GSI Gridpoint Statistical Interpolation

COMMUNITY VERSION 3.3

User's Guide

June 2014

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 first part of 2014 Gridpoint Statistical Interpolation (GSI) User's Guide geared particularly for beginners. It describes the fundamentals of using GSI Version 3.3, 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 second part of the User's Guide, *Advance GSI User's Guide*, released together with this document and the Version 3.3 code release.

This version of GSI was released on July 3, 2014. Its code and supplemental libraries and files are based on a revision of the GSI repository in April 2014.

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

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

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

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 Weather Agency 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 variational data assimilation (DA) system for both global and regional applications. It was initially developed by the National Oceanic and Atmospheric Administration (NOAA) National Centers for Environmental Prediction (NCEP) 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.

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 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) Earth System Laboratory's Mesoscale and Microscale Meteorology Division (MMM).

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 NCEP Environmental Modeling Center (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 intermediate bug fixes. 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. The committee is composed of primary GSI research and operational centers. Current members include NCEP/EMC, NASA/GMAO, NOAA/ESRL, NCAR/MMM, NOAA's Satellite and Information Service (NESDIS), Air Force Weather Agency (AFWA) and DTC. As one of the committee members, the DTC represents the research community with GSI users from all over the world.

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

Coordination and Advisory

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

GSI Code Review

- Establish and manage a unified GSI coding standard followed by all GSI developers.
- Establish and manage a process for proposal and commitment of new developments to the GSI 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 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 GSI Review Committee to avoid any potential conflict in code development. Upon the approval of the GSI review committee, the DTC will work with the prospective contributors in the preparation and integration of their code into the GSI 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 GSI Review Committee 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 2014 version 3.3 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 April 2014.

The GSI version 3.3 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). 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 3.3 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 3.3 versus version 3.2:

Major new features and enhancement:

- GSI interface to CRTM is updated to use CRTM V2.1.3:
  - Handling of potential GSI surface input inconsistencies.
  - New IR mapping
  - New microwave land emissivity
- Enhanced radiance bias correction
- FGAT capability for GFS ozone in regional model
- ARW background (binary and netcef): capability run on a grid space larger than background grid (GSI already has this capability for NMM background)

Observation specific:

- Added MLS Near Real-Time (NRT) ozone data assimilation
- Improvement of GSP RO data assimilation (for both refractivity and bending angle): particularly for super refreaction (SR) situation
- Added GOES hourly wind IR and cloud top wind assimilation
- Added water vapor cloud top satellite wind assimilation from EUMETSAT
- Added JMA profiler wind assimilation
- Improved SSMIS F18 bias correction: additional bias predictors
- Update to OSCAT data assimilation:

    - Data read-in from BUFR files, instead of using PrepBUFR files
    - QC modified
- Update to TDR data assimilation and added TDR super-observation assimialtion
- Added flight level and Stepped Frequency Miscrowave Radiometer (SFMR) High Density Observation (HDOB) data assimilation from reconnaissance aircaraft
- Added wind assimilation from tall towers
- Update to water vapor radiance assimilation

Application specific:

- NAM
    - Initial changes for development of hourly NAM/NDAS system
- NMMB
    - Use GSD's cloud analysis for radar reflectivity assimilation
- RAP
    - GSD aircraft temperature observation bias correction
    - Improvement to GSD cloud analysis
    - Improvement to soil temperature and moisture nudging

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.

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, Intel based Macintosh computers using the PGI compiler, and GNU's Gfortran compiler on Linux platforms.

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 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 ***comGSI v3.3 tarball*** to download the most recent version of the source code (as of June 2014). 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 other than Gfortran, use the big endian byte order coefficients found in the first CRTM link. When using the Gfortran compiler it is necessary to use the little endian byte order coefficients. These are available from the second CRTM link ***CRTM 2.1.3 Little_Endian coefficients*.**

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 3.3 comes in a tar file named `comGSI_v3.3.tar.gz`. The tar file may be unpacked by using the UNIX commands:

```
gunzip comGSI_v3.3.tar.gz
tar -xvf comGSI_v3.3.tar
```

This creates the top level GSI directory `comGSI_v3.3/`.

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" in this table, which is not included with the source code, is the WRF IO API's. Please note that only WRFV3.4.1, WRFV3.5, WRFV3.5.1 are tested with this current version of GSI.

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 vender 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 vender 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 vender 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 `comGSI_v3.3/` 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)
   2.  Linux x86_64, PGI compilers (pgf90 & pgcc) w/ Vender supplied MPI
   3.  Linux x86_64, PGI compilers (pgf90 & gcc)
   4.  Linux x86_64, PGI compilers (pgf90 & gcc) w/ Vender supplied MPI
   5.  Linux x86_64, GNU compilers (gfortran & gcc)
   6.  Linux x86_64, Intel compiler (ifort & icc)
   7.  Linux x86_64, Intel compiler (ifort & icc) w/ Vender supplied MPI
   8.  Linux x86_64, Intel/gnu compiler (ifort & gcc)
   9.  Linux x86_64, Intel/gnu compiler (ifort & gcc) w/ Vender supplied MPI

Enter selection [1-9] :
```

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 options for the default and vender supplied versions of MPI. This was added due to some computing hardware venders creating non-standard `mpif90` wrappers for their vender 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 "vender 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 6 or 8. The alternative options (7 & 9) are needed for certain platforms that have a vender 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 3. The alternative options (2 & 4) are needed for certain platforms that have a vender 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.

2.6 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 V3.4.1+

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

2.6.1 Compilers tested for release

Version 3.3 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 12.1.4, 12.1.5, 13.0.1, 13.1.1, 13.1.2	icc 12.1.4, 12.1.5, 13.0.1, 13.1.1, 13.1.2
Intel/gnu	ifort 12.1.4, 12.1.5, 13.0.1, 13.1.1, 13.1.2	gcc 4.4.5

PGI	pgf90 11.7,12.8, 12.10, 13.2, 13.3, 13.4, 13.8	pgcc 11.7,12.8, 12.10, 13.2, 13.3, 13.4, 13.8
PGI/gnu	pgf90 11.7,12.8, 12.10, 13.2, 13.3, 13.4, 13.8	gcc 4.4.5
Gfortran	gfortran 4.7.1, 4.7.2, 4.7.3	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 single cases
  2. GFS (e) was tested with a NCEP case
  3. RTMA (g) was tested with a simple 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
satwnd	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 all are located in a directory called *fix/* in the release package, except for CRTM coefficients.

Table 3.2 lists fixed file names 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
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
```

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 4 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
rap	Fix files for GSI RAP application

Please note released GSI 3.3 tar files doesn't include ./fix/global and ./fix/rtma for space saving. Please downloand comGSI_v3.3_fix_global.tar.gz if you need to run global case and comGSI_v3.3_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, three sample run scripts are available for different GSI applications:

- *comGSI_v3.3/run/**run_gsi.ksh*** for regional GSI
- *comGSI_v3.3/run/**run_gsi_global.ksh*** for global GSI (GFS)
- *comGSI_v3.3/util/RTMA/* ***run_gsi_rtma.ksh*** for RTMA GSI

We will introduce the regional run scripts in detail in the following sections and introduce the global and RTMA run scripts when we introduce the GSI global and RTMA applications in Advanced GSI User's 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. Check the definitions of required variables.
5. Generate a run directory for GSI (sometime called working or temporary directory).
6. Copy the GSI executable to the run directory.
7. 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.
8. Link observations to the run directory.
9. Link fixed files (statistic, control, and coefficient files) to the run directory.
10. Generate namelist for GSI.

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

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. 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`
		`#$ -S /bin/ksh`
		`#$ -N GSI_test`
		`#$ -cwd`
		`#$ -r y`
		`#$ -pe comp 64`
		`#$ -l h_rt=0:20:00`
		`#$ -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, and the GSI executable file.

```
####################################################
# 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=2011032212
  WORK_ROOT=./comGSI_v3.3/run/testgsi
  OBS_ROOT=./obs
  PREPBUFR=./obs/gdas1.t12z.prepbufr.nr
  BK_FILE=./wrfinput_d01_ARW_2011-03-22_12
  FIX_ROOT=./comGSI_v3.3/fix
  CRTM_ROOT=./CRTM_REL-2.1.3
  GSI_EXE=./comGSI_v3.3/run/gsi.exe
```

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)
# 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
```

Option `bk_core` indicates the specific WRF core used to create the background files and is used to specify the WRF core in the namelist. 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.

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=2011032212
  WORK_ROOT=./comGSI_v3.3/run/testgsi
  OBS_ROOT=./obs
  PREPBUFR=./obs/gdas1.t12z.prepbufr.nr
  BK_FILE=./wrfinput_d01_ARW_2011-03-22_12
  FIX_ROOT=./comGSI_v3.3/fix
  CRTM_ROOT=./CRTM_REL-2.1.3
  GSI_EXE=./comGSI_v3.3/run/gsi.exe

#-----------------------------------------------
# bk_core= which WRF core is used as background (NMM or ARW)
# 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
```

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 with 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.

```
ulimit_s=$(ulimit -s)
ulimit -s unlimited
##################################################################
# 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 `dfile` of the GSI namelist section `OBS_INPUT` in this example script. 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.

```
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
  else
    ANAVINFO=${FIX_ROOT}/anavinfo_arw_netcdf_glbe
  fi
else
  echo ' Use NAM background error covariance'
  BERROR=${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
  else
     ANAVINFO=${FIX_ROOT}/anavinfo_arw_netcdf
  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
```

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 two sample sets of the constants for different background error covariance options, one set is used in the GFS operations and one for the NAM operations.

```
##################################################################
# Set some parameters for use by the GSI executable and to build the namelist
echo " Build the namelist "

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,'
else
#   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,'
fi
```

```
     if [ ${bk_core} = NMM ] ; then
        bk_core_arw='.false.'
        bk_core_nmm='.true.'
     else
        bk_core_arw='.true.'
        bk_core_nmm='.false.'
     fi
```

The following large chunk 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 Advanced GSI User's Guide Appendix A.

```
# Build the GSI namelist on-the-fly
cat << EOF > gsiparm.anl
 &SETUP
   miter=2,niter(1)=10,niter(2)=10,
   write_diag(1)=.true.,write_diag(2)=.false.,write_diag(3)=.true.,
   gencode=78,qoption=2,
   factqmin=0.0,factqmax=0.0,
   ndat=87,iguess=-1,
   oneobtest=.false.,retrieval=.false.,
   nhr_assimilation=3,l_foto=.false.,
   use_pbl=.false.,
 /
 &GRIDOPTS
   JCAP=62,JCAP_B=62,NLAT=60,NLON=60,nsig=60,regional=.true.,
   wrf_nmm_regional=${bk_core_nmm},wrf_mass_regional=${bk_core_arw},
   diagnostic_reg=.false.,
   filled_grid=.false.,half_grid=.true.,netcdf=.true.,
 /
 &BKGERR
   vs=${vs_op}
   hzscl=${hzscl_op}
   bw=0.,fstat=.true.,
 /
 &ANBKGERR
 /
 &JCOPTS
 /
 &STRONGOPTS
 /
 &OBSQC
   dfact=0.75,dfact1=3.0,noiqc=.false.,c_varqc=0.02,vadfile='prepbufr',
 /
 &OBS_INPUT
dmesh(1)=120.0,dmesh(2)=60.0,dmesh(3)=60.0,dmesh(4)=60.0,dmesh(5)=120,time_window_max=1.5,
dfile(01)='prepbufr',  dtype(01)='ps',   dplat(01)=' ',    dsis(01)='ps',        dval(01)=1.0, dthin(01)=0, dsfcalc(01)=0,
dfile(02)='prepbufr'   dtype(02)='t',    dplat(02)=' ',    dsis(02)='t',         dval(02)=1.0, dthin(02)=0, dsfcalc(02)=0,

…( Please check Appendix B for complete list of this namelist section)

dfile(86)='gsndlbufr', dtype(86)='sndrd3',dplat(86)='g15', dsis(86)='sndrD3_g15', dval(86)=0.0, dthin(86)=1, dsfcalc(86)=0,
dfile(87)='gsndlbufr', dtype(87)='sndrd4',dplat(87)='g15', dsis(87)='sndrD4_g15', dval(87)=0.0, dthin(87)=1, dsfcalc(87)=0,

 /
&SUPEROB_RADAR
del_azimuth=5.,del_elev=.25,del_range=5000.,del_time=.5,elev_angle_max=5.,minnum=50,range_m
ax=100000.,
   l2superob_only=.false.,
 /
 &LAG_DATA
 /
 &HYBRID_ENSEMBLE
   l_hyb_ens=.false.,
 /
 &RAPIDREFRESH_CLDSURF
```

```
 /
 &CHEM
 /
 &SINGLEOB_TEST
   maginnov=1.0,magoberr=0.8,oneob_type='t',
   oblat=38.,oblon=279.,obpres=500.,obdattim=${ANAL_TIME},
   obhourset=0.,
 /

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, 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"
   for type in $listall; do
      count=0
      if [[ -f pe0000.${type}_${loop} ]]; then
         count=`ls pe*${type}_${loop}* | wc -l`
      fi
      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
```

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.ksh* but need to set

```
BYTE_ORDER=Little_Endian
```

Currently, GSI can only be compiled successfully with gfortran 3.7.1 and can only run with conventional observations.

3.3 GSI Analysis Result Files in Run Directory

After setup the GSI run script, it can be submitted just as other 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, AMSU-B, HIRS3, HIRS4, and MHS), and GPSRO data. The analysis time is 12Z 22 March 2011. 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.202   gsi.exe
amsubbufr                       fort.203   gsiparm.anl
anavinfo                        fort.204   hirs3bufr
berror_stats                    fort.205   hirs4bufr
convinfo                        fort.206   l2rwbufr
diag_amsua_n15_anl.2011032212   fort.207   list_run_directory
diag_amsua_n15_ges.2011032212   fort.208   mhsbufr
diag_amsua_n18_anl.2011032212   fort.209   ozinfo
diag_amsua_n18_ges.2011032212   fort.210   pcpbias_out
diag_conv_anl.2011032212        fort.211   pcpinfo
diag_conv_ges.2011032212        fort.212   prepbufr
diag_mhs_n18_anl.2011032212     fort.213   prepobs_prep.bufrtable
diag_mhs_n18_ges.2011032212     fort.214   satbias_angle
errtable                        fort.215   satbias_in
fit_p1.2011032212               fort.217   satbias_out
fit_q1.2011032212               fort.218   satinfo
fit_rad1.2011032212             fort.219   stdout
fit_t1.2011032212               fort.220   stdout.anl.2011032212
fit_w1.2011032212               fort.221   wrfanl.2011032212
fort.201                        gpsrobufr  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.2011032212/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.2011032212/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)_ (outer loop)`	Diagnostic files for conventional and satellite radiance 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.

`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:

`ndat`: number of observation variables (not observation types). This number should be consistent with the number of the observation variable lines in section `OBS_INPUT`. If adding a new observation variable, `ndat` must be incremented by one and one new line must be added in `OBS_INPUT`. Based on dimensions of the variables in `OBS_INPUT` (e.g., `dfile)`, the maximum value of `ndat` is 200 in this version.

`dfile(01)='prepbufr'`: 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(01)='ps'`: analysis variable name that GSI can read in and handle. As an example here, GSI will read all `ps` observations from the file `prepbufr`. Please note this name should be consistent with that used in the GSI code.

`dplat(01)`: sets up the observation platform for a certain observation, which observations will be read in from the file `dfile`.

`dsis(01)`: 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)=60.0,dmesh(4)=60.0,dmesh(5)=120,time_window_max=1.5,
  dfile(01)='prepbufr',  dtype(01)='ps',       dplat(01)=' ',    dsis(01)='ps',        dval(01)=1.0, dthin(01)=0

  dfile(12)='ssmirrbufr',dtype(12)='pcp_ssmi', dplat(12)='dmsp', dsis(12)='pcp_ssmi',  dval(12)=1.0, dthin(12)=-1
  dfile(13)='tmirrbufr', dtype(13)='pcp_tmi',  dplat(13)='trmm', dsis(13)='pcp_tmi',   dval(13)=1.0, dthin(13)=-1
  dfile(14)='sbuvbufr',  dtype(14)='sbuv2',    dplat(14)='n16',  dsis(14)='sbuv8_n16', dval(14)=1.0, dthin(14)=0
  dfile(15)='sbuvbufr',  dtype(15)='sbuv2',    dplat(15)='n17',  dsis(15)='sbuv8_n17', dval(15)=1.0, dthin(15)=0
  dfile(16)='sbuvbufr',  dtype(16)='sbuv2',    dplat(16)='n18',  dsis(16)='sbuv8_n18', dval(16)=1.0, dthin(16)=0
  dfile(17)='hirs2bufr', dtype(17)='hirs2',    dplat(17)='n14',  dsis(17)='hirs2_n14', dval(17)=6.0, dthin(17)=1
  dfile(18)='hirs3bufr', dtype(18)='hirs3',    dplat(18)='n16',  dsis(18)='hirs3_n16', dval(18)=0.0, dthin(18)=1,

  dfile(33)='airsbufr',  dtype(33)='amsua',    dplat(33)='aqua', dsis(33)='amsua_aqua',dval(33)=5.0, dthin(33)=2
  dfile(34)='amsubbufr', dtype(34)='amsub',    dplat(34)='n15',  dsis(34)='amsub_n15', dval(34)=3.0, dthin(34)=3
```

The two namelist variables that control the radiance data thinning are real array `dmesh` in the 1st line and the integer array `dthin` in the last 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 1 (ps from prepbufr): no thinning because `dthin(01)=0`
- Data type 12 (pcp_ssmi from dmsp): no thinning because `dthin(01)=-1`
- Data type 17 (hirs2 from NOAA-14): thinning in a 120 km grid because `dthin(17)=1` and `dmesh(1)=120`
- Data type 34 (AMSU-B from NOAA-15): thinning in a 60 km grid because `dthin(34)=3` and `dmesh(3)=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 2011032212. 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, AMSU-B, HIRS3, 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: 2011032212. 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  JUN 04,2014  16:58:11.758  155  WEN   2456813
```

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

```
 gsi_metguess_mod*init_:  2D-MET STATE VARIABLES:
 gsi_metguess_mod*init_:  3D-MET STATE VARIABLES:
 cw
 gsi_metguess_mod*init_: ALL MET STATE VARIABLES:
 cw
 state_vectors*init_anasv:  2D-STATE VARIABLES
 ps
 sst
 state_vectors*init_anasv:  3D-STATE VARIABLES
 u
 v
 tv
 tsen
 q
 oz
 cw
 p3d
 state_vectors*init_anasv: ALL STATE VARIABLES
 u
 v
 tv
 tsen
 q
 oz
 cw
 p3d
 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:  set IO server task to mype_io=           3
 INIT_IO:  reserve units lendian_in=          15  and lendian_out=          66
```

```
 for little endian i/o
```

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=   3.00000000000000
SETUP_4DVAR: winoff=   3.00000000000000
SETUP_4DVAR: hr_obsbin=   3.00000000000000
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 =   78.0000000000000     ,
FACTQMIN        =  0.000000000000000E+000,
FACTQMAX        =  0.000000000000000E+000,
FACTV   =   1.00000000000000     ,
DELTIM  =   1200.00000000000     ,
DTPHYS  =   3600.00000000000     ,
BIASCOR =  -1.00000000000000     ,
BCOPTION        =           1,
DIURNALBC       =  0.000000000000000E+000,
NDAT    =          87,
NITER   =           0, 2*10, 48*0,
NITER_NO_QC     = 51*1000000,
MITER   =           2,
QOPTION =           2,
NHR_ASSIMILATION        =           3,
MIN_OFFSET      =         180,
IOUT_ITER       =         220,
      ● ● ● ● ●

/
&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*, to *wrf_inou9* because we only ran 3DVAR with one background.

Next, the background fields for the analysis are read in and the maximum and minimum 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=        2011           3          22          12           0
          0
 dh1  =            3
rmse_var = SMOIS
ndim1 =            3
ordering = XYZ
staggering =  N/A
start_index =            1           1           1           0
end_index =          348         247           6           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 =
       348         247          50           0
 nlon,lat,sig_regional=         348         247          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         247          50           0
 k,znu(k)=           1  0.9990000
 k,znu(k)=           2  0.9960001
      ● ● ● ● ●
 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 =
       348         247          50           0
k,max,min,mid T=           1   310.8738       234.0430       280.8286
k,max,min,mid T=           2   311.2928       235.4056       280.9952
k,max,min,mid T=           3   311.4995       237.6656       281.3005
k,max,min,mid T=           4   312.2517       243.0390       282.1275
k,max,min,mid T=           5   312.9366       241.8669       283.7198
       ● ● ● ● ●
k,max,min,mid T=          48   702.1189       616.5657       651.2628
k,max,min,mid T=          49   770.3488       684.2148       718.2207
k,max,min,mid T=          50   837.6133       760.6921       792.9601
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
```

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
```

The following example demonstrates how the analysis domain is partitioned into sub-domains for each processor. 4 processors were used in this example (see Section 4.4 for more information):

```
general_DETER_SUBDOMAIN:  task,istart,jstart,ilat1,jlon1=     0      1      1    124    174
general_DETER_SUBDOMAIN:  task,istart,jstart,ilat1,jlon1=     1    125      1    123    174
general_DETER_SUBDOMAIN:  task,istart,jstart,ilat1,jlon1=     2      1    175    124    174
general_DETER_SUBDOMAIN:  task,istart,jstart,ilat1,jlon1=     3    125    175    123    174
```

Information on the horizontal dimensions of the sub-domains and set grid related variables:

```
  in general_sub2grid_create_info, k,kbegin,kend,nlevs_loc,nlevs_alloc=
          0           1          76          76          76
  in general_sub2grid_create_info, k,kbegin,kend,nlevs_loc,nlevs_alloc=
          1          77         152          76          76
  in general_sub2grid_create_info, k,kbegin,kend,nlevs_loc,nlevs_alloc=
          2         153         227          75          75
  in general_sub2grid_create_info, k,kbegin,kend,nlevs_loc,nlevs_alloc=
          3         228         302          75          75
 INIT_GRID_VARS:  number of threads           1
 INIT_GRID_VARS:  for thread            1  jtstart,jtstop =            1
        176
```


Information on using vertical levels from background fields to replace the number of vertical levels set in *anavinfo* and the start index of each state variable in the bundle array:

```
INIT_RAD_VARS:  ***WARNING*** mxlvs from the anavinfo file          30
 is different from that of the guess          50
 .  Resetting maxlvs to match NSIG from guess.
 Vars in Rad-Jacobian (dims)
 --------------------------
 tv                            0
 q                            50
 oz                          100
 u                           150
 v                           151
 sst                         152
```

Display the analysis and background file time (they should be the same):

```
 READ_wrf_mass_FILES:  analysis date,minutes         2011           3
          22          12           0    17472240
 READ_wrf_mass_FILES:  sigma guess file, nming2   0.000000000000000E+000
        2011           3          22          12           0    17472240
 READ_wrf_mass_FILES:  sigma fcst files used in analysis  :             3
   3.00000000000000                1
 READ_wrf_mass_FILES:  surface fcst files used in analysis:             3
   3.00000000000000                1
 GESINFO:  Guess    date is           12           3          22        2011
  0.000000000000000E+000
 GESINFO:  Analysis date is          2011           3          22          12
           0  2011032212   3.00000000000000
```

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         2011           3          22
          12
 RADAR_BUFR_READ_ALL:  NO RADARS KEPT IN radar_bufr_read_all,
 continue without level 2 data
```

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  1.000000  75.0000  0.00000  0  0.00000  0.00000 0
READ_CONVINFO: ps  120  0  1  3.00000  0  0  0  4.00000  3.00000  1.000000  4.00000  0.30E-03 0  0.00000  0.00000 0
READ_CONVINFO: ps  132  0 -1  3.00000  0  0  0  4.00000  3.00000  1.000000  4.00000  0.30E-03 0  0.00000  0.00000 0
          ● ● ● ● ●
READ_CONVINFO: t   120  0  1  3.00000  0  0  0  8.00000  5.60000  1.30000   8.00000  0.10E-05 0  0.00000  0.00000 0
READ_CONVINFO: t   126  0 -1  3.00000  0  0  0  8.00000  5.60000  1.30000   8.00000  0.10E-02 0  0.00000  0.00000 0
          ● ● ● ● ●
READ_CONVINFO: gps 440  0 -1  3.00000  0  0  0  10.00000 10.00000 1.000000 10.00000  0.00000  0  0.00000  0.00000 0
```

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

```
 READ_WRF_MASS_GUESS:  open lendian_in=          15  to file=sigf03
 READ_WRF_MASS_GUESS:  open lendian_in=          15  to file=sigf03
 gsi_metguess_mod*create_: alloc() for met-guess done
  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 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   2011032212 prepbufr ps
```

```
 read_obs_check: bufr file uv                  not available satwnd
 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 date is   2011032212 prepbufr q
 read_obs_check: bufr file date is   2011032212 prepbufr t
 read_obs_check: bufr file date is   2011032212 hirs3bufr hirs3     n17
 read_obs_check: bufr file airs      aqua       not available airsbufr
 read_obs_check: bufr file amsua     aqua       not available airsbufr
 read_obs_check: bufr file date is   2011032212 prepbufr uv
 read_obs_check: bufr file ssmi      f15        not available ssmitbufr
 read_obs_check: bufr file ssmis_las f16        not available ssmisbufr
 read_obs_check: bufr file sndrd1    g12        not available gsnd1bufr

 read_obs_check: bufr file date is   2011032212 amsuabufr amsua     n15
 read_obs_check: bufr file date is   2011032212 amsubbufr amsub     n17
 read_obs_check: bufr file amsub     n17        not available amsubbufrears

 data type hirs2_n14           not used in info file -- do not read file
 hirs2bufr
 data type hirs3_n16           not used in info file -- do not read file
 hirs3bufr

read_obs_check: bufr file sndrd2    g15        not available gsnd1bufr
```

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

```
  number of extra processors            2
 READ_OBS:  read   19 hirs3      hirs3_n17           using ntasks=   2   0   2
 READ_OBS:  read   21 hirs4      hirs4_metop-a       using ntasks=   2   2   2
 READ_OBS:  read   36 amsub      amsub_n17           using ntasks=   2   0   2
 READ_OBS:  read   37 mhs        mhs_n18             using ntasks=   2   2   2
 READ_OBS:  read   38 mhs        mhs_metop-a         using ntasks=   2   0   2
 READ_OBS:  read   67 mhs        mhs_n19             using ntasks=   2   2   2
 READ_OBS:  read    1 ps         ps                  using ntasks=   1   0   1
 READ_OBS:  read    2 t          t                   using ntasks=   1   1   1
 READ_OBS:  read    3 q          q                   using ntasks=   1   2   1
 READ_OBS:  read    4 pw         pw                  using ntasks=   1   3   1
 READ_OBS:  read    6 uv         uv                  using ntasks=   1   0   1
 READ_OBS:  read   10 sst        sst                 using ntasks=   1   1   1
 READ_OBS:  read   11 gps_ref    gps                 using ntasks=   1   2   1
 READ_OBS:  read   28 amsua      amsua_n15           using ntasks=   1   3   1
 READ_OBS:  read   31 amsua      amsua_n18           using ntasks=   1   0   1
 READ_OBS:  read   32 amsua      amsua_metop-a       using ntasks=   1   1   1
 READ_OBS:  read   65 hirs4      hirs4_n19           using ntasks=   1   2   1
 READ_OBS:  read   66 amsua      amsua_n19           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=
         247         348
 GETSFC: set nlat_sfc,nlon_sfc=         247         348
================================================================================
Status   Var          Mean                Min                 Max
sfcges2 FC10   1.000000000000E+00   1.000000000000E+00   1.000000000000E+00
sfcges2 SNOW   7.733149095057E+00   0.000000000000E+00   3.080491943359E+02
sfcges2 VFRC   5.299080975923E-02   0.000000000000E+00   8.883580780029E-01
sfcges2 SRGH   5.000000000008E-02   5.000000000000E-02   5.000000000000E-02
sfcges2 STMP   2.807239946831E+02   2.168921508789E+02   3.027479553223E+02
sfcges2 SMST   7.959968731205E-01   4.999999888241E-03   1.071443676949E+00
sfcges2 SST    2.807239944456E+02   2.168921356201E+02   3.027479553223E+02
sfcges2 VTYP   1.554879240542E+01   1.000000000000E+00   2.400000000000E+01
sfcges2 ISLI   4.321164316627E-01   0.000000000000E+00   2.000000000000E+00
sfcges2 STYP   1.156151984736E+01   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=    189563
ithin= 1 rmesh=120.000000 isfcalc= 1 ndata=     21736 ntask=  2
 READ_BUFRTOVS: file=hirs3bufr  type=hirs3      sis=hirs3_n17            nread=    202426
ithin= 1 rmesh=120.000000 isfcalc= 1 ndata=     23636 ntask=  2
 READ_BUFRTOVS: file=amsubbufr  type=amsub      sis=amsub_n17            nread=    213920
ithin= 3 rmesh= 60.000000 isfcalc= 1 ndata=      9163 ntask=  2

      ● ● ● ● ●

 w_rejectlist: wlistexist,nwrjs= F           0
 t_rejectlist: tlistexist,ntrjs= F           0
 t_day_rejectlist: t_day_listexist,ntrjs_day= F           0

      ● ● ● ● ●
 READ_PREPBUFR: file=prepbufr   type=q          sis=q                    nread=     28820
ithin= 0 rmesh=120.000000 isfcalc= 0 ndata=     23978 ntask=  1
 READ_GPS:      time outside window   -2.81666666666667       skip this report
 READ_GPS:      time outside window   -2.95000000000000       skip this report

      ● ● ● ● ●
 READ_PREPBUFR: file=prepbufr   type=uv         sis=uv                   nread=     91930
ithin= 0 rmesh=120.000000 isfcalc= 0 ndata=     72392 ntask=  1
 READ_BUFRTOVS: file=amsuabufr  type=amsua      sis=amsua_n18            nread=     63825
ithin= 2 rmesh= 60.000000 isfcalc= 1 ndata=     52309 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 63825 observations in the file (`nread=63825`) and 52309 in analysis domain and time-window (`ndata=52309`). 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                      2607      2878      9565      3019
OBS_PARA: t                       5172      4743     13902      5590
OBS_PARA: q                       4107      4197     11998      4090
OBS_PARA: pw                       296        92       475        83
OBS_PARA: uv                      6640      5439     18365      6147
OBS_PARA: sst                        0         0         6         3
OBS_PARA: gps_ref                 3538      5580      2277      6768
OBS_PARA: hirs3     n17              0         0       478       773
OBS_PARA: hirs4     metop-a          0         0       416       731
OBS_PARA: amsua     n15           2563      1323      1048      1669
OBS_PARA: amsua     n18           1002      2119         0       390
OBS_PARA: amsua     metop-a          0         0      1268      2279
OBS_PARA: amsub     n17              0         0      1716      2891
OBS_PARA: mhs       n18           1446      2932         0       809
OBS_PARA: mhs       metop-a          0         0      1600      2839
OBS_PARA: hirs4     n19            244      1093         0       236
OBS_PARA: amsua     n19            651      3486         0       469
OBS_PARA: mhs       n19            936      4272         0       848
```

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                            22
           0
SETUPALL:,obstype,isis,nreal,nchanl=t         t                             24
           0

      ● ● ● ● ●
SETUPALL:,obstype,isis,nreal,nchanl=gps_ref   gps                           16
           0
SETUPALL:,obstype,isis,nreal,nchanl=amsua     amsua_n15                     33
          15
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
SEcategory_ReadFile(INFORMATION) : FILE: ./NPOESS.IRland.EmisCoeff.bin; ^M
SEcategory RELEASE.VERSION:  3.01^M
  CLASSIFICATION: NPOESS,  N_FREQUENCIES=20  N_SURFACE_TYPES=20
IRwaterCoeff_ReadFile(INFORMATION) : FILE: ./Nalli.IRwater.EmisCoeff.bin; ^M
IRwaterCoeff RELEASE.VERSION:  3.02  N_ANGLES= 76  N_FREQUENCIES= 2223  N_WIND_SPEEDS= 11
      ● ● ● ● ●
```

In above section, when computing the radiance observation innovation, information on reading in CRTM coefficients is followed with 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):

```
      ● ● ● ● ●
 SETUPRAD:  write header record for mhs_n19                    7          30
```

```
          8           0           0          17           0       30303
  to file pe0000.mhs_n19_01   2011032212
```

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

Print Jo components (observation term for each observation type in cost function) at the beginning of the inner loop:

```
Begin Jo table outer loop
    Observation Type           Nobs                          Jo        Jo/n
surface pressure              15601    1.0956403515917838E+04       0.702
temperature                   11913    1.9101324500799088E+04       1.603
wind                          36496    3.8217841015360878E+04       1.047
moisture                       4416    5.8414245242861898E+03       1.323
gps                            8496    1.8501599503511450E+04       2.178
radiance                     122306    1.0639033391159725E+05       0.870
                               Nobs                          Jo        Jo/n
           Jo Global         199228    1.9900892697147268E+05       0.999
 End Jo table outer loop
```

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

```
GLBSOI:  START pcgsoi jiter=          1
Initial cost function =  1.9900892697147125E+05
Initial gradient norm =  5.898227357120081251E+03
cost,grad,step,b,step? =   1   0  1.9900892697147125E+05  5.898227357120081251E+03  7.97363704643945341O-04
0.0000000000000000E+00  good
cost,grad,step,b,step? =   1   1  1.7126937252196830 9E+05  5.4921161649248715 5E+03  7.1532285338799157 7E-04
8.6703456385534383 7E-01  good
cost,grad,step,b,step? =   1   2  1.4969284609783891 4E+05  4.0281281298064104 7E+03  5.6942251703125102 3E-04
5.3793168285733450 3E-01  good
cost,grad,step,b,step? =   1   3  1.4045350097918737 5E+05  2.8755965844713477 96E+03  1.2186404557771690 45E-03
5.0962340503181513 54E-01  good
cost,grad,step,b,step? =   1   4  1.3037649515183476 61E+05  2.4742936233302143 59E+03  1.3754410835763821 76E-03
7.4036614871819805 83E-01  good
cost,grad,step,b,step? =   1   5  1.2195586749643701 48E+05  1.6068096323034965 28E+03  2.1953593637808496 53E-03
4.2172212021440536 04E-01  good
cost,grad,step,b,step? =   1   6  1.1628780703581433 04E+05  1.3359106554643815 34E+03  4.9757660687473285 33E-03
6.9123540524183180 18E-01  good
cost,grad,step,b,step? =   1   7  1.0740776990071612 82E+05  1.1586433753547539 71E+03  5.0316113010702738 16E-03
7.5221976048946215 25E-01  good
cost,grad,step,b,step? =   1   8  1.0065306081198490 57E+05  1.0070463092588562 40E+03  5.0922106968860278 43E-03
7.5543885525218013 03E-01  good
cost,grad,step,b,step? =   1   9  9.5488834701660161 95E+04  7.6011561805079134 05E+02  7.2508842183631513 21E-03
5.6971863856841364 89E-01  good
cost,grad,step,b,step? =   1  10  9.1299449613895412 77E+04  8.7048882199406762 08E+02  4.4472718049348471 11E-03
1.3114963470485161 42E+00  good
```

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 (table starts with `wwww u`):

```
================================================================================
Status   Var          Mean                Min                 Max
analysis U      8.698952762569E+00  -5.999269779894E+01   6.930304217015E+01
analysis V     -8.299641535992E-01  -7.879347600058E+01   9.177475886417E+01
analysis TV     2.402428937497E+02   1.912660555983E+02   3.038786562805E+02
analysis Q      1.533003274421E-03   1.000000000000E-07   1.816572495337E-02
analysis TSEN   2.399786801503E+02   1.912657736485E+02   3.008116049854E+02
analysis OZ     0.000000000000E+00   0.000000000000E+00   0.000000000000E+00
analysis DUMY   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   4.097302066597E+01   1.148126437759E+00   1.038437446031E+02
analysis PS     9.917129413352E+01   6.406948766069E+01   1.039640599689E+02
```

```
 analysis SST    2.807239944456E+02   2.168921356201E+02   3.027479553223E+02
 analysis radb  -4.313680576355E-03  -3.641218651736E+01   3.125167437430E+01
 analysis pcpb   0.000000000000E+00   0.000000000000E+00   0.000000000000E+00
 analysis aftb   0.000000000000E+00   0.000000000000E+00   0.000000000000E+00
 ================================================================================
 wwww u          -1.581602534104E-02  -1.116169773420E+01   6.930467859312E+00
 wwww v           7.003916950589E-03  -1.072756598826E+01   1.018087611349E+01
 wwww tv         -2.428850872104E-01  -4.197930783056E+00   2.131571781880E+00
 wwww tsen       -2.431889490574E-01  -4.197923698213E+00   2.000890378949E+00
 wwww q           2.190067591869E-06  -3.004634726274E-03   3.747316017279E-03
 wwww oz          0.000000000000E+00   0.000000000000E+00   0.000000000000E+00
 wwww cw          0.000000000000E+00   0.000000000000E+00   0.000000000000E+00
 wwww p3d         9.638152608810E-03  -1.125491779875E-01   1.557171367235E-01
 wwww ps          2.355985673591E-02  -1.125491779875E-01   1.557171367235E-01
 wwww sst        -1.218178865623E-02  -2.311879802622E-01   4.190744577886E-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                          22
           0
 SETUPALL:,obstype,isis,nreal,nchanl=t         t                           24
           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=hirs4     hirs4_n19                   33
          19
 SETUPALL:,obstype,isis,nreal,nchanl=amsua     amsua_n19                   33
          15
 SETUPALL:,obstype,isis,nreal,nchanl=mhs       mhs_n19                     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 Jo components (observation term for each observation type in cost function) at the beginning of the inner loop:

```
 Begin Jo table outer loop
     Observation Type           Nobs                          Jo        Jo/n
 surface pressure              15616    6.333520503062793E+03       0.406
 temperature                   11913    1.2189697584303138E+04       1.023
 wind                          36698    2.5837350386172257E+04       0.704
 moisture                       4416    3.1306110794511401E+03       0.709
 gps                            8590    1.0742512590214315E+04       1.251
 radiance                     167157    9.0462388820009670E+04       0.541
                               Nobs                          Jo        Jo/n
            Jo Global         244390    1.4869608096321332E+05       0.608
  End Jo table outer loop
```

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

```
GLBSOI:  START pcgsoi jiter=           2
Initial cost function =  1.5159923106086160 99E+05
Initial gradient norm =  4.4322296246279220 21E+03
```

```
cost,grad,step,b,step? =   2   0  1.51599231060861609E+05  4.43222962462792202E+03  9.19252729034107131E-04
0.00000000000000000E+00  good
cost,grad,step,b,step? =   2   1  1.33540824254705017E+05  4.40189393744329936E+03  4.65200284359452520E-04
9.86358164687266337E-01  good
cost,grad,step,b,step? =   2   2  1.24526791750745833E+05  2.44094647237943536E+03  6.87270693526575386E-04
3.07494507998491095E-01  good
cost,grad,step,b,step? =   2   3  1.20431881978386416E+05  1.98578203564052023E+03  1.34213551355469632E-03
6.61830295655778996E-01  good
cost,grad,step,b,step? =   2   4  1.15139398350377625E+05  1.38891258779346162E+03  8.37922788203102455E-04
4.89200252872554697E-01  good
cost,grad,step,b,step? =   2   5  1.13522979786036899E+05  1.83463693236102403E+03  7.43570609860770104E-04
1.74481921703951981E+00  good
cost,grad,step,b,step? =   2   6  1.11020200918014845E+05  1.14679385526420378E+03  1.69786236823847197E-03
3.90724326058718873E-01  good
cost,grad,step,b,step? =   2   7  1.08787280392583998E+05  1.08436482545242324E+03  2.11918186569718140E-03
8.94087716950857203E-01  good
cost,grad,step,b,step? =   2   8  1.06295446595092304E+05  9.09312290150096032E+02  3.65519041117748355E-03
7.03194198313688900E-01  good
cost,grad,step,b,step? =   2   9  1.03273156639910055E+05  7.87448833744983062E+02  3.64095149761057678E-03
7.49926268267966267E-01  good
cost,grad,step,b,step? =   2  10  1.01015491216006237E+05  8.23178557658416025E+02  2.75062988275729704E-03
1.09280685438790015E+00  good
```

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      8.700885639493E+00  -6.023414847757E+01   6.949176803438E+01
analysis V     -8.273734858137E-01  -7.869117374193E+01   9.316586908929E+01
analysis TV     2.401218553674E+02   1.907887598805E+02   3.038982137212E+02
analysis Q      1.535774434957E-03   1.000000000000E-07   1.829311398617E-02
analysis TSEN   2.398571902083E+02   1.907884789448E+02   3.008301947635E+02
analysis OZ     0.000000000000E+00   0.000000000000E+00   0.000000000000E+00
analysis DUMY   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   4.098261598314E+01   1.148213293865E+00   1.038610972345E+02
analysis PS     9.917172994130E+01   6.406404008590E+01   1.039684675355E+02
analysis SST    2.807239944456E+02   2.168921356201E+02   3.027479553223E+02
analysis radb  -4.987271960242E-03  -3.591958787258E+01   3.705302496264E+01
analysis pcpb   0.000000000000E+00   0.000000000000E+00   0.000000000000E+00
analysis aftb   0.000000000000E+00   0.000000000000E+00   0.000000000000E+00

wwww u                  1.932876923959E-03  -2.395994508552E+00   3.385605823107E+00
wwww v                  2.590667785544E-03  -2.184295786292E+00   2.938095364211E+00
wwww tv                -1.210383823023E-01  -1.766803144032E+00   1.069839418916E+00
wwww tsen              -1.214904103511E-01  -1.766799595587E+00   1.069836711285E+00
wwww q                  2.771153856552E-06  -1.781791979464E-03   1.470353534801E-03
wwww oz                 0.000000000000E+00   0.000000000000E+00   0.000000000000E+00
wwww cw                 0.000000000000E+00   0.000000000000E+00   0.000000000000E+00
wwww p3d                1.782855455785E-04  -2.633255051837E-02   4.146261531356E-02
wwww ps                 4.358077820924E-04  -2.633255051837E-02   4.146261531356E-02
wwww sst               -6.591535614488E-03  -2.457554191636E-01   1.782356736455E-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)=  1.7957527E-02  3.2165379E-05
```

```
at 10.12 in wrwrfmassa,max,min(tempa)=  1.7957527E-02  0.0000000E+00
at 10.13 in wrwrfmassa,max,min(tempa)=  0.0000000E+00  0.0000000E+00
at 10.14 in wrwrfmassa,max,min(temp1)=  1.7957527E-02  3.2165379E-05
iy,m,d,h,m,s=        2011           3          22          12           0
          0
nlon,lat,sig_regional=         348         247          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=         348         247          50           0
p_top=   1000.000
rmse_var=MUB
ordering=XY
WrfType,WRF_REAL=         104         104
ndim1=           2
staggering= N/A
start_index=           1           1           1           0
end_index1=         348         247          50           0
max,min MUB=   99645.99       62950.09
max,min psfc=   103952.3       64038.40
max,min MU=   3952.305      -2029.906
rmse_var=MU
ordering=XY
WrfType,WRF_REAL=         104         104
ndim1=           2
staggering= N/A
start_index=           1           1           1           0
end_index1=         348         247          50           0
k,max,min,mid T=           1   311.0371       234.0961       280.6392
k,max,min,mid T=           2   311.4707       235.5774       280.7957


      ● ● ● ● ●
k,max,min,mid T=          49   765.8903       679.4603       707.0878
k,max,min,mid T=          50   832.5298       755.9112       779.3900

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                          22
           0
 SETUPALL:,obstype,isis,nreal,nchanl=t         t                           24
           0
 SETUPALL:,obstype,isis,nreal,nchanl=q         q                           25
           0


      ● ● ● ● ●
```

```
 SETUPRAD:  write header record for mhs_metop-a                    7          30
           8           0           0          17           0       30303
  to file pe0002.mhs_metop-a_03   2011032212
 SETUPALL:,obstype,isis,nreal,nchanl=mhs       mhs_n19                      33
           5
 SETUPRAD:  write header record for mhs_n19                        7          30
           8           0           0          17           0       30303
  to file pe0000.mhs_n19_03   2011032212
```

Print Jo components (observation term for each observation type) after the analysis, which shows the fit of the analysis results to the data if compared to the same section before the 1st outer loop:

```
Begin Jo table outer loop
    Observation Type           Nobs                          Jo        Jo/n
surface pressure              15618    5.9261033208622894E+03       0.379
temperature                   11913    1.0987921108543993E+04       0.922
wind                          36730    2.3800325985227551E+04       0.648
moisture                       4416    2.7541925367616777E+03       0.624
gps                            8604    9.9331074009793520E+03       1.154
radiance                     186521    6.8115580584421783E+04       0.365
                              Nobs                          Jo        Jo/n
           Jo Global         263802    1.2151723093679665E+05       0.461
 End Jo table outer loop
```

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:

```
        ENDING DATE-TIME    JUN 04,2014  17:01:22.233  155  WEN   2456813
        PROGRAM GSI_ANL HAS ENDED.
    * . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * .
```

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 run script (*run_gsi.ksh*):

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=45.,
oblon=260.,
obpres=500.,
obdattim= 2011032212,
obhourset=0.,
```

Note:

- Please check Appendix A in the Advanced 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.

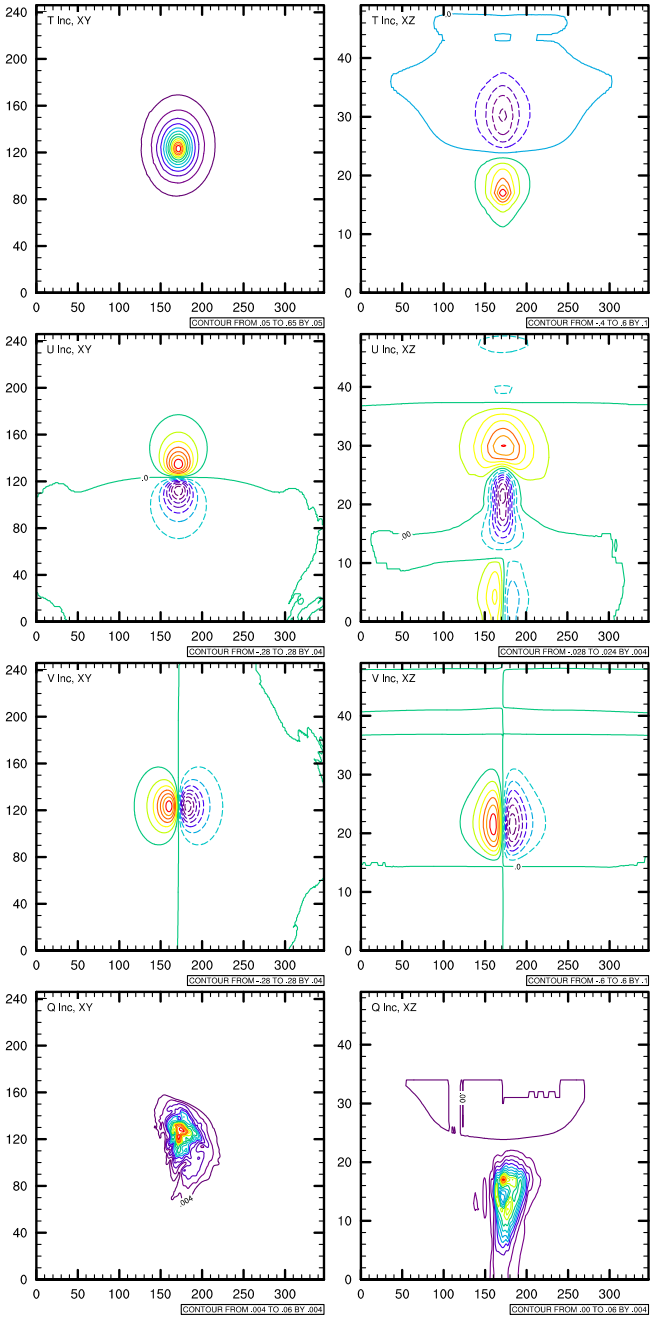


Fig. 4.1. Horizontal cross sections (left column) and vertical cross sections (right column) of analysis increment of T, U, V, and Q from a single T observation

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 run script), 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.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*
satwnd	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 run script), section &OBS_INPUT:

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

```
dfile(01)='prepbufr',  dtype(01)='ps',     dplat(01)=' ',    dsis(01)='ps',         dval(01)=1.0,  dthin(01)=0,
dfile(02)='prepbufr'   dtype(02)='t',      dplat(02)=' ',    dsis(02)='t',          dval(02)=1.0,  dthin(02)=0,
dfile(03)='prepbufr',  dtype(03)='q',      dplat(03)=' ',    dsis(03)='q',          dval(03)=1.0,  dthin(03)=0,
        ● ● ● ● ●
dfile(28)='amsuabufr', dtype(28)='amsua',  dplat(28)='n15',  dsis(28)='amsua_n15',  dval(28)=10.0, dthin(28)=2,
dfile(29)='amsuabufr', dtype(29)='amsua',  dplat(29)='n16',  dsis(29)='amsua_n16',  dval(29)=0.0,  dthin(29)=2,
        ● ● ● ● ●
```

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 -16 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:

```
dfile(28)='amsuabufr', dtype(28)='amsua',  dplat(28)='n15',  dsis(28)='amsua_n15',  dval(28)=10.0, dthin(28)=2,
```

Then, the AMSU-A observation from NOAA-15 will not be used in the analysis but the AMSU-A observations from NOAA-16 will still be used.

The observation filename in *dfile* can be different from the sample script (*run_gsi.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(28)*:

```
dfile(28)='amsuabufr_n15', dtype(28)='amsua',  dplat(28)='n15',  dsis(28)='amsua_n15',  dval(28)=10.0, dthin(28)=2,
dfile(29)='amsuabufr',     dtype(29)='amsua',  dplat(29)='n16',  dsis(29)='amsua_n16',  dval(29)=0.0,  dthin(29)=2,
```

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

```
# Link to the radiance data
ln -s ${OBS_ROOT}/le_gdas1.t12z.1bamua.tm00.bufr_d amsuabufr
ln -s ${OBS_ROOT}/le_gdas1.t12z.1bamua.tm00.bufr_d amsuabufr_n15
```

The GSI will read NOAA-16 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-16 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-16 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
tcp	112	0	1	3.0	0	0	0	75.0	5.0	1.0	75.0	0.000000	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
ps	132	0	-1	3.0	0	0	0	4.0	3.0	1.0	4.0	0.000300	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
ps	181	0	1	3.0	0	0	0	3.6	3.0	1.0	3.6	0.000300	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
ps	183	0	-1	3.0	0	0	0	4.0	3.0	1.0	4.0	0.000300	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
t	120	0	1	3.0	0	0	0	8.0	5.6	1.3	8.0	0.000001	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
t	130	0	1	3.0	0	0	0	7.0	5.6	1.3	7.0	0.001000	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
t	132	0	1	3.0	0	0	0	7.0	5.6	1.3	7.0	0.001000	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
t	134	0	-1	3.0	0	0	0	7.0	5.6	1.3	7.0	0.004000	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
t	180	0	1	3.0	0	0	0	7.0	5.6	1.3	7.0	0.004000	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
t	182	0	1	3.0	0	0	0	7.0	5.6	1.3	7.0	0.004000	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

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  icld_det
 amsua_n15               1   1    3.000     9.100    4.500   10.000    0.000      1
 amsua_n15               2   1    2.000    13.500    4.500   10.000    0.000      1
 amsua_n15               3   1    2.000     7.100    4.500   10.000    0.000      1
 amsua_n15               4   1    0.600     1.300    2.500   10.000    0.000      1
    ● ● ● ● ●
 amsua_n15              14  -1    2.000     1.400    4.500   10.000    0.000     -1
 amsua_n15              15   1    3.000    10.000    4.500   10.000    0.000      1
 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

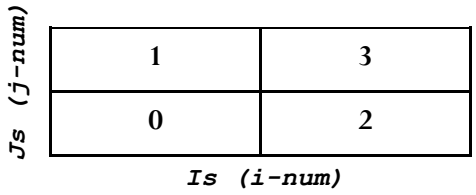
Column Name	Content of the column
sensor/instr/sat	Sensor, instrument, and satellite name
chan	Channel number for certain sensor
iuse	= 1, use this channel data =-1, don't use this channel data
error	Variance for each satellite channel
error_cld	Variance for each satellite channel if it is cloudy
ermax	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

In the standard output file (*stdout*), there is an information block that lists information regarding each sub-domain partition, including domain number (`task`), starting point (`istart,jstart`), and dimension of the sub-domain (`ilat1,jlon1`). Here is an example from the case we showed in Section 4.1, which used 4 cores in the analysis:

```
                                                    Task ID   Js     Is  j-num  i-num
general_DETER_SUBDOMAIN:  task,istart,jstart,ilat1,jlon1=     0      1      1    124    174
general_DETER_SUBDOMAIN:  task,istart,jstart,ilat1,jlon1=     1    125      1    123    174
general_DETER_SUBDOMAIN:  task,istart,jstart,ilat1,jlon1=     2      1    175    124    174
general_DETER_SUBDOMAIN:  task,istart,jstart,ilat1,jlon1=     3    125    175    123    174
```

The following diagram demonstrates how the domain is partitioned:



The standard output file (*stdout*) also 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, uv, and sst`), GPSRO (`gps_ref`), and radiance data (`amusa, amsub, hirs3/4,` and `mhs` from `Metop-a, NOAA 15, 17, 18,` and `19`) were distributed among 4 sub-domains:

```
        Observation type          …number of observations in each subdomain
 OBS_PARA: ps                        2607      2878      9565      3019
 OBS_PARA: t                         5172      4743     13902      5590
 OBS_PARA: q                         4107      4197     11998      4090
 OBS_PARA: pw                         296        92       475        83
 OBS_PARA: uv                        6640      5439     18365      6147
 OBS_PARA: sst                          0         0         6         3
 OBS_PARA: gps_ref                   3538      5580      2277      6768
 OBS_PARA: hirs3     n17                0         0       478       772
 OBS_PARA: hirs4     metop-a            0         0       416       731
```

```
OBS_PARA: amsua     n15            2563      1323      1048      1669
OBS_PARA: amsua     n18            1002      2119         0       390
OBS_PARA: amsua     metop-a           0         0      1268      2279
OBS_PARA: amsub     n17               0         0      1717      2891
OBS_PARA: mhs       n18            1446      2932         0       809
OBS_PARA: mhs       metop-a           0         0      1600      2839
OBS_PARA: hirs4     n19             244      1093         0       235
OBS_PARA: amsua     n19             651      3486         0       469
OBS_PARA: mhs       n19             935      4273         0       848
```

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 each 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 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 pcp_ssmi, pcp_tmi	
fort.209	fit of radar radial wind (rw)	
fort.210	fit of lidar wind (dw)	
fort.211	fit of super wind data (srw)	
fort.212	fit of GPS data	fractional difference
fort.213	fit of conventional sst data	C
fort.214	Tropical cyclone central pressure	
fort.215	Lagrangian data	

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*)

```
pressure levels (hPa)=   0.0 2000.0
     it     obs    type stype    count      bias       rms      cpen     qcpen
 o-g 01      ps     120 0000      141    0.2326    1.0637    1.3581    1.3516
 o-g 01      ps     180 0000     2210    0.2623    1.1016    1.4985    1.2942
 o-g 01      ps     181 0000      726    0.1441    1.2444    1.0102    1.0006
 o-g 01      ps     187 0000    12524    0.5001    1.0227    0.5366    0.5003
 o-g 01             all         15601    0.4474    1.0458    0.7023    0.6438
 o-g 01      ps rej 120 0000        2   -7.5859    8.7800    0.0000    0.0000
 o-g 01      ps rej 180 0000       16    0.4209    7.0383    0.0000    0.0000
 o-g 01      ps rej 181 0000       47    0.5775   58.2850    0.0000    0.0000
 o-g 01      ps rej 183 0000        6  -19.2918   19.5258    0.0000    0.0000
 o-g 01      ps rej 187 0000       38   -2.2101    7.4631    0.0000    0.0000
 o-g 01         rej all           109   -1.6608   38.9090    0.0000    0.0000
 o-g 01      ps mon 120 0000        2    0.5141    0.5372    0.3923    0.3923
 o-g 01      ps mon 180 0000      115   -0.1094    1.4396    5.4818    4.5090
 o-g 01      ps mon 181 0000      258    0.3884    1.2951    2.2811    2.2316
 o-g 01      ps mon 183 0000     1378   -0.8872    1.9598    0.0000    0.0000
 o-g 01      ps mon 187 0000      245   -0.2667    1.2678    3.5281    2.6850
 o-g 01         mon all          1998   -0.6002    1.7839    1.0431    0.8773
```

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     135    428    427    838    688    978   8061
 o-g 01      uv     220 0000  bias   -0.43   0.93   0.85   0.61   0.31   0.23   0.64
 o-g 01      uv     220 0000   rms    3.62   3.68   4.21   4.13   5.61   5.33   4.84
 o-g 01      uv     220 0000  cpen    0.70   0.95   1.26   1.33   1.39   1.30   1.25
 o-g 01      uv     220 0000 qcpen    0.70   0.95   1.26   1.33   1.38   1.29   1.25
 o-g 01      uv     223 0000 count       0     21    122    594    326     93   3176
 o-g 01      uv     223 0000  bias    0.00  -1.23   0.38   0.16  -0.00   0.92  -0.39
 o-g 01      uv     223 0000   rms    0.00   5.47   4.94   4.31   5.56   5.66   4.84
 o-g 01      uv     223 0000  cpen    0.00   2.39   1.86   1.24   1.27   1.06   1.29
       • • • • •
 o-g 01             all       count    1962   1398   1440   2623   1014   1071  18248
 o-g 01             all        bias   -0.31   0.93   0.59   0.05   0.21   0.29   0.28
 o-g 01             all         rms    3.02   3.87   4.34   4.35   5.59   5.36   4.67
 o-g 01             all        cpen    0.50   0.68   1.07   1.12   1.35   1.28   1.05
 o-g 01      uv rej 220 0000 count       0      0      0      0      0      0    292
 o-g 01      uv rej 220 0000  bias    0.00   0.00   0.00   0.00   0.00   0.00   2.81
 o-g 01      uv rej 220 0000   rms    0.00   0.00   0.00   0.00   0.00   0.00   8.69
 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         rej all       count      77     22    139    366      2      1   1089
 o-g 01         rej all        bias    5.58   2.18  -3.75 -10.98  24.83  28.94  -5.73
 o-g 01         rej all         rms   12.51  10.85  11.12  18.02  61.69  59.27  18.49
 o-g 01         rej all        cpen    0.00   0.00   0.00   0.00   0.00   0.00   0.00
 o-g 01      uv mon 220 0000 count       3      2      5      3     11     40    145
 o-g 01      uv mon 220 0000  bias   -1.55  -2.31   7.31   0.04  -3.49   2.02   1.74
 o-g 01      uv mon 220 0000   rms    3.92   3.68  15.18   2.94  10.65   8.67   8.93
 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         mon all       count    4651   9610   1570    676     51     63  16859
 o-g 01         mon all        bias   -0.78  -0.25   0.53  -0.79  -0.64   0.15  -0.38
 o-g 01         mon all         rms    2.88   3.22   5.71   8.18  19.81  12.48   4.24
 o-g 01         mon all        cpen    0.88   0.93   0.65   0.64   0.00   0.00   0.86
       • • • • •
```

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_w1.2011032212*) 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. Also, the "*qcpen*" lines have been deleted because they have the same values as "*cpen*" lines. Otherwise, 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     135    428    427    838    688    978    8061
 o-g 01      uv     220 0000  bias   -0.43   0.93   0.85   0.61   0.31   0.23    0.64
 o-g 01      uv     220 0000   rms    3.62   3.68   4.21   4.13   5.61   5.33    4.84
 o-g 01      uv     220 0000  cpen    0.70   0.95   1.26   1.33   1.39   1.30    1.25

 o-g 01      uv rej 220 0000 count       0      0      0      0      0      0     292
 o-g 01      uv rej 220 0000  bias    0.00   0.00   0.00   0.00   0.00   0.00    2.81
 o-g 01      uv rej 220 0000   rms    0.00   0.00   0.00   0.00   0.00   0.00    8.69
 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 mon 220 0000 count       3      2      5      3     11     40     145
 o-g 01      uv mon 220 0000  bias   -1.55  -2.31   7.31   0.04  -3.49   2.02    1.74
 o-g 01      uv mon 220 0000   rms    3.92   3.68  15.18   2.94  10.65   8.67    8.93
 o-g 01      uv mon 220 0000  cpen    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     135    428    427    838    688    978    8061
 o-g 02      uv     220 0000  bias   -0.27   0.87   0.64   0.66   0.25   0.56    0.69
 o-g 02      uv     220 0000   rms    3.29   3.26   3.43   3.28   5.15   5.08    4.31
 o-g 02      uv     220 0000  cpen    0.56   0.74   0.82   0.84   1.18   1.17    0.96

 o-g 02      uv rej 220 0000 count       0      0      0      0      0      0     292
 o-g 02      uv rej 220 0000  bias    0.00   0.00   0.00   0.00   0.00   0.00    3.22
 o-g 02      uv rej 220 0000   rms    0.00   0.00   0.00   0.00   0.00   0.00    9.05
 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 mon 220 0000 count       3      2      5      3     11     40     145
 o-g 02      uv mon 220 0000  bias   -1.13  -1.88   7.45  -0.03  -4.08   2.47    1.96
 o-g 02      uv mon 220 0000   rms    3.47   3.16  15.29   2.78  10.90   8.76    8.95
 o-g 02      uv mon 220 0000  cpen    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     135    428    427    838    688    978    8061
 o-g 03      uv     220 0000  bias   -0.27   0.84   0.60   0.63   0.20   0.57    0.67
 o-g 03      uv     220 0000   rms    3.24   3.11   3.24   3.07   5.01   5.00    4.17
 o-g 03      uv     220 0000  cpen    0.54   0.67   0.73   0.74   1.12   1.14    0.89

 o-g 03      uv rej 220 0000 count       0      0      0      0      0      0     292
 o-g 03      uv rej 220 0000  bias    0.00   0.00   0.00   0.00   0.00   0.00    3.20
 o-g 03      uv rej 220 0000   rms    0.00   0.00   0.00   0.00   0.00   0.00    8.93
 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 mon 220 0000 count       3      2      5      3     11     40     145
 o-g 03      uv mon 220 0000  bias   -0.82  -1.52   7.58  -0.04  -4.25   2.49    1.99
 o-g 03      uv mon 220 0000   rms    3.08   2.70  15.37   2.71  11.02   9.02    9.08
 o-g 03      uv mon 220 0000  cpen    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 8061 sounding observations used in the analysis. Among them, 135 are within the 1000-1200 hPa layer. Also from the "*count*" lines, in the rejection and monitoring section, there are 292 observations rejected and 145 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 0.64 and 4.84. 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    3.62   3.68   4.21   4.13   5.61   5.33    4.84
o-g 02      uv     220 0000   rms    3.29   3.26   3.43   3.28   5.15   5.08    4.31
o-g 03      uv     220 0000   rms    3.24   3.11   3.24   3.07   5.01   5.00    4.17
```

These three lines show that the rms reduced from 4.84 (o-g 01, which is O-B) to 4.31 (o-g 02, which is O-A after 1st outer loop) and then to 4.17 (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 of the file *satinfo*, which is the info file to control the data usage for radiance data. This portion begins with the line:

```
RADINFO_READ:  jpch_rad=   2680
```

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 mass bias correction, which begins with the line:

```
RADINFO_READ:  guess air mass bias correction coefficients below
```

Each channel has 5 coefficients listed in a line. Therefore, there are 2680 lines of 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        4931.80039389  1144      20     132      17     227    1881        0
                           qcpenalty     qc1     qc2     qc3     qc4     qc5     qc6     qc7
                        4931.80039389    26     986    1376   12729       0       0       0

 sat       type             penalty    nobs   iland isnoice  icoast ireduce   ivarl nlgross
n15        amsua        45673.97099101  6523     981    1705    1958    1555   44329        0
                           qcpenalty     qc1     qc2     qc3     qc4     qc5     qc6     qc7
                        45673.97099101  2447     138     304     171       0       6       6
```

```
  rad total   penalty_all=   106390.333911597088
  rad total qcpenalty_all=   106390.333911597088
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 2716   711   3.000   0.5630801   0.9124044   0.2095496   2.1696594   1.9684867
 2   2 amsua_n15 2656   773   2.000   0.1456301   1.0357824   0.2653187   2.2443727   1.9910710
 3   3 amsua_n15 3262   164   2.000   1.1713726  -1.2546513   0.3092257   2.0114653   1.5722096
 4   4 amsua_n15 3426   0     0.600  -0.2994171  -0.2676203   0.5595584   0.5738303   0.5076028
 5   5 amsua_n15 3426   10    0.300  -0.1874673  -0.2278419   0.8805730   0.3347520   0.2452487
 6   6 amsua_n15 6016   30    0.230  -1.5989361  -0.0059940   0.9904155   0.2467110   0.2466381
 7   7 amsua_n15 4009   2514  0.250  -0.5536621  -0.4306837   3.0917986   0.4714137   0.1916831
 8   8 amsua_n15 3257   3266  0.275  -0.6704360  -0.5495220   3.6424814   0.5737176   0.1648560
 9   9 amsua_n15 1472   5051  0.340  -1.2866839  -0.8251637   4.8293449   0.8342111   0.1225283
10  10 amsua_n15   77   6446  0.400  -0.9223207  -0.9628485   5.1222374   0.9669377   0.0888327
12  12 amsua_n15 1785   4738  1.000   1.2872243   0.0824760   0.0641187   0.4914450   0.4844749
15  15 amsua_n1  2371   1057  3.000   1.5901494  -1.2085643   0.2627504   2.4249496   2.1023209
      ● ● ● ● ●
61   2 hirs4_metop-a 944 200  0.600   0.4353536   0.3168343   0.7456275   0.5993171   0.5087210
62   3 hirs4_metop-a 1013 88  0.530   0.3471229   0.2084463   0.7542553   0.5073973   0.4626037
63   4 hirs4_metop-a 762  25  0.400  -0.1958993  -0.2501468   0.9649993   0.4085868   0.3230630
64   5 hirs4_metop-a 123  51  0.360  -0.4120880  -0.6791958   3.5627036   0.7121448   0.2141105
65   6 hirs4_metop-a 132  20  0.460  -0.7980088  -0.9226707   4.0302747   0.9651700   0.2832523
66   7 hirs4_metop-a 124   7  0.570  -0.8869851  -0.8847628   2.2090039   0.9848432   0.4325632
      ● ● ● ● ●
```

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	(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  n14       hirs2         0       0        0   0.0000    0.0000   0.0000   0.0000
o-g 01 rad  n16       hirs3         0       0        0   0.0000    0.0000   0.0000   0.0000
o-g 01 rad  n17       hirs3    202426   23636        0   0.0000    0.0000   0.0000   0.0000
o-g 01 rad  n18       hirs4         0       0        0   0.0000    0.0000   0.0000   0.0000
o-g 01 rad  metop-a   hirs4    189563   21736     4708   4931.8    4931.8   1.0475   1.0475
o-g 01 rad  g11       sndr          0       0        0   0.0000    0.0000   0.0000   0.0000
o-g 01 rad  g12       sndr          0       0        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  n14       msu           0       0        0   0.0000    0.0000   0.0000   0.0000
o-g 01 rad  n15       amsua    128055   84768    34473   45674.    45674.   1.3249   1.3249
```

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 (n15=NOAA-15)
instrument	instrument name (AMSU-A)
# 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
  Initial cost function =  1.990089269714727125E+05
  Initial gradient norm =  5.898227357120081251E+03
  cost,grad,step,b,step? =   1   0  1.990089269714727125E+05  5.898227357120081251E+03
7.973637046439453410E-04  0.000000000000000000E+00  good
  cost,grad,step,b,step? =   1   1  1.712693725121968309E+05  5.492116164924871555E+03
7.153228533879915727E-04  8.670345638553438317E-01  good
  cost,grad,step,b,step? =   1   2  1.496928460978389194E+05  4.028128129806410470E+03
5.694225170312510213E-04  5.379316828573345033E-01  good
        ● ● ● ● ●
  cost,grad,step,b,step? =   1   9  9.548883470166016195E+04  7.601156180507913405E+02
7.250884218363151321E-03  5.697186385684136489E-01  good
  cost,grad,step,b,step? =   1  10  9.129944961389541277E+04  8.704888219940676208E+02
4.447271804934847111E-03  1.311496347048516142E+00  good
```

The following are the iterations from the second outer loop:

```
Initial cost function =  1.515992310608616099E+05
Initial gradient norm =  4.432229624627922021E+03
cost,grad,step,b,step? =   2   0  1.515992310608616099E+05  4.432229624627922021E+03
9.192527290341071314E-04  0.000000000000000000E+00  good
cost,grad,step,b,step? =   2   1  1.335408242547050177E+05  4.401893937443299365E+03
4.652002843594525206E-04  9.863581646872663367E-01  good
cost,grad,step,b,step? =   2   2  1.245267917507458333E+05  2.440946472379435363E+03
6.872706935265753860E-04  3.074945079984910956E-01  good
        ● ● ● ● ●
cost,grad,step,b,step? =   2   9  1.032731566399100557E+05  7.874488337449830624E+02
3.640951497610576786E-03  7.499262682679662673E-01  good
cost,grad,step,b,step? =   2  10  1.010154912160062377E+05  8.231785576584160253E+02
2.750629882757297043E-03  1.092806854387900151E+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 `1.990089269714727125E+05` to `9.129944961389541277E+04` in the1st outerloop and reduced from `1.515992310608616099E+05` to `1.010154912160062377E+05` 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:

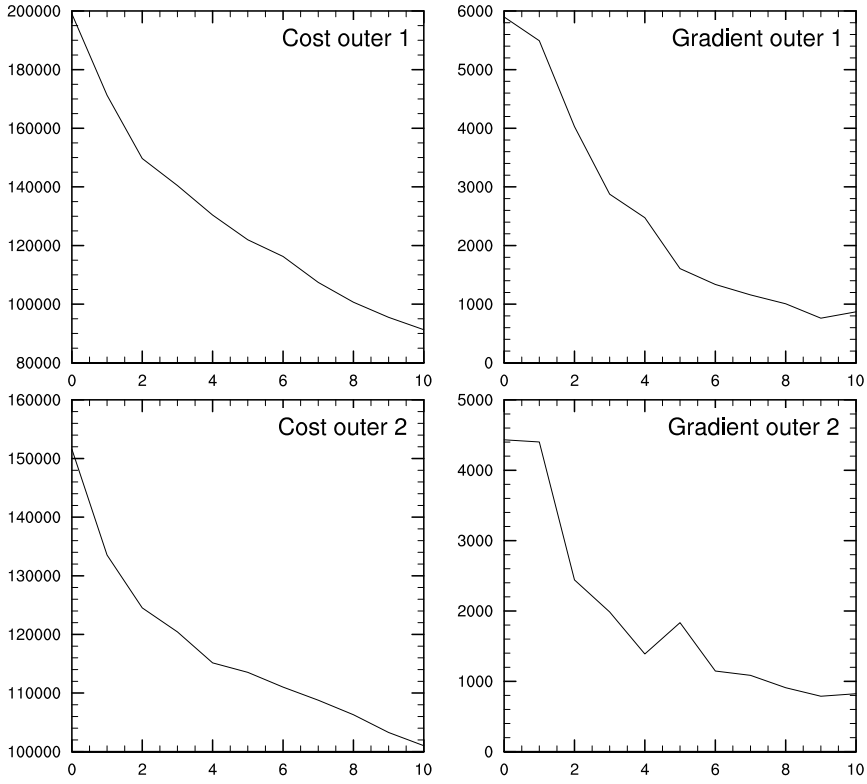


Fig.4.2 Evolution of cost function (left column) and the norm of gradient (right column) in the first outer loop (top raw) and the second outer loop (bottom raw)

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 ~/*comGSI_v3.3/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 0.12671E+01 0.56103E+00 0.10000E+10 0.68115E+00 0.10000E+10
      0.10500E+04 0.13302E+01 0.63026E+00 0.10000E+10 0.68115E+00 0.10000E+10
      0.10000E+04 0.14017E+01 0.73388E+00 0.10000E+10 0.68115E+00 0.10000E+10
      0.95000E+03 0.14543E+01 0.86305E+00 0.10000E+10 0.71307E+00 0.10000E+10
      0.90000E+03 0.14553E+01 0.99672E+00 0.10000E+10 0.74576E+00 0.10000E+10
      0.85000E+03 0.13865E+01 0.11210E+01 0.10000E+10 0.77845E+00 0.10000E+10

      220 OBSERVATION TYPE
      0.11000E+04 0.10000E+10 0.10000E+10 0.17721E+01 0.10000E+10 0.10000E+10
      0.10500E+04 0.10000E+10 0.10000E+10 0.20338E+01 0.10000E+10 0.10000E+10
      0.10000E+04 0.10000E+10 0.10000E+10 0.22927E+01 0.10000E+10 0.10000E+10
      0.95000E+03 0.10000E+10 0.10000E+10 0.24559E+01 0.10000E+10 0.10000E+10
      0.90000E+03 0.10000E+10 0.10000E+10 0.25377E+01 0.10000E+10 0.10000E+10
      0.85000E+03 0.10000E+10 0.10000E+10 0.25705E+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 three columns:

!otype	type	sub	iuse	twindow	numgrp	ngroup	nmiter	**gross**	**ermax**	**ermin**	var_b	var_pg	ithin
ps	120	0	1	3.0	0	0	0	**4.0**	**3.0**	**1.0**	4.0	0.000300	0
ps	180	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 *~/comGSI_v3.3/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  JUN 04,2014  16:58:11.758  155  WEN   2456813

       • • • • •

     ENDING DATE-TIME    JUN 04,2014  17:01:22.233  155  WEN   2456813
     PROGRAM GSI_ANL HAS ENDED.
* . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * .
```

This tells us this case started at `16:58:11.758` and ended at `17:01:22.233`. Meaning GSI used 3 minutes and 10.5 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                        = 190.488858
The total amount of time in user mode                = 183.787060
The total amount of time in sys mode                 = 3.763427
The maximum resident set size (KB)                   = 2388128
Number of page faults without I/O activity           = 155262
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                 = 7723
Number of InVoluntary Context Switches               = 229
****************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 3.2) 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_2011-03-22_12:00:00
   - The horizontal grid spacing is 30-km with 51 vertical sigma levels

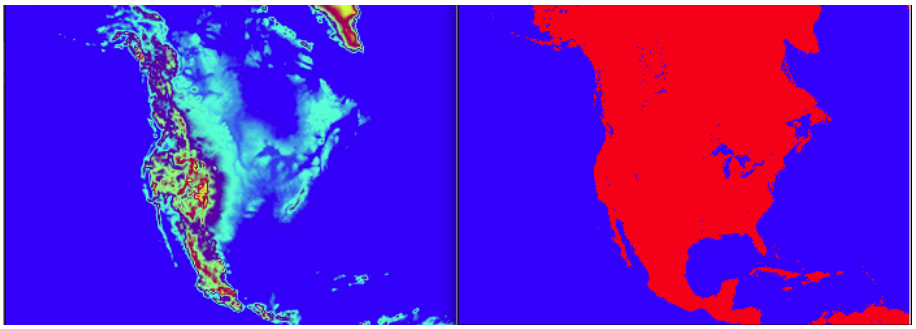


Figure 5.1: The terrain (left) and land mask (right) of the background used in this case study

2. Conventional data: NAM PrepBUFR data from 12UTC 22 March 2011.
   - File: *nam.t12z.prepbufr.tm00.nr*

3. Radiance and GPS RO data: GDAS PREPBUFR data from 12 UTC 22 March 2011
   - Files: *gdas.t12z.1bamua.tm00.bufr_d*
     gdas.t12z.1bamub.tm00.bufr_d
     gdas.t12z.1bhrs4.tm00.bufr_d
     gdas.t12z.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:

/scratch1/NA30km/bk

all the observations are located at:

/scratch1/NA30km/obs20110322

and the GSI release version 3.3 is located at

/scratch1/comGSI_v3.3

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 *./comGSI_v3.3* to run the GSI using the sample script *run_gsi.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.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=2011032212
```

  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=/scratch1/gsiprd_${ANAL_TIME}_prepbufr
```

  Set path to the background file:

```
BK_FILE=/scratch1/NA30km/bk/wrfinput_d01_2011-03-22_12:00:00
```

  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=/scratch1/NA30km/obs20110322
PREPBUFR=${OBS_ROOT}/nam.t12z.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=/scratch1/crtm/CRTM_REL-2.1.3
FIX_ROOT=/scratch1/comGSI_v3.3/fix
GSI_EXE=/scratch1/comGSI_v3.3/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, GSI can be run through the run script. On our test machine, the GSI run is submitted as follows:

```
$ bsub < run_gsi.ksh
```

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

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

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

A directory named *gsiprd_2011032212_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
Initial cost function =  6.909058019829032128E+04
Initial gradient norm =  7.929268252729451660E+02
cost,grad,step,b,step? =   1   0  6.909058019829032128E+04  7.929268252729451660E+02
2.019794516087224795E-02  0.000000000000000000E+00  good
cost,grad,step,b,step? =   1   1  5.639146654856126406E+04  5.784092536538506693E+02
2.127082220424394987E-02  5.321134586410080081E-01  good
cost,grad,step,b,step? =   1   2  4.927515845372552576E+04  5.045660928301126660E+02
1.903121055877515441E-02  7.609667129861801271E-01  good
```

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                  fit_w1.2011032212  fort.210  fort.221               satbias_angle
berror_stats              fort.201           fort.211  gsi.exe                satbias_in
convinfo                  fort.202           fort.212  gsiparm.anl            satbias_out
diag_conv_anl.2011032212  fort.203           fort.213  l2rwbufr               satinfo
diag_conv_ges.2011032212  fort.204           fort.214  list_run_directory     stdout
errtable                  fort.205           fort.215  ozinfo               stdout.anl.2011032212
fit_p1.2011032212         fort.206           fort.217  pcpbias_out            wrfanl.2011032212
fit_q1.2011032212         fort.207           fort.218  pcpinfo                wrf_inout
fit_rad1.2011032212       fort.208           fort.219  prepbufr
fit_t1.2011032212         fort.209           fort.220  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 avaninfo and namelist
   The following lines show GSI started normally and has read in the anavinfo and namelist:

```
 state_vectors*init_anasv:  2D-STATE VARIABLES ps          sst
 state_vectors*init_anasv:  3D-STATE VARIABLES u           v
 tv          tsen        q           oz          cw          p3d
 state_vectors*init_anasv: ALL STATE VARIABLES u           v
 tv          tsen        q           oz          cw          p3d
 ps          sst
  ... ...
 GSI_4DVAR:  nobs_bins =            1
 SETUP_4DVAR: l4dvar= F
 SETUP_4DVAR: l4densvar= F
 SETUP_4DVAR: winlen=   3.00000000000000
 SETUP_4DVAR: winoff=   3.00000000000000
 SETUP_4DVAR: hr_obsbin=   3.00000000000000
  ... ...
&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=        2011           3          22          12           0
          0
  dh1  =            3
 rmse_var = SMOIS
 ndim1 =            3
 ordering = XYZ
 staggering =  N/A
 start_index =            1           1           1           0
 end_index =          348         247           6           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 =
        349         247          50           0
  k,max,min,mid U=           1   20.05023      -21.65548      -6.996003
  k,max,min,mid U=           2   20.64079      -22.58930      -7.982791
  k,max,min,mid U=           3   21.84538      -24.38444      -9.791903
  k,max,min,mid U=           4   24.33893      -27.59095      -12.01432
  k,max,min,mid U=           5   27.30596      -29.83475      -14.94501
  k,max,min,mid U=           6   28.86383      -31.97169      -14.71747
  k,max,min,mid U=           7   30.24547      -33.28873      -12.40972
```

```
 k,max,min,mid U=           8   32.03448      -33.63768      -9.878036
```

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                        2352      2572      8367      2673
   OBS_PARA: t                         4617      4331     12418      4852
   OBS_PARA: q                         3828      3908     11096      3632
   OBS_PARA: pw                          89        31       141        23
   OBS_PARA: uv                        5704      4835     15025      4900
   OBS_PARA: sst                          0         0         2         0
```

   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:

```
LBSOI:  START pcgsoi jiter=           1
Initial cost function =  6.9090580198290321 28E+04
Initial gradient norm =  7.9292682527294516 60E+02
cost,grad,step,b,step? =   1   0  6.9090580198290321 28E+04  7.9292682527294516 60E+02
2.0197945160872247 95E-02  0.0000000000000000 00E+00  good
cost,grad,step,b,step? =   1   1  5.6391466548561264 06E+04  5.7840925365385066 93E+02
2.1270822204243949 87E-02  5.3211345864100800 81E-01  good
cost,grad,step,b,step? =   1   2  4.9275158453725525 76E+04  5.0456609283011266 60E+02
1.9031210558775154 41E-02  7.6096671298618012 71E-01  good
... ...
```

   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  44  3.5710742725481199 16E+04  1.2281457510933782 23E-02
1.0795612154218505 59E-02  1.6475964023213414 13E+00  good
cost,grad,step,b,step? =   2  45  3.5710742723852861 67E+04  7.1981471318132385 01E-03
3.9613876431373276 63E-02  3.4351344255047139 29E-01  good
 PCGSOI: WARNING **** Stopping inner iteration ***
 gnorm  0.8240910884607622 73E-10 less than  0.1000000000000000 4E-09
```

   Clearly, at the 45th 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 `6.9090580198290321 28E+04` at the

beginning and a value of `3.571074272385286167E+04` at the final iteration. This means the value has reduced by almost half, 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=   103989.1       64030.38
max,min MU=   3989.133      -2038.406
rmse_var=MU
ordering=XY
WrfType,WRF_REAL=         104         104
ndim1=           2
staggering= N/A
start_index=           1           1           1           0
end_index1=         348         247          50           0
k,max,min,mid T=           1   310.9237       233.9488       280.6478
k,max,min,mid T=           2   311.3738       235.8140       280.8113
k,max,min,mid T=           3   311.6963       238.1567       280.9817
k,max,min,mid T=           4   312.7513       243.5092       281.3736
k,max,min,mid T=           5   313.2865       242.0755       282.9548
k,max,min,mid T=           6   313.7715       247.9672       285.4344
k,max,min,mid T=           7   314.9392       250.0288       289.2765
k,max,min,mid T=           8   316.4144       251.4609       293.2987
k,max,min,mid T=           9   317.5127       254.6357       295.5367
k,max,min,mid T=          10   318.7207       256.3144       298.0647
k,max,min,mid T=          11   319.6920       257.8738       299.7420
k,max,min,mid T=          12   321.2617       260.0837       302.3682
... ...
```

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

```
    ENDING DATE-TIME    JUN 06,2014  16:47:10.518  157  FRI   2456815
    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_t1.2011032212 (fort.203)*
  This file shows how the background and analysis fields fit to temperature observations. The contents of this file show three data types were used in the analysis: 120, 130, and 180. 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:

it	obs	type	styp	ptop / pbot	1000.0 / 1200.0	900.0 / 1000.0	800.0 / 900.0	600.0 / 800.0	400.0 / 600.0	300.0 / 400.0	250.0 / 300.0	200.0 / 250.0	150.0 / 200.0	100.0 / 150.0	50.0 / 100.0	0.0 / 2000.0
o-g 01	t	120	0000	count	185	527	573	995	**1117**	637	226	443	642	979	855	8692
o-g 01	t	120	0000	bias	0.62	0.88	-0.22	-0.20	**-0.19**	-0.38	-1.07	-0.89	-0.95	-0.99	-1.63	-0.92
o-g 01	t	120	0000	rms	2.62	2.57	2.01	1.31	**0.90**	1.02	1.66	1.71	1.94	1.86	2.67	2.27
o-g 01	t	130	0000	count	0	0	0	0	0	11	177	626	67	0	0	881
o-g 01	t	130	0000	bias	0.00	0.00	0.00	0.00	0.00	0.18	0.06	-0.02	-1.23	0.00	0.00	-0.09
o-g 01	t	130	0000	rms	0.00	0.00	0.00	0.00	0.00	0.96	0.97	1.48	2.28	0.00	0.00	1.46
o-g 01	t	180	0000	count	1260	28	0	0	0	0	0	0	0	0	0	1288
o-g 01	t	180	0000	bias	0.67	0.80	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.68
o-g 01	t	180	0000	rms	1.76	1.19	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.75
o-g 01		all		count	1445	555	573	995	1117	648	403	1069	709	979	855	**10861**
o-g 01		all		bias	0.67	0.88	-0.22	-0.20	-0.19	-0.37	-0.57	-0.38	-0.97	-0.99	-1.63	**-0.66**
o-g 01		all		rms	1.89	2.52	2.01	1.31	0.90	1.02	1.40	1.58	1.97	1.86	2.67	**2.16**
o-g 03	t	120	0000	count	185	527	573	995	1117	637	226	443	642	979	855	8692
o-g 03	t	120	0000	bias	0.38	0.51	-0.15	-0.04	-0.02	0.02	-0.24	-0.17	-0.03	-0.17	-0.32	-0.10
o-g 03	t	120	0000	rms	2.13	2.06	1.59	1.00	0.61	0.58	0.90	1.01	1.27	1.36	1.93	1.46
o-g 03	t	130	0000	count	0	0	0	0	0	11	177	626	67	0	0	881
o-g 03	t	130	0000	bias	0.00	0.00	0.00	0.00	0.00	0.19	-0.03	0.08	-0.36	0.00	0.00	0.02
o-g 03	t	130	0000	rms	0.00	0.00	0.00	0.00	0.00	0.67	0.77	1.11	1.52	0.00	0.00	1.09
o-g 03	t	180	0000	count	1260	28	0	0	0	0	0	0	0	0	0	1288
o-g 03	t	180	0000	bias	0.42	0.21	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.41
o-g 03	t	180	0000	rms	1.39	0.61	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.38
o-g 03		all		count	1445	555	573	995	1117	648	403	1069	709	979	855	**10861**
o-g 03		all		bias	0.41	0.50	-0.15	-0.04	-0.02	0.03	-0.15	-0.03	-0.06	-0.17	-0.32	**-0.03**
o-g 03		all		rms	1.51	2.01	1.59	1.00	0.61	0.58	0.84	1.07	1.30	1.36	1.93	**1.42**

  For example: data type 120 has 1117 observations in layer 400.0-600.0 hPa, a bias of -0.19, and a rms of 0.90. 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 10861 observations total, a bias of -0.66, and a rms of 2.16.

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 10861 total observations, a bias of -0.03, and a rms of 1.42. This shows that from the background to the analysis the bias reduced from -0.66 to -0.03, and the rms reduced from 2.16 to 1.42. This is about a 34% reduction, which is a reasonable value for large-scale analysis.

- *fit_w1.2011032212 (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	1161	1172	1181	2152	1944	1084	558	1134	653	877	915	**14730**
o-g 01		all		bias	-0.31	1.02	0.65	0.12	-0.07	-0.06	-0.10	0.85	1.05	0.20	0.28	**0.35**
o-g 01		all		rms	3.43	3.90	4.27	4.34	4.86	4.93	5.06	5.39	5.74	5.60	5.59	**4.76**
o-g 03		all		count	1163	1173	1222	2186	1953	1096	558	1132	651	881	915	**14829**
o-g 03		all		bias	0.27	0.62	0.29	0.24	0.16	0.17	0.09	0.36	0.51	0.22	0.56	**0.38**
o-g 03		all		rms	2.70	2.80	2.67	2.51	2.71	2.48	2.70	3.16	3.76	4.30	4.93	**3.31**

  O-B: 14730 observations in total, bias is 0.35 and rms is 4.76
  O-A: 14829 observations in total, bias is 0.38 and rms is 3.31
  The total bias was not reduced, however the rms reduced from 4.76 to 3.31 (~30% reduction).

- *fit_q1.2011032212 (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	686	228	312	343	228	561	423	473	571	415	0	**4240**
o-g 01		all		bias	1.37	-0.29	1.15	0.92	0.93	0.23	-4.64	-7.81	-8.29	-8.55	0.00	**-2.84**
o-g 01		all		rms	12.92	17.73	16.92	15.55	15.36	15.72	19.37	21.03	21.42	17.97	0.00	**17.61**
o-g 03		all		count	686	229	312	343	228	561	423	473	571	415	0	**4241**
o-g 03		all		bias	0.12	-1.58	0.31	0.42	-0.51	0.65	-1.00	-0.72	-1.16	-3.98	0.00	**-0.68**
o-g 03		all		rms	7.08	10.27	10.59	9.65	10.50	11.49	14.65	14.35	14.33	13.09	0.00	**11.91**

  O-B: 4240 observations in total and bias is -2.84 and rms is 17.61

O-A: 4241 observations in total and bias is -0.68 and rms is 11.91

The total bias and rms were reduced.

- *fit_p1.2011032212 (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          13988    0.4224    1.0437    0.7592    0.6952

o-g 03             all          14004    0.0169    0.6790    0.3253    0.3119
```

O-B: 13988 observations in total and bias is 0.4224 and rms is 1.0437

O-A: 14004 observations in total and bias is 0.0169 and rms is 0.6790

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  6.909058019829032128E+04  7.929268252729451660E+02
1   1  5.639146654856126406E+04  5.784092536538506693E+02
1   2  4.927515845372552576E+04  5.045660928301126660E+02
1   3  4.443006075436472020E+04  3.674119598509449247E+02
1   4  4.185273756874063838E+04  3.006040049451294180E+02
          … …
2  41  3.571074274584063824E+04  1.678617300186252895E-02
2  42  3.571074273608899966E+04  1.226717362922756516E-02
2  43  3.571074272893991292E+04  9.568074118432539146E-03
```

```
2  44  3.57107427254811991 6E+04  1.22814575109337822 3E-02
2  45  3.57107427238528616 7E+04  7.19814713181323850 1E-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:

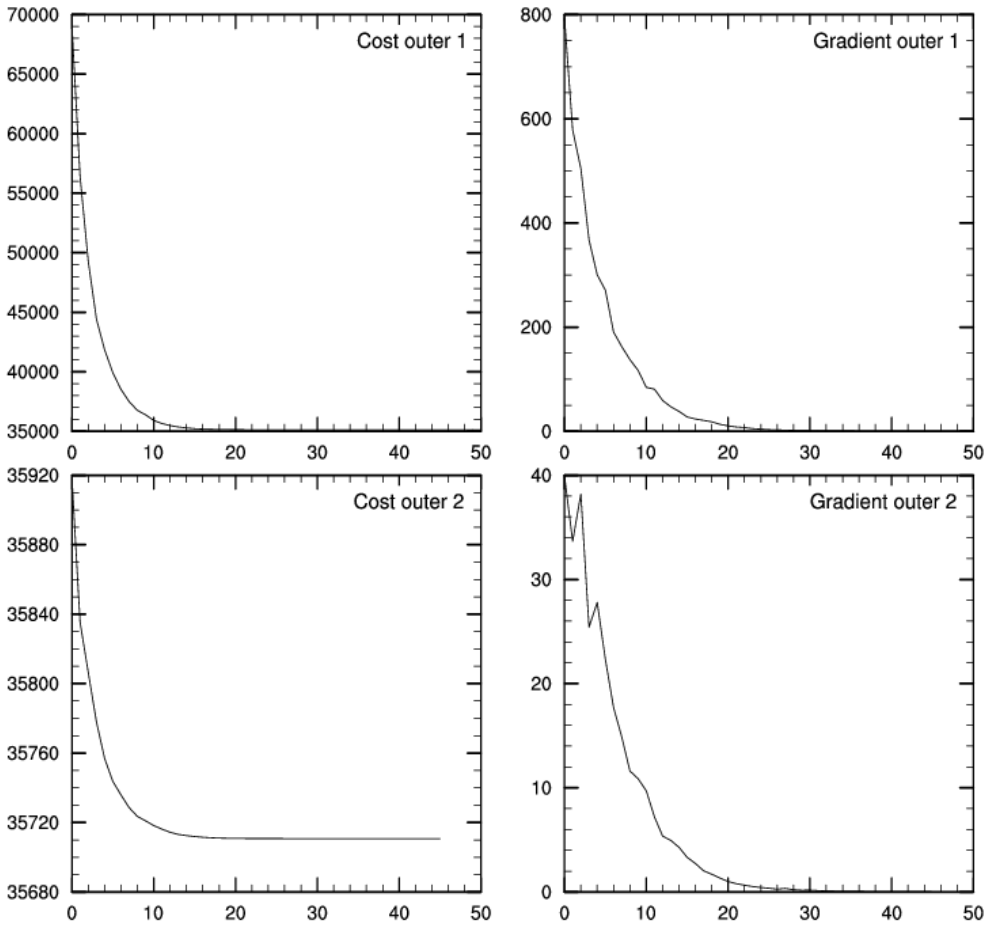


Figure 5.2: Cost function and norm of gradient change with iteration steps

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.

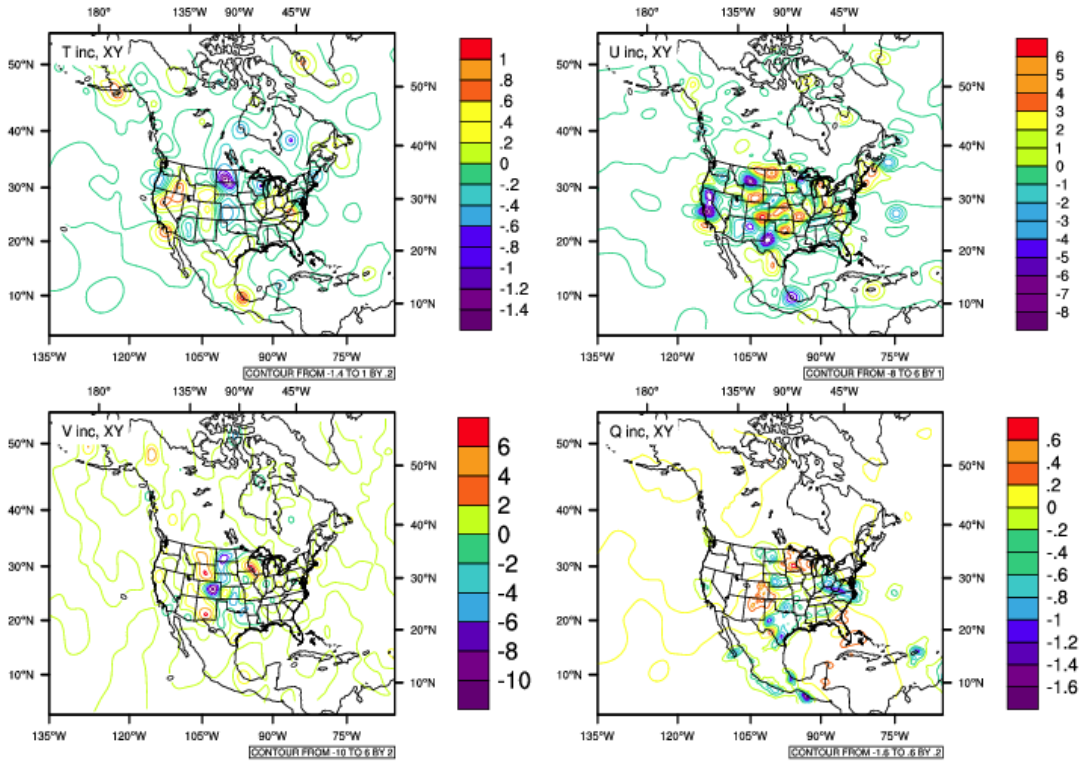


Figure 5.3: Analysis increment at the 15th level

It can be clearly seen that the U.S. CONUS domain has many upper level observations 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 three radiance BUFR files:

AMSU-A: *gdas1.t12z.1bamua.tm00.bufr_d*
AMSU-B: *gdas1.t12z.1bamub.tm00.bufr_d*
HIRS4: *gdas1.t12z.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.ksh:*

```
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.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-17 should be read from a BUFR file named as '*amsuabufr'*:

```
dfile(30)='amsuabufr',dtype(30)='amsua',dplat(30)='n17',dsis(30)='amsua_n17',dval(30)=0.0,dthin(30)=2,
```

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)=60.0,dmesh(4)=60.0,dmesh(5)=120

dfile(30)='amsuabufr', dtype(30)='amsua',dplat(30)='n17', dsis(30)='amsua_n17', dval(30)=0.0, dthin(30)=2, dsfcalc(30)=1,
```

  The first line of `&OBS_INPUT` lists multiple mesh grids as elements of the array `dmesh`. For the line specifying a data type, the 2nd last element of that line is a specification such as, '`dthin(30)=2`'. This selects the mesh grid to be used for thinning. It can be seen that the data thinning option for NOAA-17 AMSU-A observations is 60 km because the `dmesh(2)` is set to be 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 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 3.3, 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.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 three 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                    2352     2572     8367     2673
OBS_PARA: t                     4617     4331    12418     4852
OBS_PARA: q                     3828     3908    11096     3632
OBS_PARA: pw                      89       31      141       23
OBS_PARA: uv                    5704     4835    15025     4900
OBS_PARA: sst                      0        0        2        0
OBS_PARA: hirs4     metop-a        0        0      416      731
OBS_PARA: amsua     n15         2563     1323     1048     1669
OBS_PARA: amsua     n18         1002     2119        0      390
OBS_PARA: amsua     metop-a        0        0     1268     2279
OBS_PARA: amsub     n17            0        0     1717     2891
OBS_PARA: hirs4     n19          244     1093        0      235
OBS_PARA: amsua     n19          651     3486        0      469
```

When comparing this output to the content in step 3 of section 5.1.3, it can be seen that there are 7 new radiance data types that have been read in: HIRS4 from METOP-A and NOAA-19, AMSU-A from NOAA-15, NOAA-18, NOAA-19, and METOP-A, and AMSU-B from NOAA-17. 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   128055   84768   34473   45674.   45674.   1.3249   1.3249
o-g 01 rad  n17    amsub   213920   9164        0   0.0000   0.0000   0.0000   0.0000
```

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

```
o-g 03 rad  n15    amsua   128055   84768   55754   11706.  11706.    0.20995  0.20995
o-g 03 rad  n17    amsub   213920    9164       0   0.0000   0.0000    0.0000  0.0000
```

From the above information, it can be seen that AMSU-A data from NOAA-15 have 128055 observations within the analysis time window and domain. After thinning, 84768 of this data type remained, and only 34473 were used in the analysis. The penalty for this data decreased from 45674 to 11706 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 55754 from 34473. In this analysis, AMSU-B data from NOAA-17 have 213920 within the analysis time window and domain. After thinning, only 9164 were left and none of them were used in the analysis because of quality control in GSI.

The statistics for each channel can be view 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          2716    711      3.000    0.5630801    0.9124044    0.2095496    2.1696594    1.9684867
   2    2 amsua_n15          2656    773      2.000    0.1456301    1.0357824    0.2653187    2.2443727    1.9910710
   3    3 amsua_n15          3262    164      2.000    1.1713726   -1.2546513    0.3092257    2.0114653    1.5722096
   4    4 amsua_n15          3426      0      0.600   -0.2994171   -0.2676203    0.5595584    0.5738303    0.5076028
   5    5 amsua_n15          3426     10      0.300   -0.1874673   -0.2278419    0.8805730    0.3347520    0.2452487
   6    6 amsua_n15          6016     30      0.230   -1.5989361   -0.0059940    0.9904155    0.2467110    0.2466381
   7    7 amsua_n15          4009   2514      0.250   -0.5536621   -0.4306837    3.0917986    0.4714137    0.1916831
   8    8 amsua_n15          3257   3266      0.275   -0.6704360   -0.5495220    3.6424814    0.5737176    0.1648560
   9    9 amsua_n15          1472   5051      0.340   -1.2866839   -0.8251637    4.8293449    0.8342111    0.1225283
  10   10 amsua_n15            77   6446      0.400   -0.9223207   -0.9628485    5.1222374    0.9669377    0.0888327
  12   12 amsua_n15          1785   4738      1.000    1.2872243    0.0824760    0.0641187    0.4914450    0.4844749
  15   15 amsua_n15          2371   1057      3.000    1.5901494   -1.2085643    0.2627504    2.4249496    2.1023209
```

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

```
1    1 amsua_n15          3335    536      3.000    0.1532669    0.3705822    0.0859183    1.6728000    1.6312353
   2    2 amsua_n15          3265    606      2.000    0.5596236    0.6186692    0.1029505    1.8052958    1.6959780
   3    3 amsua_n15          3665    205      2.000    1.1658108   -1.0049837    0.1797988    1.7751874    1.4633175
   4    4 amsua_n15          3870      0      0.600   -0.1883404    0.0322754    0.1056377    0.3250804    0.3234742
   5    5 amsua_n15          3870      1      0.300    0.0500893   -0.0069152    0.1573053    0.1635352    0.1633890
   6    6 amsua_n15          6508      0      0.230   -1.2026015    0.0082707    0.2211057    0.1213394    0.1210572
   7    7 amsua_n15          6523      0      0.250   -0.1563331   -0.0025910    0.2134444    0.1235107    0.1234835
   8    8 amsua_n15          6523      0      0.275   -0.2381956   -0.0024279    0.1909097    0.1299197    0.1298970
   9    9 amsua_n15          6477     46      0.340   -0.5467628   -0.0302137    0.2847807    0.2002734    0.1979812
  10   10 amsua_n15          6272    251      0.400   -0.3893920   -0.1402203    0.4749189    0.3195797    0.2871749
  12   12 amsua_n15          2462   4061      1.000    3.0432797    0.0716991    0.0411064    0.4017417    0.3952918
  15   15 amsua_n15          2984    886      3.000    1.7837674   -0.9168598    0.1358733    1.9468460    1.7174334
```

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 to the observation more or less.

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).

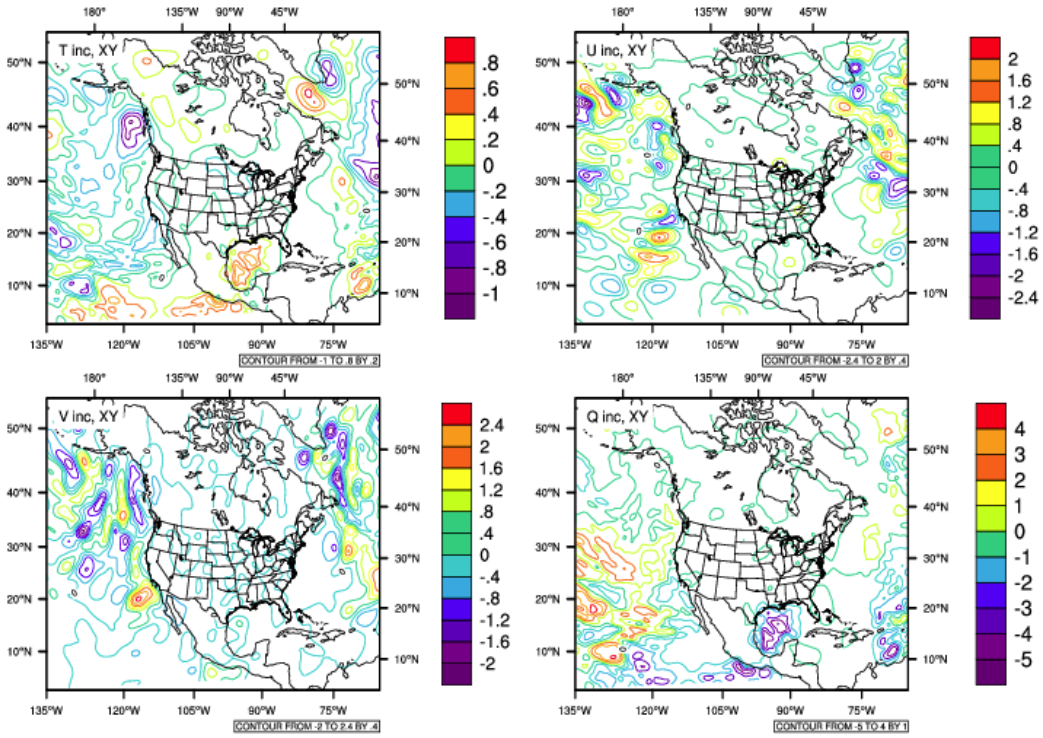


Figure 5.4: Analysis increment fields of PrepBUFR and Radiance data analysis comparing to the analysis with PREPBUFR only at level 6

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. Chanel 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.

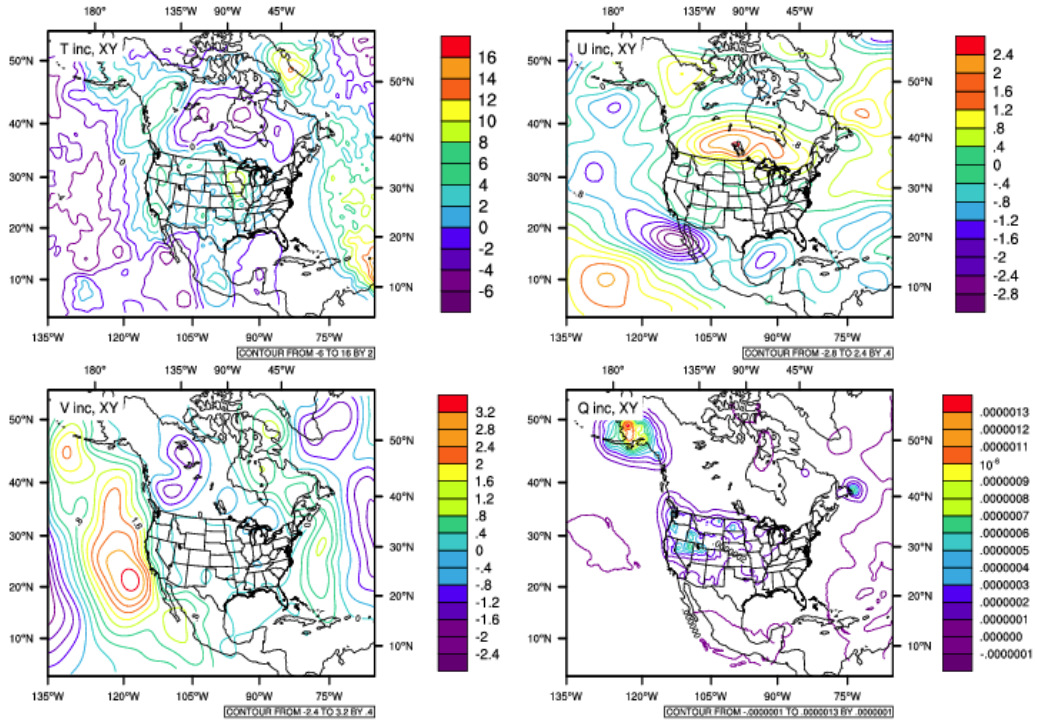


Figure 5.5: Analysis increment fields PrepBUFR and Radiance data analysis comparing to the analysis with PREPBUFR only at level 49

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, Channel 11 and 14 from amsua_n15 and channel 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 two grid-spaces, which may 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 scripts 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.t12z.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(11)='gpsrobufr', dtype(11)='gps_ref', dplat(11)=' ', dsis(11)='gps', dval(11)=1.0, dthin(11)=0, dsfcalc(11)=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.ksh* has been submitted, move into the working directory, *gsiprd_2011032212_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                    2352       2572       8367       2673
OBS_PARA: t                     4617       4331      12418       4852
OBS_PARA: q                     3828       3908      11096       3632
OBS_PARA: pw                      89         31        141         23
OBS_PARA: uv                    5704       4835      15025       4900
OBS_PARA: sst                      0          0          2          0
OBS_PARA: gps_ref               3538       5580       2277       6768
```

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	gps	722	0000	count	0	0	0	0	12	12	7	8	9	12	19	109
o-g 01	gps	722	0000	bias	0.00	0.00	0.00	0.00	0.53	-0.21	-0.27	-0.16	-0.05	0.49	-0.13	-0.07
o-g 01	gps	722	0000	rms	0.00	0.00	0.00	0.00	0.76	0.28	0.28	0.18	0.38	0.56	0.84	0.67
o-g 01	gps	740	0000	count	1	13	20	96	153	102	60	72	90	128	219	1352
o-g 01	gps	740	0000	bias	-0.24	-0.44	-0.05	0.20	0.03	-0.08	0.13	0.20	0.14	0.34	-0.07	-0.05
o-g 01	gps	740	0000	rms	0.24	1.25	1.18	1.27	0.63	0.45	0.36	0.40	0.41	0.62	0.53	0.72
…																
o-g 01	gps	786	0000	count	0	0	0	0	0	9	7	9	10	14	23	116
o-g 01	gps	786	0000	bias	0.00	0.00	0.00	0.00	0.00	-0.06	-0.02	0.07	0.81	-0.56	-0.25	-0.25
o-g 01	gps	786	0000	rms	0.00	0.00	0.00	0.00	0.00	0.27	0.04	0.35	1.12	0.91	0.61	0.91
o-g 01		all		count	4	77	138	494	851	634	411	497	618	855	1416	8496
o-g 01		all		bias	-0.50	-0.24	-0.11	-0.16	-0.12	-0.02	0.12	0.16	0.09	0.07	-0.06	-0.06
o-g 01		all		rms	0.77	0.88	0.80	1.03	0.77	0.42	0.44	0.50	0.54	0.62	0.57	0.66

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	gps	004	0000	count	0	0	0	0	0	29	48	66	86	117	165	759
o-g 03	gps	004	0000	bias	0.00	0.00	0.00	0.00	0.00	-0.07	-0.08	-0.02	-0.00	-0.04	-0.03	0.00
o-g 03	gps	004	0000	rms	0.00	0.00	0.00	0.00	0.00	0.31	0.20	0.20	0.20	0.21	0.36	0.37
o-g 03	gps	722	0000	count	0	0	0	0	12	12	7	8	9	12	18	109
o-g 03	gps	722	0000	bias	0.00	0.00	0.00	0.00	0.19	-0.14	-0.11	-0.08	-0.09	0.11	0.03	0.02
o-g 03	gps	722	0000	rms	0.00	0.00	0.00	0.00	0.33	0.19	0.14	0.11	0.20	0.15	0.74	0.50
…																
o-g 03	gps	786	0000	count	0	0	0	0	0	9	7	9	10	14	24	121
o-g 03	gps	786	0000	bias	0.00	0.00	0.00	0.00	0.00	-0.09	-0.06	0.04	0.11	-0.19	0.09	0.06
o-g 03	gps	786	0000	rms	0.00	0.00	0.00	0.00	0.00	0.24	0.08	0.19	0.35	0.48	0.63	0.61

o-g 03	all	count	5	93	171	561	858	633	413	500	631	868	1418	8669
o-g 03	all	bias	-0.32	-0.09	-0.08	-0.01	-0.02	-0.04	-0.00	0.01	0.01	-0.01	-0.01	0.01
o-g 03	all	rms	0.44	0.66	0.81	0.80	0.56	0.28	0.20	0.21	0.24	0.28	0.40	0.48

It can be seen that there are many types of GPSRO data used in the analysis. Most of the GPS RO data are located in the upper levels, with a total of 8496 observations used in the analysis during the 1st outer loop, and 8669 used to calculate O-A. After the analysis, the data bias reduced from -0.06 to 0.01, and the rms was reduced from 0.66 to 0.48. 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.

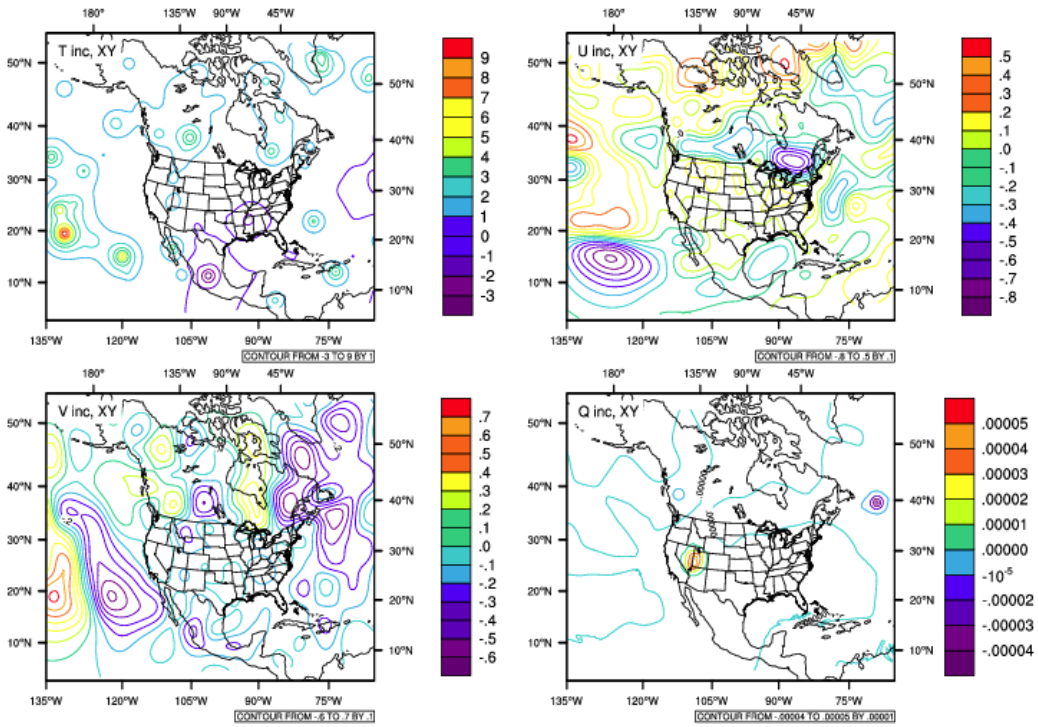


Figure 5.6: Analysis increment fields comparing to the analysis with PrepBUFR only at level 48

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. Here we only give an example of changes for ARW netcdf format. There is an assumption that all the ensemble members are located under the directory located in the path 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: Setup 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,v; otherwise, ensemble perturbation wind variables are stream function, 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 chapter 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 actually has a broader applications other than regional analysis with WRF. 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_n15_anl.2011032212  diag_amsua_n19_ges.2011032212
diag_amsua_n15_ges.2011032212  diag_conv_anl.2011032212
diag_amsua_n18_anl.2011032212  diag_conv_ges.2011032212
diag_amsua_n18_ges.2011032212  diag_hirs4_n19_anl.2011032212
diag_amsua_n19_anl.2011032212  diag_hirs4_n19_ges.2011032212
```

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"** is radiance observation from AMSU-A in 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.2011032212` and `diag_conv_ges.2011032212`

read_diag_rad.f90 : Reads the diagnosis files for radiance observation. For example:
`diag_amsua_n15_anl.2011032212 diag_amsua_n15_ges.2011032212`
`diag_hirs4_n19_anl.2011032212 diag_hirs4_n19_ges.2011032212`

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.2011032212` 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.2011032212',
  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:

```
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 @ 21997     : 180       0.00    42.65   190.98 1013.20     1    1013.2     -0.35
t  @ PADK      : 187      -0.07    51.88   183.35  993.34     1     276.0      0.33
uv @ RD01709C : 242      -1.48    40.50   189.00  886.30     1     14.04      3.12  -11.80  -0.24
```

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:

```
  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
  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
  diagbuf(22) = vegetation fraction
  diagbuf(23) = snow depth
  diagbuf(24) = surface wind speed (m/s)

  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.202 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*.

Before making plots, there are two lines in ncl script *GSI_cost_gradient.ncl* that needs to be changed based on content of *cost_gradient.txt*:

line 6: nloop1= inner iteration number in 1st outer loop
line 7: nloop2= inner iteration number in 2nd outer loop

Then, in the same directory as the file *cost_gradient.txt*, run the ncl script:

```
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 (like *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

```
BS_INPUT
 dmesh(1)=120.0,dmesh(2)=60.0,dmesh(3)=60.0,dmesh(4)=60.0,dmesh(5)=120,time_window_max=1.5,
 dfile(01)='prepbufr',  dtype(01)='ps',        dplat(01)=' ',       dsis(01)='ps',               dval(01)=1.0, dthin(01)=0, dsfcalc(01)=0,
 dfile(02)='prepbufr'   dtype(02)='t',         dplat(02)=' ',       dsis(02)='t',                dval(02)=1.0, dthin(02)=0, dsfcalc(02)=0,
 dfile(03)='prepbufr',  dtype(03)='q',         dplat(03)=' ',       dsis(03)='q',               dval(03)=1.0, dthin(03)=0, dsfcalc(03)=0,
 dfile(04)='prepbufr',  dtype(04)='pw',        dplat(04)=' ',       dsis(04)='pw',               dval(04)=1.0, dthin(04)=0, dsfcalc(04)=0,
 dfile(05)='satwnd',    dtype(05)='uv',        dplat(05)=' ',       dsis(05)='uv',               dval(05)=1.0, dthin(05)=0, dsfcalc(05)=0,
 dfile(06)='prepbufr',  dtype(06)='uv',        dplat(06)=' ',       dsis(06)='uv',               dval(06)=1.0, dthin(06)=0, dsfcalc(06)=0,
 dfile(07)='prepbufr',  dtype(07)='spd',       dplat(07)=' ',       dsis(07)='spd',              dval(07)=1.0, dthin(07)=0, dsfcalc(07)=0,
 dfile(08)='prepbufr',  dtype(08)='dw',        dplat(08)=' ',       dsis(08)='dw',               dval(08)=1.0, dthin(08)=0, dsfcalc(08)=0,
 dfile(09)='radarbufr', dtype(09)='rw',        dplat(09)=' ',       dsis(09)='rw',               dval(09)=1.0, dthin(09)=0, dsfcalc(09)=0,
 dfile(10)='prepbufr',  dtype(10)='sst',       dplat(10)=' ',       dsis(10)='sst',              dval(10)=1.0, dthin(10)=0, dsfcalc(10)=0,
 dfile(11)='gpsrobufr', dtype(11)='gps_ref',   dplat(11)=' ',       dsis(11)='gps',              dval(11)=1.0, dthin(11)=0, dsfcalc(11)=0,
 dfile(12)='ssmirrbufr',dtype(12)='pcp_ssmi',  dplat(12)='dmsp',    dsis(12)='pcp_ssmi',         dval(12)=1.0, dthin(12)=-1,dsfcalc(12)=0,
 dfile(13)='tmirrbufr', dtype(13)='pcp_tmi',   dplat(13)='trmm',    dsis(13)='pcp_tmi',          dval(13)=1.0, dthin(13)=-1,dsfcalc(13)=0,
 dfile(14)='sbuvbufr',  dtype(14)='sbuv2',     dplat(14)='n16',     dsis(14)='sbuv8_n16',        dval(14)=1.0, dthin(14)=0, dsfcalc(14)=0,
 dfile(15)='sbuvbufr',  dtype(15)='sbuv2',     dplat(15)='n17',     dsis(15)='sbuv8_n17',        dval(15)=1.0, dthin(15)=0, dsfcalc(15)=0,
 dfile(16)='sbuvbufr',  dtype(16)='sbuv2',     dplat(16)='n18',     dsis(16)='sbuv8_n18',        dval(16)=1.0, dthin(16)=0, dsfcalc(16)=0,
 dfile(17)='hirs2bufr', dtype(17)='hirs2',     dplat(17)='n14',     dsis(17)='hirs2_n14',        dval(17)=6.0, dthin(17)=1, dsfcalc(17)=1,
 dfile(18)='hirs3bufr', dtype(18)='hirs3',     dplat(18)='n16',     dsis(18)='hirs3_n16',        dval(18)=0.0, dthin(18)=1, dsfcalc(18)=1,
 dfile(19)='hirs3bufr', dtype(19)='hirs3',     dplat(19)='n17',     dsis(19)='hirs3_n17',        dval(19)=6.0, dthin(19)=1, dsfcalc(19)=1,
 dfile(20)='hirs4bufr', dtype(20)='hirs4',     dplat(20)='n18',     dsis(20)='hirs4_n18',        dval(20)=0.0, dthin(20)=1, dsfcalc(20)=1,
 dfile(21)='hirs4bufr', dtype(21)='hirs4',     dplat(21)='metop-a', dsis(21)='hirs4_metop-a',    dval(21)=6.0, dthin(21)=1, dsfcalc(21)=1,
 dfile(22)='gsndrbufr', dtype(22)='sndr',      dplat(22)='g11',     dsis(22)='sndr_g11',         dval(22)=0.0, dthin(22)=1, dsfcalc(22)=0,
 dfile(23)='gsndrbufr', dtype(23)='sndr',      dplat(23)='g12',     dsis(23)='sndr_g12',         dval(23)=0.0, dthin(23)=1, dsfcalc(23)=0,
 dfile(24)='gimgrbufr', dtype(24)='goes_img',  dplat(24)='g11',     dsis(24)='imgr_g11',         dval(24)=0.0, dthin(24)=1, dsfcalc(24)=0,
 dfile(25)='gimgrbufr', dtype(25)='goes_img',  dplat(25)='g12',     dsis(25)='imgr_g12',         dval(25)=0.0, dthin(25)=1, dsfcalc(25)=0,
 dfile(26)='airsbufr',  dtype(26)='airs',      dplat(26)='aqua',    dsis(26)='airs281SUBSET_aqua', dval(26)=20.0,dthin(26)=1, dsfcalc(26)=1,
 dfile(27)='msubufr',   dtype(27)='msu',       dplat(27)='n14',     dsis(27)='msu_n14',          dval(27)=2.0, dthin(27)=2, dsfcalc(27)=1,
 dfile(28)='amsuabufr', dtype(28)='amsua',     dplat(28)='n15',     dsis(28)='amsua_n15',        dval(28)=10.0,dthin(28)=2, dsfcalc(28)=1,
 dfile(29)='amsuabufr', dtype(29)='amsua',     dplat(29)='n16',     dsis(29)='amsua_n16',        dval(29)=0.0, dthin(29)=2, dsfcalc(29)=1,
 dfile(30)='amsuabufr', dtype(30)='amsua',     dplat(30)='n17',     dsis(30)='amsua_n17',        dval(30)=0.0, dthin(30)=2, dsfcalc(30)=1,
 dfile(31)='amsuabufr', dtype(31)='amsua',     dplat(31)='n18',     dsis(31)='amsua_n18',        dval(31)=10.0,dthin(31)=2, dsfcalc(31)=1,
 dfile(32)='amsuabufr', dtype(32)='amsua',     dplat(32)='metop-a', dsis(32)='amsua_metop-a',    dval(32)=10.0,dthin(32)=2, dsfcalc(32)=1,
 dfile(33)='airsbufr',  dtype(33)='amsua',     dplat(33)='aqua',    dsis(33)='amsua_aqua',       dval(33)=5.0, dthin(33)=2, dsfcalc(33)=1,
 dfile(34)='amsubbufr', dtype(34)='amsub',     dplat(34)='n15',     dsis(34)='amsub_n15',        dval(34)=3.0, dthin(34)=3, dsfcalc(34)=1,
 dfile(35)='amsubbufr', dtype(35)='amsub',     dplat(35)='n16',     dsis(35)='amsub_n16',        dval(35)=3.0, dthin(35)=3, dsfcalc(35)=1,
 dfile(36)='amsubbufr', dtype(36)='amsub',     dplat(36)='n17',     dsis(36)='amsub_n17',        dval(36)=3.0, dthin(36)=3, dsfcalc(36)=1,
 dfile(37)='mhsbufr',   dtype(37)='mhs',       dplat(37)='n18',     dsis(37)='mhs_n18',          dval(37)=3.0, dthin(37)=3, dsfcalc(37)=1,
 dfile(38)='mhsbufr',   dtype(38)='mhs',       dplat(38)='metop-a', dsis(38)='mhs_metop-a',      dval(38)=3.0, dthin(38)=3, dsfcalc(38)=1,
 dfile(39)='ssmitbufr', dtype(39)='ssmi',      dplat(39)='f13',     dsis(39)='ssmi_f13',         dval(39)=0.0, dthin(39)=4, dsfcalc(39)=0,
 dfile(40)='ssmitbufr', dtype(40)='ssmi',      dplat(40)='f14',     dsis(40)='ssmi_f14',         dval(40)=0.0, dthin(40)=4, dsfcalc(40)=0,
 dfile(41)='ssmitbufr', dtype(41)='ssmi',      dplat(41)='f15',     dsis(41)='ssmi_f15',         dval(41)=0.0, dthin(41)=4, dsfcalc(41)=0,
 dfile(42)='amsrebufr', dtype(42)='amsre_low', dplat(42)='aqua',    dsis(42)='amsre_aqua',       dval(42)=0.0, dthin(42)=4, dsfcalc(42)=1,
 dfile(43)='amsrebufr', dtype(43)='amsre_mid', dplat(43)='aqua',    dsis(43)='amsre_aqua',       dval(43)=0.0, dthin(43)=4, dsfcalc(43)=1,
 dfile(44)='amsrebufr', dtype(44)='amsre_hig', dplat(44)='aqua',    dsis(44)='amsre_aqua',       dval(44)=0.0, dthin(44)=4, dsfcalc(44)=1,
 dfile(45)='ssmisbufr', dtype(45)='ssmis_las', dplat(45)='f16',     dsis(45)='ssmis_f16',        dval(45)=0.0, dthin(45)=1, dsfcalc(45)=0,
 dfile(46)='ssmisbufr', dtype(46)='ssmis_uas', dplat(46)='f16',     dsis(46)='ssmis_f16',        dval(46)=0.0, dthin(46)=1, dsfcalc(46)=0,
 dfile(47)='ssmisbufr', dtype(47)='ssmis_img', dplat(47)='f16',     dsis(47)='ssmis_f16',        dval(47)=0.0, dthin(47)=1, dsfcalc(47)=0,
 dfile(48)='ssmisbufr', dtype(48)='ssmis_env', dplat(48)='f16',     dsis(48)='ssmis_f16',        dval(48)=0.0, dthin(48)=1, dsfcalc(48)=0,
 dfile(49)='gsndlbufr', dtype(49)='sndrd1',    dplat(49)='g12',     dsis(49)='sndrD1_g12',       dval(49)=1.5, dthin(49)=5, dsfcalc(49)=0,
 dfile(50)='gsndlbufr', dtype(50)='sndrd2',    dplat(50)='g12',     dsis(50)='sndrD2_g12',       dval(50)=1.5, dthin(50)=5, dsfcalc(50)=0,
 dfile(51)='gsndlbufr', dtype(51)='sndrd3',    dplat(51)='g12',     dsis(51)='sndrD3_g12',       dval(51)=1.5, dthin(51)=5, dsfcalc(51)=0,
 dfile(52)='gsndlbufr', dtype(52)='sndrd4',    dplat(52)='g12',     dsis(52)='sndrD4_g12',       dval(52)=1.5, dthin(52)=5, dsfcalc(52)=0,
 dfile(53)='gsndlbufr', dtype(53)='sndrd1',    dplat(53)='g11',     dsis(53)='sndrD1_g11',       dval(53)=1.5, dthin(53)=5, dsfcalc(53)=0,
 dfile(54)='gsndlbufr', dtype(54)='sndrd2',    dplat(54)='g11',     dsis(54)='sndrD2_g11',       dval(54)=1.5, dthin(54)=5, dsfcalc(54)=0,
 dfile(55)='gsndlbufr', dtype(55)='sndrd3',    dplat(55)='g11',     dsis(55)='sndrD3_g11',       dval(55)=1.5, dthin(55)=5, dsfcalc(55)=0,
 dfile(56)='gsndlbufr', dtype(56)='sndrd4',    dplat(56)='g11',     dsis(56)='sndrD4_g11',       dval(56)=1.5, dthin(56)=5, dsfcalc(56)=0,
 dfile(57)='gsndlbufr', dtype(57)='sndrd1',    dplat(57)='g13',     dsis(57)='sndrD1_g13',       dval(57)=1.5, dthin(57)=5, dsfcalc(57)=0,
 dfile(58)='gsndlbufr', dtype(58)='sndrd2',    dplat(58)='g13',     dsis(58)='sndrD2_g13',       dval(58)=1.5, dthin(58)=5, dsfcalc(58)=0,
 dfile(59)='gsndlbufr', dtype(59)='sndrd3',    dplat(59)='g13',     dsis(59)='sndrD3_g13',       dval(59)=1.5, dthin(59)=5, dsfcalc(59)=0,
 dfile(60)='gsndlbufr', dtype(60)='sndrd4',    dplat(60)='g13',     dsis(60)='sndrD4_g13',       dval(60)=1.5, dthin(60)=5, dsfcalc(60)=0,
 dfile(61)='iasibufr',  dtype(61)='iasi',      dplat(61)='metop-a', dsis(61)='iasi616_metop-a',  dval(61)=20.0,dthin(61)=1, dsfcalc(61)=1,
 dfile(62)='gomebufr',  dtype(62)='gome',      dplat(62)='metop-a', dsis(62)='gome_metop-a',     dval(62)=1.0, dthin(62)=6, dsfcalc(62)=0,
 dfile(63)='omibufr',   dtype(63)='omi',       dplat(63)='aura',    dsis(63)='omi_aura',         dval(63)=1.0, dthin(63)=6, dsfcalc(63)=0,
 dfile(64)='sbuvbufr',  dtype(64)='sbuv2',     dplat(64)='n19',     dsis(64)='sbuv8_n19',        dval(64)=1.0, dthin(64)=0, dsfcalc(64)=0,
 dfile(65)='hirs4bufr', dtype(65)='hirs4',     dplat(65)='n19',     dsis(65)='hirs4_n19',        dval(65)=6.0, dthin(65)=1, dsfcalc(65)=1,
 dfile(66)='amsuabufr', dtype(66)='amsua',     dplat(66)='n19',     dsis(66)='amsua_n19',        dval(66)=10.0,dthin(66)=2, dsfcalc(66)=1,
 dfile(67)='mhsbufr',   dtype(67)='mhs',       dplat(67)='n19',     dsis(67)='mhs_n19',          dval(67)=3.0, dthin(67)=3, dsfcalc(67)=1,
 dfile(68)='tcvitl'     dtype(68)='tcp',       dplat(68)=' ',       dsis(68)='tcp',              dval(68)=1.0, dthin(68)=0, dsfcalc(68)=0,
 dfile(69)='mlsbufr',   dtype(69)='mls',       dplat(69)='aura',    dsis(69)='mls_aura',         dval(69)=1.0, dthin(69)=0, dsfcalc(69)=0,
 dfile(70)='seviribufr',dtype(70)='seviri',    dplat(70)='m08',     dsis(70)='seviri_m08',       dval(70)=0.0, dthin(70)=1, dsfcalc(70)=0,
 dfile(71)='seviribufr',dtype(71)='seviri',    dplat(71)='m09',     dsis(71)='seviri_m09',       dval(71)=0.0, dthin(71)=1, dsfcalc(71)=0,
 dfile(72)='seviribufr',dtype(72)='seviri',    dplat(72)='m10',     dsis(72)='seviri_m10',       dval(72)=0.0, dthin(72)=1, dsfcalc(72)=0,
 dfile(73)='hirs4bufr', dtype(73)='hirs4',     dplat(73)='metop-b', dsis(73)='hirs4_metop-b',    dval(73)=0.0, dthin(73)=1, dsfcalc(73)=0,
 dfile(74)='amsuabufr', dtype(74)='amsua',     dplat(74)='metop-b', dsis(74)='amsua_metop-b',    dval(74)=0.0, dthin(74)=1, dsfcalc(74)=0,
 dfile(75)='mhsbufr',   dtype(75)='mhs',       dplat(75)='metop-b', dsis(75)='mhs_metop-b',      dval(75)=0.0, dthin(75)=1, dsfcalc(75)=0,
 dfile(76)='iasibufr',  dtype(76)='iasi',      dplat(76)='metop-b', dsis(76)='iasi616_metop-b',  dval(76)=0.0, dthin(76)=1, dsfcalc(76)=0,
 dfile(77)='gomebufr',  dtype(77)='gome',      dplat(77)='metop-b', dsis(77)='gome_metop-b',     dval(77)=0.0, dthin(77)=2, dsfcalc(77)=0,
 dfile(78)='atmsbufr',  dtype(78)='atms',      dplat(78)='npp',     dsis(78)='atms_npp',         dval(78)=0.0, dthin(78)=1, dsfcalc(78)=0,
 dfile(79)='crisbufr',  dtype(79)='cris',      dplat(79)='npp',     dsis(79)='cris_npp',         dval(79)=0.0, dthin(79)=1, dsfcalc(79)=0,
 dfile(80)='gsndlbufr', dtype(80)='sndrd1',    dplat(80)='g14',     dsis(80)='sndrD1_g14',       dval(80)=0.0, dthin(80)=1, dsfcalc(80)=0,
 dfile(81)='gsndlbufr', dtype(81)='sndrd2',    dplat(81)='g14',     dsis(81)='sndrD2_g14',       dval(81)=0.0, dthin(81)=1, dsfcalc(81)=0,
 dfile(82)='gsndlbufr', dtype(82)='sndrd3',    dplat(82)='g14',     dsis(82)='sndrD3_g14',       dval(82)=0.0, dthin(82)=1, dsfcalc(82)=0,
 dfile(83)='gsndlbufr', dtype(83)='sndrd4',    dplat(83)='g14',     dsis(83)='sndrD4_g14',       dval(83)=0.0, dthin(83)=1, dsfcalc(83)=0,
 dfile(84)='gsndlbufr', dtype(84)='sndrd1',    dplat(84)='g15',     dsis(84)='sndrD1_g15',       dval(84)=0.0, dthin(84)=1, dsfcalc(84)=0,
 dfile(85)='gsndlbufr', dtype(85)='sndrd2',    dplat(85)='g15',     dsis(85)='sndrD2_g15',       dval(85)=0.0, dthin(85)=1, dsfcalc(85)=0,
 dfile(86)='gsndlbufr', dtype(86)='sndrd3',    dplat(86)='g15',     dsis(86)='sndrD3_g15',       dval(86)=0.0, dthin(86)=1, dsfcalc(86)=0,
 dfile(87)='gsndlbufr', dtype(87)='sndrd4',    dplat(87)='g15',     dsis(87)='sndrD4_g15',       dval(87)=0.0, dthin(87)=1, dsfcalc(87)=0,
```