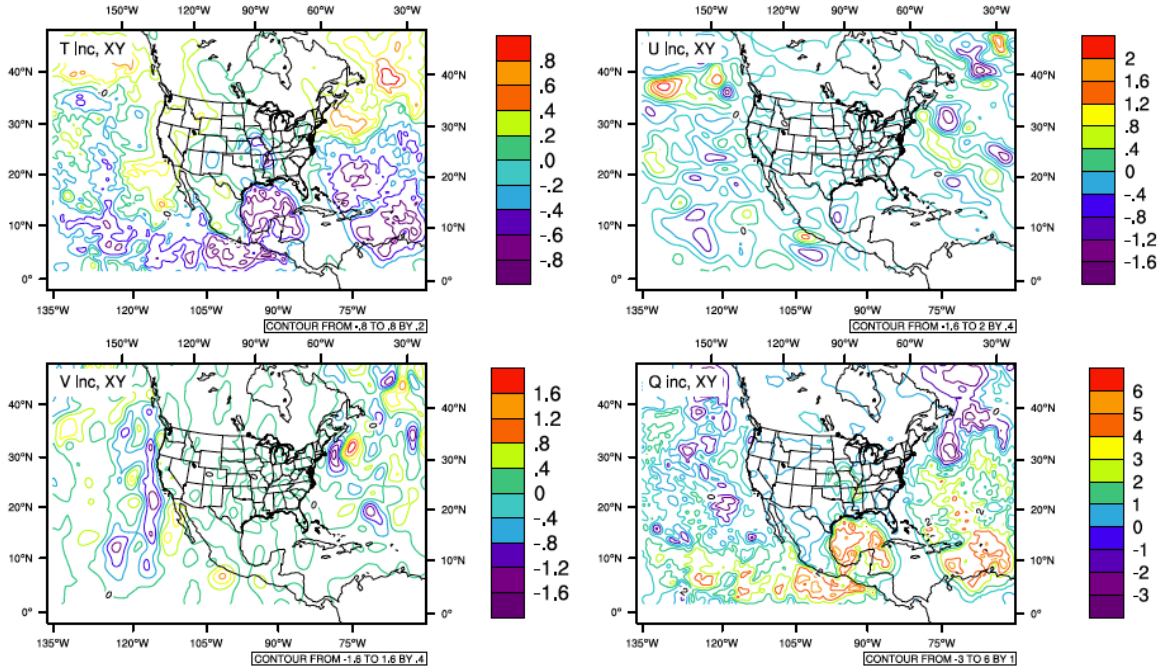

Figure 5.4: Analysis increment fields of PrepBUFR and Radiance data analysis comparing to the analysis with PREPBUFR only at level 6

In order to fully understand the analysis results, the following needs to be understood:

1. The weighting functions of each channel and the data coverage at this analysis time. There are several sources on the Internet to show the weighting function for the AMSU-A channels. Channel 1 is the moisture channel, while the others are mainly temperature channels (Channels 2, 3 and 15 also have large moisture signals). Because

a model top of 10 mb was specified for this case study, the actual impact should come from channels below channel 12.

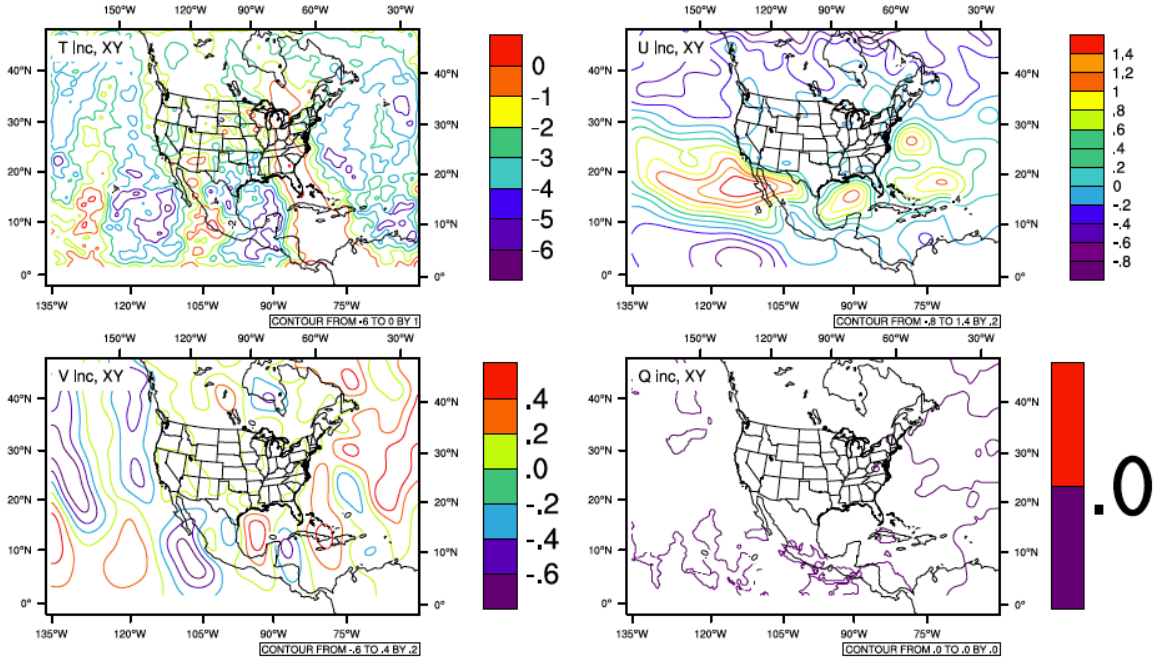

Figure 5.5: Analysis increment fields PrepBUFR and Radiance data analysis comparing to the analysis with PREPBUFR only at level 49

2. The usage of each channel is located in the file named '*satinfo*' in the run directory. The first two columns show the observation type and platform of the channels and the third column tells us if this channel is used in the analysis. Because a lot of amsua_n15 and amsua_n18 data were used, they should be checked in detail. In this case, Channels 6, 11 and 14 from amsua_n15 and channels 9 and 14 from amsua_n18 were turned off.

3. Thinning information: a quick look at the namelist in the run directory: *gsiparm.anl* shows that both amsua_n15 and amsu_n18 using thinning grid 2, which is 60 km. In this case, the grid spacing is 30 km, which indicates to use the satellite observations every four grid-spaces, which might be a little dense.

4. Bias correction: radiance bias correction was previously discussed. It is very important for a successful radiance data analysis. The run script can only link to the old bias correction coefficients that are provided as an example in *./fix*:

```
SATANGL=${FIX_ROOT}/global_satangbias.txt
cp ${FIX_ROOT}/ sample.satbias    ./satbias_in
```

Users can also download the operational bias correction coefficients during the experiment period as a starting point to calculate the coefficients suitable for their experiments.

Radiance bias correction for regional analysis is a difficult issue because of the limited coverage of radiance data. This topic is out of the scope of this document, but this issue should be considered and understood when using GSI with radiance applications.

5.3. Assimilating GPS Radio Occultation Data with Regional GSI

5.3.1: Run script

The addition of GPS Radio Occultation (RO) data into the GSI analysis is similar to that of adding radiance data. In the example below, the RO data is used as refractivity. There is also an option to use the data as bending angles. The same run scripts used in sections 5.1.1 and 5.2.1 can be used with the addition of the following link to the observations:

```
ln -s ${OBS_ROOT}/gdas1.t00z.gpsro.tm00.bufr_d gpsrobufr
```

For this case study, the GPS RO BUFR file was downloaded and saved in the `OBS_ROOT` directory. The file is linked to the name *gpsrobufr*, following the namelist section `&OBS_INPUT`:

```
!  dfile          dtype       dplat     dsis                 dval    dthin dsfcalc
   gpsrobufr      gps_ref     null      gps                  1.0     0     0
```

This indicates that GSI is expecting a GPS refractivity BUFR file named `gpsrobufr`. In the following example, GPS RO and conventional observations are both assimilated. Change the run directory name in the run scripts to reflect this test:

```
WORK_ROOT=/scratch1/gsiprd_${ANAL_TIME}_gps_prepbufr
```

5.3.2: Run GSI and check the run status

The process of running GSI is the same as described in section 5.1.2. Once *run_gsi_regional.ksh* has been submitted, move into the working directory, *gsiprd_2014061700_gps_prepbufr*, to check the GSI analysis results. The run directory will look exactly the same as with the conventional data, with the exception of the link to the GPS RO BUFR files used in this case. Following the same steps as in section 5.1.3, check the *stdout* file to see if GSI has run through each part of the analysis process

successfully. In addition to the information outlined for the conventional run, the GPS RO BUFR files should have been read in and distributed to each sub domain:

```
OBS_PARA: ps                    1429      3190      4655      6774
OBS_PARA: t                     2564      5200      7057     11128
OBS_PARA: q                     2346      4626      6148      8128
OBS_PARA: pw                      65        80        63        49
OBS_PARA: uv                    3358      6453      8091     11998
OBS_PARA: gps_ref               1799      1368      2664      3520
```

Comparing the output to the content in section 5.1.3, it can be seen that the GPS RO refractivity data have been read in and distributed to four sub-domains successfully.

5.3.3: Diagnose GSI analysis results

5.3.3.1: Check file *fort.212*

The file *fort.212* is the file for the fit of gps data in fractional difference. It has the same structure as the fit files for conventional data. Below is a quick look to be sure the GPS RO data were used:

Observation – Background (O-B)

```
                              ptop  1000.0  900.0  800.0  600.0  400.0  300.0  250.0  200.0
150.0  100.0   50.0     0.0
     it     obs    type styp  pbot  1200.0 1000.0  900.0  800.0  600.0  400.0  300.0  250.0
200.0  150.0  100.0  2000.0
-------------------------------------------------------------------------------------------
-------------------------------
 o-g 01             all      count       0     13     58    223    355    342    232    261
326    440    729    3740
 o-g 01             all       bias    0.00  -0.76  -0.03  -0.06  -0.04   0.01  -0.03   0.04
-0.04  -0.16  -0.18   -0.14
 o-g 01             all        rms    0.00   1.41   0.75   0.96   0.79   0.35   0.32   0.42
0.54   0.57   0.55    0.59
```

Observation – Analysis (O-A)

```
                              ptop  1000.0  900.0  800.0  600.0  400.0  300.0  250.0  200.0
150.0  100.0   50.0     0.0
    it     obs    type styp  pbot  1200.0 1000.0  900.0  800.0  600.0  400.0  300.0  250.0
200.0  150.0  100.0  2000.0
-------------------------------------------------------------------------------------------
-------------------------------
 o-g 03             all      count       1     18     65    229    355    342    231    266
330    440    731    3776
 o-g 03             all       bias   -0.40  -0.43   0.03   0.02  -0.02  -0.01  -0.02   0.00
0.01  -0.01  -0.02    0.00
 o-g 03             all        rms    0.40   1.03   0.54   0.59   0.70   0.26   0.14   0.20
0.24   0.28   0.39    0.41
```

It can be seen that there most of the GPS RO data are located in the upper levels, with a total of 3740 observations used in the analysis during the 1st outer loop, and 3776 used to calculate O-A. After the analysis, the data bias reduced from -0.14 to 0.00, and the rms was reduced from 0.59 to 0.41. It can be concluded that the analysis with GPS RO data looks reasonable from these statistics.

5.3.3.2: Check the analysis increment

The same methods for checking the minimization in section 5.1.4.2 can be used for the GPS RO assimilation.

The following figures give detailed information of how the new data impacts the analysis result. Using the NCL script used in section 5.1.4.3, analysis increment fields are plotted comparing the analysis results with GPS RO and conventional data to the analysis results with conventional data assimilation only for level 48, which represents the maximum temperature increment.

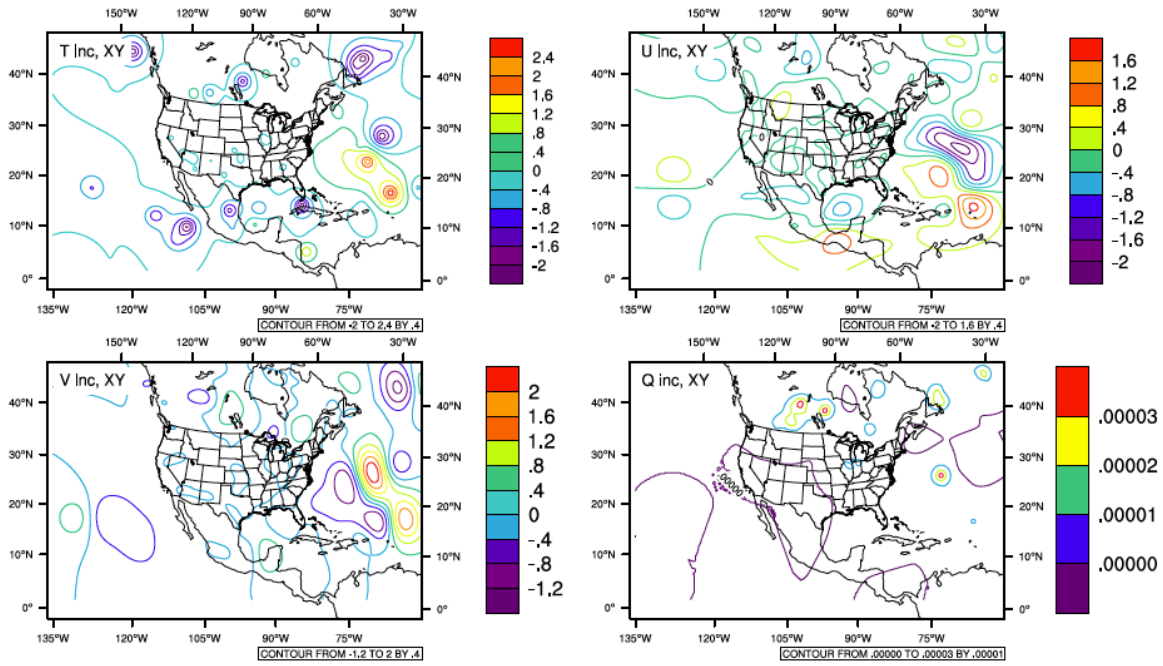

Figure 5.6: Analysis increment fields comparing to the analysis with PrepBUFR only at level 48

5.4 Introduction to GSI hybrid analysis

The hybrid ensemble-3DVAR analysis is an important analysis option in the GSI system that has been used by operations. It provides the ability to bring the flow dependent background error covariance into the analysis based on ensemble forecasts. If ensemble forecasts have been generated, setting up GSI to do a hybrid analysis is straightforward and only requires two changes in the run script in addition to the current 3DVAR run script:

Change 1: Link the ensemble members to the GSI run directory

This change is to link the ensemble members to the GSI run directory and assign each ensemble member a name that GSI recognizes. The release version GSI can accept 4 kinds of ensemble forecasts, which is controlled by the namelist variable *regional_ensemble_option*. Table 5.1 gives a list of options for *regional_ensemble_option* and the naming convention for linking the ensemble to GSI recognized names

Table 5.1 the list of ensemble forecasts that can be read by GSI hybrid

regional_ ensemble_ option	explanation	Function called	GSI recognized ensemble file names
1	GFS ensemble internally interpolated to hybrid grid	get_gefs_for_regional	*filelist : a text file include path and name of ensemble files*
2	ensembles are WRF NMM (HWRF) format	get_wrf_nmm_ensperts	*d01_en001, d01_en002, ...*
3	ensembles are ARW netcdf format	get_wrf_mass_ensperts_netcdf	*wrf_en001, wrf_en002, ...*
4	ensembles are NEMS NMMB format	get_nmmb_ensperts	*nmmb_ens_mem001, nmmb_ens_mem002, ...*

Users have to change the GSI run script to add the links to the ensemble forecasts if they want to use the GSI hybrid function. Below is an example of using ensembles of ARW netcdf format, assuming that all the ensemble members are located in a directory defined by the parameter *${mempath}* and the ensemble members have a name such as: *wrfout_d01_${iiimem}*, where *${iiimem}* is an integer indicating the ensemble member ID. The following lines should be added to the run script with loop *iiimem* from 1 to ensemble member number:

```
if [ -r  ${mempath}/wrfout_d01_${iiimem} ]; then
   ln -sf ${mempath}/wrfout_d01_${iiimem} ./wrf_en${iiimem}
else
```

```
   echo "member ${mempath}/wrfout_d01_${iiimem} does not exit"
fi
```

Change 2: Set up the namelist options in section HYBRID_ENSEMBLE

Users need to set `l_hyb_ens=.true.` to turn on hybrid ensemble analysis. Commonly used namelist options for the hybrid analysis are listed below:

`l_hyb_ens` - if true, turn on hybrid ensemble option

`uv_hyb_ens` - if true, ensemble perturbation wind variables are u and v; otherwise, ensemble perturbation wind variables are stream function and velocity potential functions.

`generate_ens` - if true, generate internal ensemble based on existing background error; recommended to be false.

`n_ens` - number of ensemble members.

`beta1_inv` - (1/beta1), the weight given to the static background error covariance. 0 <= *beta1_inv* <= 1, should be tuned for optimal performance; *beta2_inv = 1 - beta1_inv* is the weight given to the ensemble derived covariance
=1, ensemble information turned off
=0, static background errors turned off

`s_ens_h` - homogeneous isotropic horizontal ensemble localization scale (km)

`s_ens_v` - vertical localization scale.
If positive, in grid units;
if negative, in lnp unit.

`regional_ensemble_option` - integer, used to select type of ensemble to read in for regional applications. Currently takes values from 1 to 4:
=1: use GEFS internally interpolated to ensemble grid.
=2: ensembles are in WRF NMM format
=3: ensembles are in ARW netcdf format.
=4: ensembles are in NEMS NMMB format.

`grid_ratio_ens` -for regional runs, ratio of ensemble grid resolution to analysis grid resolution. If turned on and specified an appropriate value, could increase the computational efficiency.

Please note: the parameters `s_ens_h`, `s_ens_v`, and `beta1_inv` are tunable parameters. They should be tuned for best performance.

After setup of the namelist parameters and the path and name of the ensemble members, GSI can be run following the other 3DVAR cases introduced in this Chapter. And the same procedures could be followed as in the previous sections to check the run status and diagnose the GSI analysis.

Summary

This chapter applied the knowledge from the previous chapters to demonstrate how to set up, run, and analyze GSI for various applications. It is important to always check for successful GSI analysis, as running to completion does not always indicate a successful run. Using the tools and methods described in this chapter, a complete picture of the GSI analysis can be obtained.

It is important to realize that GSI applications are not limited to regional analysis with WRF. Other GSI applications, including the global analysis for GFS, the 2D surface analysis within RTMA, and more details about the GSI hybrid analysis will be introduced in the Advanced GSI User's Guide.

Appendix A: GSI Community Tools

A.1 BUFR Format and BUFR Tools

Under *./util/bufr_tools*, there are many Fortran examples to illustrate basic BUFR/PrepBUFR file process skills such as encoding, decoding, and appending. For details of these examples and the BUFR format, please see the BUFR/PrepBUFR User's Guide, which is freely available on-line

http://www.dtcenter.org/com-GSI/BUFR/docs/index.php

The observation BUFR files generated by NCEP (for example, PrepBUFR and BUFR files from NCEP ftp server or *gdas1.t12z.prepbufr.nr* in tutorial case) are in Big Endian binary format. A small C program called *ssrc.c* under *./util/bufr_tools* can be used to convert Big Endian BUFR file to Little Endian file.The program *ssrc.c* may be compiled using any standard C compiler. After compiling *ssrc.c* to generate executable: *ssrc.exe,* you can byte-swap a Big Endian BUFR/PrepBUFR file to a Little_Endian file by:

ssrc.exe < name of Big Endian bufr file > name of Little Endian bufr file

In the on-line tutorial, the PrepBUFR file *little_endian.gdas1.t12z.prepbufr.nr* has the same content as *gdas1.t12z.prepbufr.nr*, but has been byte-swapped to Little Endian.

Since release 3.2, BUFRLIB can automatically identify and handle either byte orders. For Intel and PGI compilers on Linux, the Big Endian BUFR/PrepBUFR files can be used by GSI **without** byte swap.

A.2 Read GSI Diagnostic Files

Lots of useful information about how one observation was used in the analysis such as innovation, observation values, observation error and adjusted observation error, and quality control information, has been saved in diagnostic files. To generate these diagnosis files, namelist variable `write_diag` in namelist section &`SETUP` needs to be true. The `write_diag` variable has been introduced in Part 4 of Section 3.4. The following is an example of using the `write_diag` variable to control diagnostic files. When we set the number of outer loops to 2, and set the `write_diag` namelist variable to the following:

```
write_diag(1)=.true.,write_diag(2)=.false.,write_diag(3)=.true.,
```

GSI will write out diagnostic files before the start of the 1st outer loop start (O-B) and after the completion of the 2nd outer loop finish (O-A). We don't want GSI to write out diagnosis files after the 1st outer loop because we set `write_diag(2)=.false.`

This is what we set in our example case described in section 5.2. From this case, we can see the following diagnostic files generated from the GSI analysis:

```
diag_amsua_metop-a_anl.2014061700  diag_amsua_n18_ges.2014061700
diag_amsua_metop-a_ges.2014061700  diag_conv_anl.2014061700
diag_amsua_n15_anl.2014061700      diag_conv_ges.2014061700
diag_amsua_n15_ges.2014061700      diag_hirs4_metop-a_anl.2014061700
diag_amsua_n18_anl.2014061700      diag_hirs4_metop-a_ges.2014061700
```

All files are identified with a filename containing three elements. The first element "**diag"** indicates these are combined diagnostic files. The second element identifies the observation type (here, "**conv"** means conventional observation from prepbufr and "**amsua_n15"** corresponds to radiance observation AMSU-A from NOAA 15). The last element identifies which step of outer loop the files were generated for. Here, "**anl**" means the contents were written after the last outer loop (from `write_diag(3)=.true.`) and "**ges**" means the contents were written before the first output loop (from `write_diag(1)=.true.`).

To help users to read the information from these diagnostic files, we have provided two Fortran programs in the *./util/Analysis_Utilities/read_diag/* directory:

read_diag_conv.f90 : Reads the diagnostic files for conventional observations. For example: `diag_conv_anl.2014061700` and `diag_conv_ges.2014061700`

read_diag_rad.f90 : Reads the diagnostic files for radiance observation. For example:

```
diag_amsua_n15_ges.2014061700 diag_hirs4_metop-a_anl.2014061700
diag_amsua_n18_anl.2014061700 diag_hirs4_metop-a_ges.2014061700
```

To compile the programs, use the makefile provided:

./make

Note: since information in the GSI *include* directory is required, the GSI must have been compiled first.

To run *read_diag_conv.exe*, a namelist file *namelist.conv* needs to be in the directory along with the executable. The *namelist.conv* only has two parameters:

```
&iosetup
  infilename='./diag_conv_anl',      : The path and name of GSI diagnosis file
  outfilename='./results_conv_anl', : The path and name of a text file used to
                                       save the content of the diagnostic file
 /
```

The user can set the test case directory and file `diag_conv_anl.2014061700` from section 5.2 as the entry for *infilename* in the namelist, then run the executable

./read_diag_conv.exe

The results are placed in the file specified by the *outfilename* entry in the namelist. In this case that would be a file `results_conv_anl` located in the directory where the executable was run.

Similarly, to run *read_diag_rad.exe,* the namelist file *namelist.rad* is needed. It contains the same parameters as *namelist.conv* but it links to radiance diag files. After setting it to use the same case from section 5.2, such as:

```
&iosetup
  infilename='(test directory)/diag_amsua_n18_ges.2014061700',
  outfilename='./results_amsua_n18_ges',
 /
```

Running the executable creates the text file *results_amsua_n18_ges* specified by the namelist in the directory *read_diag_rad.exe* runs.

For the conventional observations, the data is stored in two arrays: **rdiagbuf** and **cdiagbuf**. Their contents are listed as follows, for temperature as an example:

```
cdiagbuf     = station id
rdiagbuf(1)  = observation type
rdiagbuf(2)  = observation subtype
rdiagbuf(3)  = observation latitude (degrees)
rdiagbuf(4)  = observation longitude (degrees)
rdiagbuf(5)  = station elevation (meters)
rdiagbuf(6)  = observation pressure (hPa)
rdiagbuf(7)  = observation height (meters)
rdiagbuf(8)  = observation time (hours relative to analysis time)
rdiagbuf(9)  = input prepbufr qc or event mark
rdiagbuf(10) = setup qc or event mark (currently qtflg only)
rdiagbuf(11) = read_prepbufr data usage flag
rdiagbuf(12) = analysis usage flag (1=use, -1=not used)
rdiagbuf(13) = nonlinear qc relative weight
rdiagbuf(14) = prepbufr inverse obs error (K**-1)
rdiagbuf(15) = read_prepbufr inverse obs error (K**-1)
rdiagbuf(16) = final inverse observation error (K**-1)
rdiagbuf(17) = observation (K)
rdiagbuf(18) = obs-ges used in analysis (K)
rdiagbuf(19) = obs-ges without bias correction (K)
```

For wind observations, the content after index 16 is:

```
rdiagbuf(17) = earth relativeu wind component observation (m/s)
rdiagbuf(18) = earth relativeu obs-ges used in analysis (m/s)
rdiagbuf(19) = earth relativeu obs-ges w/o bias correction (m/s)
rdiagbuf(20) = earth relativev wind component observation (m/s)
rdiagbuf(21) = earth relativev obs-ges used in analysis (m/s)
rdiagbuf(22) = earth relativev obs-ges w/o bias correction (m/s)
```

The *read_diag_conv.exe* reads these arrays and outputs important information in the text file *results_conv_anl* specified by the user in the `&iosetup` namelist. Example:

```
     station          obs          obs       obs       obs       obs        usag      obs        O-B
       ID             type         time      latitude  longitude pressure           value
ps @ 46047     : 180      -0.17    32.40   240.50 1014.30    1    1014.30   -0.09
 t @ 72293     : 120      -0.98    32.85   242.88  996.00    1     297.25    1.26
uv @ 72293     : 220      -0.98    32.85   242.88  996.00    1       3.90    0.38    3.30     2.66
```

For wind, the last 4 columns are the wind components in the order of: U observation, O-B for U, V observation, O-B for V.

For radiance observations, the data is stored in two arrays: **diagbuf** and **diagbufchan**. Their contents are listed as follows (please refer to *src/main/setuprad.f90* for more details):

```
  diagbuf(1)  = observation latitude (degrees)
  diagbuf(2)  = observation longitude (degrees)
  diagbuf(3)  = model (guess) elevation at observation location
  diagbuf(4)  = observation time (hours relative to analysis time)

  diagbuf(5)  = sensor scan position
  diagbuf(6)  = satellite zenith angle (degrees)
  diagbuf(7)  = satellite azimuth angle (degrees)
  diagbuf(8)  = solar zenith angle (degrees)
  diagbuf(9)  = solar azimuth angle (degrees)
  diagbuf(10) = sun glint angle (degrees) (sgagl)

  diagbuf(11) = surface fractional coverage by water
  diagbuf(12) = surface fractional coverage by land
  diagbuf(13) = surface fractional coverage by ice
  diagbuf(14) = surface fractional coverage by snow
if(.not. retrieval)then
  diagbuf(15) = surface temperature over water (K)
  diagbuf(16) = surface temperature over land (K)
  diagbuf(17) = surface temperature over ice (K)
  diagbuf(18) = surface temperature over snow (K)
  diagbuf(19) = soil temperature (K)
  diagbuf(20) = soil moisture
  diagbuf(21) = surface land type
else
  diagbuf(15) = SST first guess used for SST retrieval
  diagbuf(16) = NCEP SST analysis at t
  diagbuf(17) = Physical SST retrieval
  diagbuf(18) = Navy SST retrieval
  diagbuf(19) = d(ta) corresponding to sstph
  diagbuf(20) = d(qa) corresponding to sstph
  diagbuf(21) = data type
endif
  diagbuf(22) = vegetation fraction
  diagbuf(23) = snow depth
  diagbuf(24) = surface wind speed (m/s)
!   Note:  The following quantities are not computed for all sensors
  if (.not.microwave) then
     diagbuf(25)  = cloud fraction (%)
     diagbuf(26)  = cloud top pressure (hPa)
  else
     diagbuf(25)  = cloud liquid water (kg/m**2)
```

```
     diagbuf(26)  = total column precip. water (km/m**2)
  endif

  diagbuf(27) = foundation temperature: Tr
  diagbuf(28) = diurnal warming: d(Tw) at depth zob
  diagbuf(29) = sub-layer cooling: d(Tc) at depth zob
  diagbuf(30) = d(Tz)/d(Tr)
```

diagbufchan include loop through channel *i* from *1* to *nchanl*:

```
    diagbufchan(1,i)= observed brightness temperature (K)
    diagbufchan(2,i)= observed - simulated Tb with bias corrrection (K)
    diagbufchan(3,i)= observed - simulated Tb with no bias correction (K)
    diagbufchan(4,i)= inverse observation error
    diagbufchan(5,i)= quality control mark or event indicator
    diagbufchan(6,i)= surface emissivity
    diagbufchan(7,i)= stability index
    diagbufchan(8,i)= d(Tb)/d(Ts)
    do j=1,npred+1
       diagbufchan(7+j,i)= Tb bias correction terms (K)
    end do
```

In the sample output file *results_amsua_n18_ges*, only the observation location and time are written in the file. Users can write out other information based on the list.

A.3 Read and Plot Convergence Information from *fort.220*

In section 4.6, we introduced how to check the convergence information in the *fort.220* file. Further detail on the *fort.220* convergence information can be found in the Advanced User's Guide. Here, we provide tools to filter this file and to plot the values of the cost function and the norm of gradient during each iteration.

These tools - one ksh script and one ncl script – are in:
./util/Analysis_Utilities/plot_cost_grad directory:

The ksh script, *filter_fort220.ksh*, only has one line:

```
grep 'cost,grad,step,b' fort.220 | sed -e 's/cost,grad,step,b,step? =   //g' | sed -e
's/good//g' > cost_gradient.txt
```

To run *filter_fort220.ksh*, the *fort.220* needs to be in the same directory. The script will filter out the values of the cost function and the norm of gradient at each iteration from *fort.220* into a text file called *cost_gradient.txt*.

Once the file *cost_gradient.txt* is ready, run ncl script *GSI_cost_gradient.ncl* to generate the plot:

```
ncl GSI_cost_gradient.ncl
```

The pdf file *GSI_cost_gradient.pdf* is created. The pdf file contains plots of the convergence of the GSI analysis like those in section 4.6.

A.4 Plot Single Observation Test Result and Analysis Increment

In Section 4.2, we introduced how to do a single observation test for GSI. Here we provide users with the ncl scripts to plot the single observation test results.

There are 5 ncl scripts in the . *util/Analysis_Utilities/plots_ncl* directory:

GSI_singleobs_arw.ncl	Plot single observation test with ARW NetCDF background
GSI_singleobs_nmm.ncl	Plot single observation test with NMM NetCDF background
Analysis_increment.ncl	Plot analysis increment from the case with ARW NetCDF background
Analysis_increment_nmm.ncl	Plot analysis increment from the case with NMM NetCDF background
fill_nmm_grid2.ncl	E grid to A grid convertor

The main difference between the ARW and NMM core used in GSI is that ARW is on a C grid, while NMM is on an E grid. *GSI_singleobs_nmm.ncl* calls *fill_nmm_grid2.ncl* to convert the E grid to an A grid for plotting, while *GSI_singleobs_arw.ncl* itself includes a C grid to A grid convertor.

Before running ncl scripts, users need to set up two links:

cdf_analysis	Link to analysis result in NetCDF format (*wrf_inout*)
cdf_bk	Link to background file in netCDF format

These scripts read in the analysis and background fields of temperature (T), U component of wind (U), V component of wind (V), and moisture (Q) and calculate the difference of the analysis field minus the background field. Then XY sections (left column) and XZ sections (right column) are plotted for T, U, V, and Q through the point that has maximum analysis increment of single observation. Here the default single observation test is T. If the user conducts other single observation tests, the corresponding changes should be made based on the current scripts.

The scripts *Analysis_increment.ncl* and *Analysis_increment_nmm.ncl* are very similar the one for the single observation but only the XY section for a certain analysis level is plotted.

For more information on how to use ncl, please check the NCL website at:

http://www.ncl.ucar.edu/

Appendix B: Content of Namelist Section OBS_INPUT

```
&OBS_INPUT
   dmesh(1)=120.0,dmesh(2)=60.0,dmesh(3)=30,time_window_max=1.5,ext_sonde=.true.,
 /
OBS_INPUT::
!  dfile          dtype       dplat     dsis                dval    dthin dsfcalc
   prepbufr       ps          null      ps                  1.0     0     0
   prepbufr       t           null      t                   1.0     0     0
   prepbufr       q           null      q                   1.0     0     0
   prepbufr       pw          null      pw                  1.0     0     0
   satwndbufr     uv          null      uv                  1.0     0     0
   prepbufr       uv          null      uv                  1.0     0     0
   prepbufr       spd         null      spd                 1.0     0     0
   prepbufr       dw          null      dw                  1.0     0     0
   radarbufr      rw          null      rw                  1.0     0     0
   prepbufr       sst         null      sst                 1.0     0     0
   gpsrobufr      gps_ref     null      gps                 1.0     0     0
   ssmirrbufr     pcp_ssmi    dmsp      pcp_ssmi            1.0    -1     0
   tmirrbufr      pcp_tmi     trmm      pcp_tmi             1.0    -1     0
   sbuvbufr       sbuv2       n16       sbuv8_n16           1.0     0     0
   sbuvbufr       sbuv2       n17       sbuv8_n17           1.0     0     0
   sbuvbufr       sbuv2       n18       sbuv8_n18           1.0     0     0
   hirs3bufr      hirs3       n16       hirs3_n16           0.0     1     0
   hirs3bufr      hirs3       n17       hirs3_n17           6.0     1     0
   hirs4bufr      hirs4       metop-a   hirs4_metop-a       6.0     2     0
   hirs4bufr      hirs4       n18       hirs4_n18           0.0     1     0
   hirs4bufr      hirs4       n19       hirs4_n19           1.0     2     0
   hirs4bufr      hirs4       metop-b   hirs4_metop-b       1.0     1     0
   gimgrbufr      goes_img    g11       imgr_g11            0.0     1     0
   gimgrbufr      goes_img    g12       imgr_g12            0.0     1     0
   airsbufr       airs        aqua      airs281SUBSET_aqua 20.0     2     0
   amsuabufr      amsua       n15       amsua_n15          10.0     2     0
   amsuabufr      amsua       n18       amsua_n18          10.0     2     0
   amsuabufr      amsua       n19       amsua_n19          10.0     2     0
   amsuabufr      amsua       metop-a   amsua_metop-a      10.0     2     0
   amsuabufr      amsua       metop-b   amsua_metop-b      10.0     2     0
   airsbufr       amsua       aqua      amsua_aqua          5.0     2     0
   amsubbufr      amsub       n17       amsub_n17           1.0     1     0
   mhsbufr        mhs         n18       mhs_n18             3.0     2     0
   mhsbufr        mhs         n19       mhs_n19             3.0     2     0
   mhsbufr        mhs         metop-a   mhs_metop-a         3.0     2     0
   mhsbufr        mhs         metop-b   mhs_metop-b         3.0     2     0
   ssmitbufr      ssmi        f13       ssmi_f13            0.0     2     0
   ssmitbufr      ssmi        f14       ssmi_f14            0.0     2     0
   ssmitbufr      ssmi        f15       ssmi_f15            0.0     2     0
   amsrebufr      amsre_low   aqua      amsre_aqua          0.0     2     0
   amsrebufr      amsre_mid   aqua      amsre_aqua          0.0     2     0
   amsrebufr      amsre_hig   aqua      amsre_aqua          0.0     2     0
   ssmisbufr      ssmis_las   f16       ssmis_f16           0.0     2     0
   ssmisbufr      ssmis_uas   f16       ssmis_f16           0.0     2     0
   ssmisbufr      ssmis_img   f16       ssmis_f16           0.0     2     0
   ssmisbufr      ssmis_env   f16       ssmis_f16           0.0     2     0
   gsnd1bufr      sndrd1      g12       sndrD1_g12          1.5     1     0
   gsnd1bufr      sndrd2      g12       sndrD2_g12          1.5     1     0
   gsnd1bufr      sndrd3      g12       sndrD3_g12          1.5     1     0
   gsnd1bufr      sndrd4      g12       sndrD4_g12          1.5     1     0
   gsnd1bufr      sndrd1      g11       sndrD1_g11          1.5     1     0
   gsnd1bufr      sndrd2      g11       sndrD2_g11          1.5     1     0
   gsnd1bufr      sndrd3      g11       sndrD3_g11          1.5     1     0
   gsnd1bufr      sndrd4      g11       sndrD4_g11          1.5     1     0
   gsnd1bufr      sndrd1      g13       sndrD1_g13          1.5     1     0
   gsnd1bufr      sndrd2      g13       sndrD2_g13          1.5     1     0
   gsnd1bufr      sndrd3      g13       sndrD3_g13          1.5     1     0
   gsnd1bufr      sndrd4      g13       sndrD4_g13          1.5     1     0
   gsnd1bufr      sndrd1      g15       sndrD1_g15          1.5     2     0
   gsnd1bufr      sndrd2      g15       sndrD2_g15          1.5     2     0
```

```
gsnd1bufr      sndrd3      g15       sndrD3_g15           1.5     2     0
gsnd1bufr      sndrd4      g15       sndrD4_g15           1.5     2     0
iasibufr       iasi        metop-a   iasi616_metop-a     20.0     1     0
gomebufr       gome        metop-a   gome_metop-a         1.0     2     0
omibufr        omi         aura      omi_aura             1.0     2     0
sbuvbufr       sbuv2       n19       sbuv8_n19            1.0     0     0
tcvitl         tcp         null      tcp                  1.0     0     0
seviribufr     seviri      m08       seviri_m08           1.0     1     0
seviribufr     seviri      m09       seviri_m09           1.0     1     0
seviribufr     seviri      m10       seviri_m10           1.0     1     0
iasibufr       iasi        metop-b   iasi616_metop-b      0.0     1     0
gomebufr       gome        metop-b   gome_metop-b         0.0     2     0
atmsbufr       atms        npp       atms_npp             0.0     1     0
crisbufr       cris        npp       cris_npp             0.0     1     0
mlsbufr        mls30       aura      mls30_aura           0.0     0     0
oscatbufr      uv          null      uv                   0.0     0     0
prepbufr       mta_cld     null      mta_cld              1.0     0     0
prepbufr       gos_ctp     null      gos_ctp              1.0     0     0
refInGSI       rad_ref     null      rad_ref              1.0     0     0
lghtInGSI      lghtn       null      lghtn                1.0     0     0
larcInGSI      larccld     null      larccld              1.0     0     0
```

Appendix C: GSI Namelist: Name, Default value, Explanation

The following are lists and explanations of the GSI namelist variables. You can also find them in the source code ***gsimod.F90***.

Variable name	Default value	Description
&SETUP		**General control namelist**
gencode	80	source generation code
factqmin	1	weighting factor for negative moisture constraint
factqmax	1	weighting factor for supersaturated moisture constraint
clip_supersaturation	.false.	flag to remove supersaturation during each outer loop
factv	1	weighting factor for negative visibility constraint
deltim	1200	model timestep
dtphys	3600	physics timestep
biascor	-1	background error bias correction coefficient
bcoption	1	0=ibc (no bias correction to bkg); 1= sbc(original implementation)
diurnalbc	0	1= diurnal bias; 0= persistent bias
niter(0:50)	0, …	Maximum number of inner loop iterations for each outer loop
niter_no_qc (0:50)	1000000	Inner loop iteration at which to turn on variational quality control
miter	1	number of outer loops
qoption	1	option for moisture analysis variable; 1:q/qsatg 2:normalized RH
pseudo_q2	.false.	breed between q1/q2 options, that is, (q1/sig(q))
nhr_assimilation	6	assimilation time interval (currently 6 hours for global, 3 hours for regional)
min_offset	3	time of analysis in assimilation window
iout_iter	220	output file number for iteration information
npredp	6	number of predictors for precipitation bias correction

retrieval	.false.	logical to turn off or on the SST physical retrieval
nst_gsi	0	indicator to control the Tr Analysis mode: 0 = no nst info ingsi at all; 1 = input nst info, but used for monitoring only 2 = input nst info, and used in CRTM simulation, but no Tr analysis 3 = input nst info, and used in CRTM simulation and Tr analysis is on
nst_tzr	0	indicator to control the Tzr_QC mode: 0 = no Tz retrieval; 1 = Do Tz retrieval and applied to QC
nstinfo	0	number of nst variables
fac_dtl	0	index to apply diurnal thermocline layer or not: 0 = no; 1 = yes
fac_tsl	0	index to apply thermal skin layer or not: 0 = no; 1 = yes.
tzr_bufrsave	.false.	logical to turn off or on the bufr Tz retrieval file true=on
diag_rad	.true.	logical to turn off or on the diagnostic radiance file (true=on)
diag_pcp	.true.	logical to turn off or on the diagnostic precipitation file (true=on)
diag_conv	.true.	logical to turn off or on the diagnostic conventional file (true=on)
diag_ozone	.true.	logical to turn off or on the diagnostic ozone file (true=on)
diag_aero	.false.	logical to turn off or on the diagnostic aerosol file (true=on)
diag_co	.false.	logical to turn off or on the diagnostic carbon monoxide file (true=on)
iguess	1	flag for guess solution (currently not working) -1 do not use guess file 0 write only guess file 1 read and write guess file 2 read only guess file
write_diag	.false., …	logical to write out diagnostic files for outer iteration
reduce_diag	.false.	namelist logical to produce reduced radiance diagnostic files

oneobtest	.false.	one observation test flag true=on
sfcmodel	.false.	if true, then use boundary layer forward model for surface temperature data.
dtbduv_on	.true.	logical for switching on (.true.) sensitivity of uv winds to microwave brightness temperatures if true, use d(microwave brightness temperature)/d(uv wind) in inner loop
ifact10	0	flag for recomputing 10m wind factor = 1 compute using GFS surface physics = 2 compute using MM5 surface physics = 0 or any other value - DO NOT recompute - use value from guess file
l_foto	.false.	option for First-Order Time extrapolation to observation
offtime_data	.false.	if true, then allow use of obs files with ref time different from analysis time. default value = .false., in which case analysis fails if observation file reference time is different from analysis time.
npred_conv_max	0	maximum number of conventional observation bias correction coefficients
id_bias_ps	0	prepbufr id to have conv_bias added for testing
id_bias_t	0	prepbufr id to have conv_bias added for testing
id_bias_spd	120	prepbufr id to have conv_bias added for testing
conv_bias_ps	0	magnitude of ps bias(mb)
conv_bias_t	0	magnitude of t bias(deg K)
conv_bias_spd	0	magnitude of spd bias(m/sec)
stndev_conv_ps	1.0	
stndev_conv_t	1.0	
stndev_conv_spd	1.0	
use_pbl	.false.	Logical flag to include PBL effects in tendency model.
use_compress	.false.	option to turn on the use of compressibility factors in geopotential heights
nsig_ext	13	number of layers above the model top which are necessary to compute the bending angle for gpsro
gpstop	30.0	maximum height for gpsro data assimilation. Reject anything above this height. (km)
perturb_obs	.false.	logical flag to perturb observation (true=on)

perturb_fact	1	magnitude factor for observation perturbation
oberror_tune	.false.	logical to tune (=true) oberror
preserve_restart_date	.false.	if true, then do not update regional restart file date.
crtm_coeffs_path	./	path of directory w/ CRTM coeffs files
berror_stats	berror_stats	filename if other than "berror_stats"
newpc4pred	.false.	option for additional preconditioning for pred coeff
adp_anglebc	.false.	option to perform variational angle bias correction
angord	0	order of polynomial for variational angle bias correction
passive_bc	.false.	option to turn on bias correction for passive (monitored) channels
use_edges	.true.	option to exclude radiance data on scan edges
biaspredvar	0.1	set background error variance for radiance bias coeffs
lobsdiagsave	.false.	write out additional observation diagnostics
l4dvar	.false.	turn 4D-Var on/off (default=off=3D-Var)
lbicg	.false.	use B-precond w/ bi-conjugate gradient for minimization
lsqrtb	.false.	Use sqrt(B) preconditioning
lcongrad	.false.	Use conjugate gradient/Lanczos minimizer
lbfgsmin	.false.	Use L-BFGS minimizer
ltlint	.false.	Use TL inner loop (ie TL intall)
nhr_obsbin	-1	length of observation bins
nhr_subwin	-1	length of weak constraint 4d-Var sub-window intervals
nwrvecs	-1	Number of precond vectors (Lanczos) or pairs of vectors (QN) being saved
iorthomax	0	max number of vectors used for orthogonalization of various CG options
ladtest	.false.	Run adjoint test
ladtest_obs	.false.	if true, doing the adjoint check for the observation operators
lgrtest	.false.	Run gradient test

lobskeep	.false.	keep obs from first outer loop for subsequent OL
lsensrecompute	.false.	does adjoint by recomputing forward solution
jsiga	-1	calculate approximate analysis errors from lanczos for jiter=jsiga
ltcost	.false.	calculate true cost when using Lanczos (this is very expensive)
lobsensfc	.false.	compute forecast sensitivity to observations
lobsensjb	.false.	compute Jb sensitivity to observations
lobsensincr	.false.	compute increment sensitivity to observations
lobsensadj	.false.	use adjoint of approx. Hessian to compute obs sensitivity
lobsensmin	.false.	use minimisation to compute obs sensitivity
iobsconv	0	compute convergence test in observation space =1 at final point, =2 at every iteration
idmodel	.false.	uses identity model when running 4D-Var (test purposes)
iwrtinc	.false.	when .t., writes out increments instead of analysis
jiterstart	1	first outloop iteration number
jiterend	1	last outloop iteration number
lobserver	.false.	when .t., calculate departure vectors only
lanczosave	.false.	save lanczos vectors for forecast sensitivity computation
llancdone	.false.	use to tell adjoint that Lanczos vecs have been pre-computed
lferrscale	.false.	Something related to forecast error
print_diag_pcg	.false.	logical turn on of printing of GMAO diagnostics in pcgsoi.f90
tsensible	.false.	option to use sensible temperature as the analysis variable. Works only for twodvar_regional=.true.
lgschmidt	.false.	option for re-biorthogonalization of the {gradx} and {grady} set from pcgsoi when twodvar_regional=.true.
lread_obs_save	.false.	option to write out collective obs selection info
lread_obs_skip	.false.	option to read in collective obs selection info

use_gfs_ozone	.false.	option to read in gfs ozone and interpolate to regional model domain
check_gfs_ozone_date	.false.	option to date check gfs ozone before interpolating to regional model domain
regional_ozone	.false.	option to turn on ozone in regional analysis
lwrite_predterms	.false.	option to write out actual predictor terms instead of predicted bias to the radiance diagnostic files
lwrite_peakwt	.false.	option to writ out the approximate pressure of the peak of the weighting function for satellite data to the radiance diagnostic files
use_gfs_nemsio	.false.	option to use nemsio to read global model NEMS/GFS first guess
liauon	.false.	treat 4dvar CV as tendency perturbation (default=false)
use_prepb_satwnd	.false.	allow using satwnd's from prepbufr (historical) file
l4densvar	.false.	logical to turn on ensemble 4dvar
ens4d_nstarthr	3	start hour for ensemble perturbations (generally should match min_offset)
use_gfs_stratosphere		When true, a guess gfs valid at the same time as the nems-nmmb guess is used to replace the upper levels with gfs values. The purpose of this is to allow direct use of gdas derived sat radiance bias correction coefs.
pblend0	152	The nems-nmmb vertical coordinate is smoothly merged with gfs above this level. Below this level, is original nems-nmmb.
pblend1	79.0	The nems-nmmb vertical coordinate is smoothly merged with gfs below this level. Above this level,is gfs.
step_start	1.e-4	initial stepsize in minimization
diag_precon	.false.	if true do preconditioning
lrun_subdirs	.false.	logical to toggle use of subdirectires at runtime for pe specific files
emiss_bc	.false.	option to turn on emissivity bias predictor
upd_pred	1	bias update indicator for radiance bias correction; 1.0=bias correction coefficients evolve
use_reflectivity	.false.	option of using reflectivity
lnested_loops	.false.	allow for nested resolution outer/inner loops

lwrite4danl	.false.	logical to write out 4d analysis states if 4dvar or 4denvar mode
lsingleradob	.false.	logical for single radiance observation assimilation. Uses existing bufr file and rejects all radiances that don't fall within a tight threshold around oblat/oblon (SINGLEOB_TEST)
ssmis_method	1	choose method for SSMIS noise reduction 0=no smoothing 1=default
ssmis_precond	0.01	weighting factor for SSMIS preconditioning (if not using newpc4pred)
R_option	.false.	Option to use variable correlation length for lcbas based on data density - follows Hayden and Purser (1995) (twodvar_regional only)

NOTE: for now, if in regional mode, then iguess=-1 is forced internally.

& GRIDOPTS		**Grid setup variables, including regional specific variables**
jcap	62	spectral resolution of the analysis
jcap_b	62	spectral resolution of background (model guess field)
nsig	42	number of sigma levels
nlat	96	number of latitudes
nlon	384	number of longitudes
hybrid		logical hybrid data file flag true=hybrid
nlat_regional	0	Number of y grid point in whole regional domain
nlon_regional	0	Number of x grid point in whole regional domain
diagnostic_reg	.false.	logical for regional debugging
update_regsfc	.false.	logical to write out updated surface fields to the regional analysis file (default = false)
netcdf	.false.	if true, then wrf files are in netcdf format, otherwise wrf files are in binary format.
regional	.false.	logical for regional GSI run
wrf_nmm_regional	.false.	logical for input from WRF NMM
nems_nmmb_regional	.false.	logical for input from NEMS NMMB
wrf_mass_regional	.false.	logical for input from WRF MASS-CORE (ARW)
twodvar_regional	.false.	logical for regional 2d-var analysis
filled_grid	.false.	logical to fill in points on WRF-NMM E-grid

half_grid	.false.	logical to use every other row of WRF-NMM E-Grid
nvege_type	24	number of types of vegetation; old=24, IGBP=20
nlayers(100)	1	number of sub-layers to break indicated model layer into prior to calling radiative transfer model
cmaq_regional	.false.	Background input is from CMAQ model
nmmb_reference_grid	H	='H', then analysis grid covers H grid domain = 'V', then analysis grid covers V grid domain
grid_ratio_nmmb	sqrt(2)	ratio of analysis grid to nmmb model grid in nmmb model grid units.
grid_ratio_wrfmass	1.0	ratio of analysis grid to wrf mass grid in wrf grid units
jcap_gfs		spectral truncation used to transform high wavenumber spectral coefficients to a coarser resolution grid,when use_gfs_ozone = .true. or use_gfs_stratosphere = .true.
use_sp_eqspac	.false.	if .true., then ensemble grid is equal spaced, staggered 1/2 grid unit off poles. if .false., then gaussian grid assumed for ensemble (global only)

&BKGERR		**Background error related variables**
vs	1/1.5	scale factor for vertical correlation lengths for background error
nhscrf	3	number of horizontal scales for recursive filter
hzscl(3)	1, 1, 1	scale factor for horizontal smoothing, n=1,number of scales (3 for now) specifies factor by which to reduce horizontal scales (i.e. 2 would then apply 1/2 of the horizontal scale)
hswgt(3)	1/3, 1/3, 1/3	empirical weights to apply to each horizontal scale
norh	2	order of interpolation in smoothing
ndeg	4	degree of smoothing in recursive filters
noq	3	1/4 of accuracy in compact finite differencing
bw	0	factor in background error calculation
norsp	0	order of interpolation for smooth polar cascade routine default is norsp=0, in which case norh is used with original polar cascade interpolation (global only).
fstat	.false.	logical to separate f from balance projection
pert_berr	.false.	logical to turn on random inflation/deflation of background error tuning parameters

pert_berr_fct	0	factor for increasing/decreasing berror parameters, this is multiplied by random number
bkgv_flowdep	.false.	flag to turn on flow dependence to background error variances
bkgv_rewgtfct	0	factor used to perform flow dependent reweighting of error variances
bkgv_write	.false.	flag to turn on=.true. /off=.false. generation of binary file with reweighted variances
fpsproj	.true.	controls full nsig projection to surface pressure
fut2ps		controls the projection from unbalance T to surface pressure
adjustozvar		adjusts ozone variances in the stratosphere based on guess field
cwcoveqqcov		sets cw Bcov to be the same as B-cov(q) (presently glb default)

&ANBKGERR		**Anisotropic background error related variables**
anisotropic	.false.	if true, then use anisotropic background error covariance
ancovmdl	0	covariance model settings - 0: pt-based, 1: ensemble based
triad4	.true.	for 2d variables, if true, use blended triad algorithm
ifilt_ord	4	filter order for anisotropic filters
npass	1	2×npass = number of factors in background error
normal	200	number of random vectors to use for filter normalization (if < 0 then slightly slower, but results independent of number of processors)
binom	.true.	if true, weight correlation lengths of factors using binomial distribution, with shortest scales on outside, longest scales on inside. This can help to produce smoother correlations in the presence of strong anisotropy
ngauss	3	number of Gaussians to add together in each factor
rgauss	0	multipliers on reference aspect tensor for each Gaussian factor
anhswgt	1.0	empirical weights to apply to each gaussian
an_vs	1	scale factor for background error vertical scales (temporary carry over from isotropic inhomogeneous option)

grid_ratio	2.0	ratio of coarse to fine grid in fine grid units
grid_ratio_p	0	ratio of coarse to fine grid in fine grid units for polar patches
nord_f2a	4	order of interpolation for transfer operators between filter grid and analysis grid
an_flen_u	1	coupling parameter for connecting horizontal wind to background error
an_flen_t	1	coupling parameter for connecting grad(potential temperature) to background error
an_flen_z	1	coupling parameter for connecting grad(terrain) to background error
rtma_subdomain _option	.false.	if true, then call alternative code which calls recursive filter directly from subdomain mode, bypassing transition to/from horizontal slabs. This is mainly to improve efficiency for 2d rtma analysis. at the moment, this only works for twodvar_regional=.true. rtma_subdomain_option will be forced to false when twodvar_regional=.false.
lreadnorm	.false.	if true, then read normalization from fixed files
nsmooth	0	number of 1-2-1 smoothing passes before and after background error application
nsmooth_shapiro	0	number of 2nd moment preserving (shapiro) smoothing passes before and after background error application. NOTE: default for nsmooth and nsmooth_shapiro is 0. if both are > 0, then nsmooth will be forced to zero.
afact0	0.0	anistropy effect parameter, the range must be in 0.0-1.0.
covmap	.false.	if true, covariance map would be drawn

&JCOPTS		**Constraint term in cost function (Jc)**
ljcdfi	.false.	if .false., uses original formulation based on wind, temp, and ps tends when .t. uses digital filter initialization of increments (4dvar)
alphajc	10.0	parameter for digital filter
switch_on_ derivatives	.false., …	if true, then compute horizontal derivatives of all state variables (to be used eventually for time derivatives, dynamic constraints and observation forward models that need horizontal derivatives)

tendsflag	.false.	if true, compute time tendencies
ljcpdry	.false.	when .t. uses dry pressure constraint on increment
bamp_jcpdry	0.0	parameter for pdry_jc
eps_eer	-1.0	Errico-Ehrendofer parameter for q-term in energy norm
ljc4tlevs	.false.	when true and in 4D mode, apply any weak constraints over all time levels instead of just at a single time

&STRONGOPTS **Strong dynamic constraint**

reg_tlnmc_type	1	=1 for 1st version of regional strong constraint =2 for 2nd version of regional strong constraint
tlnmc_option	0	integer flag for strong constraint (various capabilities for hybrid): =0: no TLNMC =1: TLNMC for 3DVAR mode =2: TLNMC on total increment for single time level only (for 3D EnVar) or if 4D EnVar mode, TLNMC applied to increment in center of window =3: TLNMC on total increment over all time levels (if in 4D EnVar mode) =4: TLNMC on static contribution to increment ONLY for any EnVar mode
nstrong	0	if > 0, then number of iterations of implicit normal mode initialization to apply for each inner loop iteration
period_max	1000000.0	cutoff period for gravity waves included in implicit normal mode initialization (units = hours)
period_width	1.0	defines width of transition zone from included to excluded gravity waves
nvmodes_keep	0	number of vertical modes to use in implicit normal mode initialization
baldiag_full	.false.	flag to toggle balance diagnostics for the full fields
baldiag_inc	.false.	flag to toggle balance diagnostics for the analysis increment

&OBSQC **Observation quality control variables**

Parameters used for gross error checks are set in file **convinfo** (ermin, ermax, ratio)

Parameters below used for nonlinear (variational) quality control

dfact	0	factor for duplicate observation at same location for conventional data

dfact1	3.0	time factor for duplicate observation at same location for conventional data
erradar_inflate	1	radar error inflation factor
tdrerr_inflate	.false.	logical for tdr obs error inflation
tdrgross_fact	1	factor applied to tdr gross error
oberrflg	.false.	logical for reading in new observation error table (if set to true)
vadfile	'none'	character(10) variable holding name of VAD wind bufr file
noiqc	.false	logical flag to bypass OI QC (if set to true)
c_varqc	1	constant number to control variance qc turning on speed
blacklst	.false.	logical for reading in raob blacklist (if set to true)
use_poq7	.false.	Logical to toggle accept (.true.) or reject (.false.) SBUV/2 ozone observations flagged with profile ozone quality mark
hilbert_curve	.false.	option for hilbert-curve based cross-validation. works only with twodvar_regional=.true.
tcp_refps	1000.0	reference pressure for tcps oberr calculation (mb)
tcp_width	50.0	parameter for tcps oberr inflation (width, mb)
tcp_ermin	0.75	parameter for tcps oberr inflation (minimum oberr, mb)
tcp_ermax	5.0	parameter for tcps oberr inflation (maximum oberr, mb)
qc_noirjaco3	.false.	controls whether to use O3 Jac from IR instruments
qc_noirjaco3_pole	.false.	controls wheter to use O3 Jac from IR instruments near poles
qc_satwnds	.true.	allow bypass sat-winds qc normally removing lots of mid-tropo obs
aircraft_t_bc_pof	.false.	logical for aircraft temperature bias correction, pof is used for predictor
aircraft_t_bc	.false.	logical for aircraft temperature bias correction
aircraft_t_bc_ext	.false.	logical for reading aircraft temperature bias correction from external file
buddycheck_t	.false.	When true, run buddy check algorithm on temperature observations
buddydiag_save	.false.	When true, output files containing buddy check QC info for all obs run through the buddy check
biaspredt	1	berror var for temperature bias correction coefficients
upd_aircraft	.true.	indicator if update bias at 06Z & 18Z

cleanup_tail	.false.	logical to remove tail number no longer used

& OBS_INPUT		**Controls input data**
dfile	‘ ‘	input observation file name
dtype	‘ ‘	observation type
dplat	‘ ‘	satellite (platform) id (for satellite data)
dsis	‘ ‘	sensor/instrument/satellite flag from satinfo files
dthin	‘ ‘	satellite group
dval	‘ ‘	relative value of each profile within group relative weight for observation = dval/sum(dval) within grid box
dmesh(max(dthin))		thinning mesh for each group mesh size (km) for radiance thinning grid (used in satthin)
dsfcalc	‘ ‘	specifies method to determine surface fields within a FOV. when equal to one, integrate model fields over FOV. when not one, bilinearly interpolate model fields to FOV center.
time_window_max	3	upper limit on time window for all input data
ext_sonde	.false.	logical for extended forward model on sonde data
l_foreaft_thin	.false.	separate TDR fore/aft scan for thinning

NOTE: current value for ndatmax is 200.

&SINGLEOB_TEST		**Single observation test case setup**
maginnov	1	magnitude of innovation for one observation
magoberr	1	magnitude of observational error
oneob_type	‘ ‘	observation type (t, u, v, etc.)
oblat	0	observation latitude
oblon	0	observation longitude
obpres	1000.0	observation pressure (hPa)
obdattim	2000010100	observation date (YYYYMMDDHH)
obhourset	0	observation delta time from analysis time
pctswitch	.false.	if .true. innovation & oberr are relative (%) of background value (level ozone only)
obchan	0	if > 0, selects the channel number. If <= zero, it will

		use all channels that pass qc in setuprad.
SUPEROB_RADAR		**Level 2 bufr file to radar wind superobs**
del_azimuth	5.0	azimuth range for superob box (default 5 degrees)
del_elev	0.25	elevation angle range for superob box (default .05 degrees)
del_range	5000.0	radial range for superob box (default 5 km)
del_time	0.5	1/2 time range for superob box (default .5 hours)
elev_angle_max	5.0	max elevation angle (default of 5 deg)
minnum	50	minimum number of samples needed to make a superob
range_max	100000.0	max radial range in meters to use in constructing superobs (default 100km)
l2superob_only	.false.	if true, then process level 2 data creating superobs, then quit. (added for easier retrospective testing, since level 2 bufr files are very large and hard to work with)

LAG_DATA		**Lagrangian data assimilation related variables**
lag_accur	1.0e-6	Accuracy used to decide whether or not a balloon is on the grid
infile_lag	inistate_lag.dat	File containing the initial position of the balloon
lag_stepduration	900.0	Duration of one time step for the propagation model
lag_nmax_bal	1000	Maximum number of balloons at starting time
lag_vorcore_stderr_a	2.0e3	Observation error for vorcore balloon
lag_vorcore_stderr_b	0.0	error = b + a*timestep(in hours)

HYBRID_ENSEMBLE		**Parameters for use with hybrid ensemble option**
l_hyb_ens	.false.	if true, then turn on hybrid ensemble option
uv_hyb_ens	.false.	if true, then ensemble perturbation wind variables are u,v, otherwise, ensemble perturbation wind variables are stream, pot. Functions.
q_hyb_ens	.false.	if true, then use specific humidity ensemble perturbations, otherwise, use relative humidity
aniso_a_en	.false.	if true, then use anisotropic localization of hybrid

		ensemble control variable a_en.
generate_ens	.true.	if true, then generate internal ensemble based on existing background error
n_ens	0	number of ensemble members.
nlon_ens	0	number of longitudes on ensemble grid (may be different from analysis grid nlon)
nlat_ens	0	number of latitudes on ensemble grid (may be different from analysis grid nlat)
jcap_ens	0	for global spectral model, spectral truncation
pseudo_hybens	.false.	if true, turn on pseudo ensemble hybrid for HWRF
merge_two_grid_ ensperts	.false.	if true, merge ensemble perturbations from two forecast domains to analysis domain (one way to deal with hybrid DA for HWRF moving nest)
regional_ensemble_ option	0	integer, used to select type of ensemble to read in for regional application. Currently takes values from 1 to 4 =1: use GEFS internally interpolated to ensemble grid. =2: ensembles are WRF NMM format =3: ensembles are ARW netcdf format. =4: ensembles are NEMS NMMB format.
full_ensemble	.false.	if true, first ensemble perturbation on first guess istead of on ens mean
betaflg	.false.	if true, use vertical weighting on beta1_inv and beta2_inv, for regional
coef_bw	0.9	fraction of weight given to the vertical boundaries when betaflg is true
pwgtflg	.false.	if true, use vertical integration function on ensemble contribution of Psfc
jcap_ens_test	0	for global spectral model, test spectral truncation (to test dual resolution)
beta1_inv	1	1/beta1, the default weight given to static background error covariance if (.not. readin_beta) 0 <= beta1_inv <= 1, tuned for optimal performance =1, then ensemble information turned off =0, then static background turned off the weights are applied per vertical level such that : betas_inv(:) = beta1_inv , vertically varying weights given to static B ; betae_inv(:) = 1 - beta1_inv , vertically varying weights given ensemble derived covariance. If (readin_beta) then betas_inv and betae_inv are read

		from a file and beta1_inv is not used.
s_ens_h	2828	homogeneous isotropic horizontal ensemble localization scale (km)
s_ens_v	30	vertical localization scale (grid units for now) s_ens_h, s_ens_v, and beta1_inv are tunable parameters.
use_gfs_ens	.true.	controls use of global ensemble: .t. use GFS (default); .f. uses user-defined ens
readin_ localization	.false.	flag to read (.true.)external localization information file
readin_beta	.false.	flag to read (.true.) the vertically varying beta parameters betas_inv and betae_inv from a file.
eqspace_ensgrid	.false.	if .true., then ensemble grid is equal spaced, staggered 1/2 grid unit off ploes. if .false., then gaussian grid assumed for ensemble (global only)
use_localization_grid	.false.	if true, then use extra lower res gaussian grid for horizontal localization (global runs only--allows possiblity for non-gaussian ensemble grid)
grid_ratio_ens	1	for regional runs, ratio of ensemble grid resolution to analysis grid resolution default value = 1 (dual resolution off)
oz_univ_static	.false.	if true, decouple ozone from other variables and defaults to static B (ozone only)
write_ens_sprd	.false.	writing global ensemble spread in byte addressable format for plotting with grads
enspreproc	.false.	flag to read(.true.) pre-processed ensemble data already
i_en_perts_io	0	flag to read in ensemble perturbations in ensemble grid. This is to speed up RAP/HRRR hybrid runs because the same ensemble perturbations are used in 6 cycles =0: No ensemble perturbations IO (default) =2: skip get_gefs_for_regional and read in ensemble perturbations from saved files.
l_ens_in_diff_time	.false.	if use ensembles that are available at different time from analysis time. =false: only ensembles available at analysis time can be used for hybrid. (default)

		=true: ensembles available time can be different from analysis time in hybrid analysis
rapidrefresh_cldsurf		**Options for cloud analysis and surface enhancement for RR appilcation**
dfi_radar_latent_heat_time_period	30.0	DFI forward integration window in minutes
metar_impact_radius	10.0	metar cloud observation impact radius in grid number
metar_impact_radius_lowCloud	4.0	impact radius for METAR cloud observation that indicate low cloud base
l_gsd_terrain_match_surfTobs	.false.	if .true., GSD terrain match for surface temperature observation
l_sfcobserror_ramp_t	.false.	namelist logical for adjusting surface temperature observation error
l_sfcobserror_ramp_q	.false.	namelist logical for adjusting surface moisture observation error
l_PBL_pseudo_SurfobsT	.false.	if .true. produce pseudo-obs in PBL layer based on surface obs T
l_PBL_pseudo_SurfobsQ	.false.	if .true. produce pseudo-obs in PBL layer based on surface obs Q
l_PBL_pseudo_SurfobsUV	.false.	if .true. produce pseudo-obs in PBL layer based on surface obs UV
pblH_ration	0.75	percent of the PBL height within which to add pseudo-obs
pps_press_incr	30hPa	pressure increase for each additional pseudo-obs on top of previous level
l_gsd_limit_ocean_q	.false.	if .true. do GSD limitation of Q over ocean
l_pw_hgt_adjust	.false.	if .true. do GSD PW adjustment for model vs. obs station height
l_limit_pw_innov	.false.	if .true. do GSD limitation of PW obs
max_innov_pct	0.1	sets limit of PW ob to a percent of the background value (0-1)
l_cleanSnow_WarmTs	.false.	if .true. do GSD limitation of using retrieved snow over warn area (Ts > r_cleanSnow_WarmTs_threshold)

l_conserve_thetaV	.false.	if .true. conserve thetaV during moisture adjustment in cloud analysis
r_cleanSnow_WarmTs _threshold	8.0	threshold for using retrieved snow over warn area
i_conserve_thetaV_ iternum	3	iteration number for conserving thetaV during moisture adjustment
l_gsd_soilTQ_nudge	.false.	if .true. do GSD GOES cloud building
l_cld_bld	.false.	if .true. do GSD soil T and Q nudging based on the lowest t analysis increment
cld_bld_hgt	1200m	sets limit below which GOES cloud building occurs
build_cloud_frac_p	0.95	sets the threshold for building clouds from satellite
clear_cloud_frac_p	0.1	sets the threshold for clearing clouds from satellite
nesdis_npts_rad	1	NESDIS cloud product impact radiu (grid points)
iclean_hydro_withRef	1	if =1, then clean hydrometeors if the grid point has no echo and maxref=0
iclean_hydro_withRef_allcol	0	if =1, then clean whole column hydrometeors if the observed max ref =0 and satellite cloud shows clean
l_use_2mq4b	0	background used for calculate surface moisture observation innovation =0 Use Q from the 1st model level. (default) =1 use 2m Q as part of background
i_use_2mt4b	0	background used for calculate surface temperature observation innovation =0 Use T from the 1st model level. (default) =1 use 2m T as part of background
i_gsdcldanal_type	0	options for how GSD cloud analysis should be conducted =0. no cloud analysis (default) =1. cloud analysis after var analysis =5. skip cloud analysis and NETCDF file update
i_gsdsfc_uselist	0	options for how to use surface observation use or rejection list =0 . EMC method (default) =1 . GSD method
i_lightpcp	0	options for how to deal with light precipitation =0 . don't add light precipitation (default) =1 . add light precipitation in warm section
i_sfct_gross	0	if use extended threshold for surface T gross check =0 use threshold from convinfo (default) =1 for cold surface, threshold for gross check is enlarged to bring more large negative innovation into analysis.

CHEM		**Chemistry data assimialtion**
berror_chem	.false.	if berror file is supplied for chemistry
oneobtest_chem	.false.	single observation test for chemistry
maginnov_chem	30.0	if oneobtest_chem=T magnitude of innovation for chemistry
magoberr_chem	2.0	if oneobtest_chem=T magnitude of observation error for chemistry
oneob_type_chem	pm2_5	if oneobtest_chem=T type of chemical observation
oblat_chem	45.0	if oneobtest_chem=T latitude of the observation
oblon_chem	270.0	if oneobtest_chem=T longitude of the observation
obpres_chem	1000.0	if oneobtest_chem=T pressure of the observation
diag_incr	.false.	if user wishes to output to a binary file increment
elev_tolerance	500.0	for surface chemical observation sometimes elevation (elev_obs) of the measurement is available (sometimes not).
tunable_error	0.5	tuning parameter to specify representativeness error for in-situ observations
in_fname	cmaq_input.bin	name of background file for cmaq
out_fname	cmaq_output.bin	name analysis file for cmaq
incr_fname	chem_increment.bin	if diag_incr=T name of the binary dump for pm2_5
laeroana_gocart	.false.	when true, do chem analysis with wrfchem and modis