



User's Guide Version 3.5

August 2016

Ming Hu

*National Oceanic and Atmospheric Administration (NOAA)/Earth System Research Laboratory
Cooperative Institute for Research in Environmental Sciences (CIRES)*

Hui Shao, Don Stark, Kathryn Newman, Chunhua Zhou
National Center for Atmospheric Research (NCAR)

Xin Zhang

NOAA/Earth System Research Laboratory and CIRES



Acknowledgement

This user's guide is constructed with contributions from distributed GSI developers. We give our special acknowledgement to these contributors and reviewers, including, but not limit to:

National Centers for Environmental Prediction (NCEP) Environmental Modeling Center (EMC):

John Derber, Russ Treadon, Mike Lueken, Wan-Shu Wu, Andrew Collard, and Ed Safford

National Center for Atmospheric Research (NCAR):

Xiang-Yu Huang, Syed Rizvi, Zhiquan Liu, and Arthur Mizzi

National Oceanic and Atmospheric Administration (NOAA) Earth System Research Laboratory (ESRL):

Steve Weygandt, Dezso Devenyi, Joseph Olson, and Jeff Beck

The GSI community support and code management effort is sponsored by NOAA's Office of Oceanic and Atmospheric Research (OAR). This work is also facilitated by NCAR. NCAR is supported by the National Science Foundation (NSF).

Forward

This document is the 2016 Gridpoint Statistical Interpolation (GSI) User's Guide geared particularly for beginners. It describes the fundamentals of using GSI version (v) 3.5 released in July 2016. Advanced features of GSI as well as details of assimilation of specific data types can be found in the Advance GSI User's Guide, released together with this document and the v3.5 code release.

This User's Guide includes six chapters and three appendices:

Chapter 1 provides a background introduction of GSI.

Chapter 2 contains basic information about how to install and compile 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 the GSI applications for regional ARW cases, including the setup of different data types such as conventional, radiance, and GPSRO data and different analysis functions available in the GSI such as hybrid analysis.

Chapter 6 illustrates the GSI applications for global case and chemical cases.

Appendix A introduces the community tools available for GSI users.

Appendix B is content of the GSI namelist section OBS_INPUT.

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

For the latest version of GSI User's Guide and released code, please visit the GSI User's Website:

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

Please send questions and comments to the GSI help desk:

gsi-help@ucar.edu

This document and the annual GSI releases are made available through a community GSI effort jointly led by the Developmental Testbed Center (DTC) and the National Centers for Environmental Prediction (NCEP) Environmental Modeling Center (EMC), in collaboration

with other GSI developers. To help sustain this effort, we recommend for those who use the community released GSI, the GSI helpdesk, the GSI User's Guide, and other DTC GSI services, please refer to this community GSI effort in their work and publications.

For referencing this user's guide, please use:

Hu, M., H. Shao, D. Stark, K. Newman, C. Zhou, and X. Zhang, 2016: Grid-point Statistical Interpolation (GSI) User's Guide Version 3.5. Developmental Testbed Center. Available at <http://www.dtcenter.org/com-GSI/users/docs/index.php>, 141 pp.

For referencing the general aspect of the GSI community effort, please use:

Shao, H., J. Derber, X.-Y. Huang, M. Hu, K. Newman, D. Stark, M. Lueken, C. Zhou, L. Nance, Y.-H. Kuo, B. Brown, 2016: Bridging Research to Operations Transitions: Status and Plans of Community GSI. Bulletin of the American Meteorological Society, doi:10.1175/BAMS-D-13-00245.1, in press

Contents

1. Overview	1
1.1. GSI History and Background	1
1.2. GSI Becomes Community Code	2
1.2.1. GSI Code Management and Review Committee	2
1.2.2. Community Code Contributions	3
1.3. About This GSI Release	3
1.3.1. What Is New in This Release Version	4
1.3.2. Observations Used by This Version	4
2. Software Installation	7
2.1. Introduction	7
2.2. Obtaining and Setting Up the Source Code	8
2.3. Directory Structure, Source Code and Supplemental Libraries	9
2.4. Compiling GSI	10
2.4.1. Build Overview	10
2.4.2. Environment Variables	10
2.4.3. Configure and Compile	12
2.5. Example of Build	13
2.5.1. Intel Build	13
2.5.2. PGI Build	14
2.5.3. GNU Build	14
2.6. System Requirements and External Libraries	15
2.6.1. Compilers Tested for Release	15
2.7. Getting Help and Reporting Problems	16
3. Running GSI	17
3.1. Input Data Required to Run GSI	17
3.1.1. Background or First Guess Field	17
3.1.2. Observations	18
3.1.3. Fixed Files (Statistics and Control Files)	19
3.2. GSI Run Script	22
3.2.1. Steps in the GSI Run Script	23
3.2.2. Customization of the GSI Run Script	24

Contents

3.2.3.	Description of the Sample Regional Run Script to Run GSI	27
3.3.	GSI Analysis Result Files in Run Directory	38
3.4.	Introduction to Frequently Used GSI Namelist Options	40
3.4.1.	Set Up the Number of Outer Loop and Inner Loop	40
3.4.2.	Set Up the Analysis Variable for Moisture	41
3.4.3.	Set Up the Background File	41
3.4.4.	Set Up the Output of Diagnostic Files	42
3.4.5.	Set Up the GSI Recognized Observation Files	42
3.4.6.	Set Up Observation Time Window	43
3.4.7.	Set Up Data Thinning	43
3.4.8.	Set Up Background Error Factor	44
3.4.9.	Single Observation Test	44
4.	GSI Diagnostics and Tuning	45
4.1.	Understanding Standard Output (<i>stdout</i>)	45
4.2.	Single Observation Test	60
4.2.1.	Setup a Single Observation Test	61
4.2.2.	Examples of Single Observation Tests for GSI	61
4.3.	Control Data Usage	63
4.4.	Domain Partition for Parallelization and Observation Distribution	67
4.5.	Observation Innovation Statistics	68
4.5.1.	Conventional observations	68
4.5.2.	Satellite Radiance	73
4.6.	Convergence Information	77
4.7.	Conventional Observation Errors	79
4.7.1.	Getting Original Observation Errors	79
4.7.2.	Observation Error Gross Error Check within GSI	80
4.8.	Background Error Covariance	81
4.8.1.	Tuning Background Error Covariance through Namelist and Anavinfo	81
4.9.	Analysis Increments	82
4.10.	Running Time and Memory Usage	82
5.	GSI Applications for Regional 3DVar and 3D Hybrid EnVar	84
5.1.	Assimilating Conventional Observations with Regional GSI	86
5.1.1.	Run Script	86
5.1.2.	Run GSI and Check the Run Status	87
5.1.3.	Check for Successful GSI Completion	88
5.1.4.	Diagnose GSI Analysis Results	91
5.1.4.1.	Check Analysis Fit to Observations	91
5.1.4.2.	Check the Minimization	93
5.1.4.3.	Check the Analysis Increment	95
5.2.	Assimilating Radiance Data with Regional GSI	95
5.2.1.	Run Script	95
5.2.2.	Run GSI and Check Run Status	97
5.2.3.	Diagnose GSI Analysis Results	98
5.2.3.1.	Check File fort.207	98
5.2.3.2.	Check the Analysis Increment	99

Contents

5.3.	Assimilating GPS Radio Occultation Data with Regional GSI	101
5.3.1.	Run Script	101
5.3.2.	Run GSI and Check the Run Status	101
5.3.3.	Diagnose GSI Analysis Results	102
5.3.3.1.	Check File <i>fort.212</i>	102
5.3.3.2.	Check the Analysis Increment	102
5.4.	Introduction to GSI 3D Hybrid EnVar Analysis	103
5.5.	Summary	104
6.	Introduction to more GSI Applications	107
6.1.	Introduction to Global GSI analysis	107
6.1.1.	The Difference between Global and Regional GSI	108
6.1.2.	Global GFS Scripts	109
6.1.3.	Sample Results	112
6.2.	Introduction to Chemical Analysis	113
6.2.1.	Setup GSI Run Scripts for Chemical Analysis	113
6.2.2.	Sample Results	115
A.	GSI Community Tools	117
A.1.	BUFR Format and BUFR Tools	117
A.2.	Read GSI Diagnostic Files	117
A.3.	Read and Plot Convergence Information from <i>fort.220</i>	121
A.4.	Plot Single Observation Test Result and Analysis Increment	122
B.	Contents of Namelist Section OBS_INPUT	123
C.	GSI Namelist: Name, Default Value, Explanation	125
	Bibliography	125



1

Overview

1.1 GSI History and Background

The Gridpoint Statistical Interpolation (GSI) system is a unified data assimilation (DA) system for both global and regional applications. It was initially developed by the National Centers for Environmental Prediction (NCEP) Environmental Modeling Center (EMC) as a next generation analysis system based on the then operational Spectral Statistical Interpolation (SSI) analysis system ([4]; [1]; [2]). Instead of being constructed in spectral space like the SSI, the GSI is constructed in physical space and is designed to be a flexible, state-of-art system that is efficient on available parallel computing platforms. Starting with a three-dimensional variational (3DVar) data assimilation technique, current GSI can be run as a data assimilation system of 2DVar (for surface data analysis), 3DVar, 3D ensemble-variational (3D EnVar), 4D EnVar, 3D/4D hybrid EnVar, or 4DVar (if coupled with an adjoint model of GSI supported forecast systems).

After initial development, the GSI analysis system became operational as the core of the North American Data Assimilation System (NDAS) for the North American Mesoscale (NAM) system in June 2006 and the Global Data Assimilation System (GDAS) for the Global Forecast System (GFS) in May 2007 at National Oceanic and Atmospheric Administration (NOAA). Since then, the GSI system has been adopted in various operational systems, including the National Aeronautics and Space Administration (NASA) Goddard Earth Observing System Model (GEOS), the United States Air Force (USAF) mesoscale data assimilation system, the NOAA Real-Time Mesoscale Analysis (RTMA) system, the Hurricane Weather Research and Forecasting (WRF) model (HWRF), and the Rapid Refresh (RAP) and High Resolution Rapid Refresh (HRRR) system, etc. The number of groups and institutes involved in operational GSI development has also increased throughout these years.

1.2 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 ([3]). The DTC complements the development groups in providing GSI documentation, porting GSI to multiple platforms, and testing GSI in an independent and objective environment, while still maintaining functionally equivalent to operational centers. Working with EMC, the DTC is maintaining a community GSI repository, which is equivalent to the operational developmental repository, and facilitates community users to develop GSI. Based on the repository, the DTC releases GSI code annually with updated documentation. The first community version of the GSI system was released in 2009. This user's guide describes the 2016 release of GSI (v3.5) in July 2016. The DTC provides user support through the GSI Helpdesk (gsi-help@ucar.edu), and tutorials and workshops. More information about the GSI community services can be found at the DTC GSI webpage (<http://www.dtcenter.org/com-GSI/users/index.php>).

1.2.1 GSI Code Management and Review Committee

The GSI code development and maintenance are managed by the Data Assimilation Review Committee (DARC). It was originally formed as the GSI Review Committee in 2010, with a goal of incorporating all major GSI development teams in the United States within a unified community framework. In 2014, EMC and DTC decided to merge their GSI code repository with the code repository of the NOAA ensemble Kalman filter (EnKF) data assimilation system. Such a merging enabled coordinated development of both systems and joint community support. Following the repository merging, the GSI Review Committee was transitioned to DARC, incorporating new members representing the EnKF development and applications. Currently, DARC contains members from NCEP/EMC, NASA's Goddard Global Modeling and Assimilation Office (GMAO), NOAA's Earth System Research Laboratory (ESRL), the National Center for Atmospheric Research (NCAR) Mesoscale & Microscale Meteorology Laboratory (MMM), the National Environmental Satellite, Data, and Information Service (NESDIS), USAF, the University of Maryland, and the DTC (chair). The DTC also releases this EnKF system, along with GSI, annually. Please refer to the community EnKF user's webpage (<http://www.dtcenter.org/EnKF/users/index.php>) for more information.

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

- Coordination and advisory
 - Propose and shepherd new development
 - Coordinate on-going and new development
 - Establish and manage a code review and transition process
 - Community support recommendation
- Code review

1. Overview

- Establish and manage a unified coding standard followed by all GSI/EnKF developers
- 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

1.2.2 Community Code Contributions

GSI is a community data assimilation system, open to contributions from scientists and software engineers from both the operational and research communities. DARC oversees the code transition from prospective contributors. This committee reviews proposals for code commits to the GSI repository and monitors that coding standards and tests are being fulfilled. Once the committee reaches approval, the contributed code will be committed to the GSI code repository and available for operational implementation and public release.

To facilitate this process, the DTC is providing code transition assistance to the general research community. Prospective contributors of code to the GSI system should contact the DTC GSI helpdesk (gsi-help@ucar.edu) for the preparation and integration of their code. It is the contributor's responsibility to ensure that a proposed code change is correct, meets the GSI coding standards, and its expected impact is documented. The DTC will help the contributors run the regression tests and merge the code with the top of the repository trunk. Prospective contributors can also apply to the DTC visitor program for their GSI research and code transition. The visitor program is open to applications year-round. Please check the visitor program webpage (www.dtcenter.org/visitors/) for the latest announcement of opportunity and application procedures.

1.3 About This GSI Release

As a critical part of the GSI user support, this document is provided to assist users in applying GSI to data assimilation and analysis studies. It was composed by the DTC and reviewed by the DARC 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 and chemical applications are available with this release. This documentation describes the GSI v3.5 release. Active users can contact the DTC (gsi-help@ucar.edu) for developmental versions of GSI.

The GSI v3.5 can be used either as a 3DVar system, a 3D (hybrid) EnVar system, a 4D (hybrid) EnVar system. Currently, most of NOAA applications are using 3D hybrid EnVar system. Coupled with a forecast model and its adjoint model, GSI can be turned into a 4DVar system with embedded 4DVar features (e.g., GEOS).

1.3.1 What Is New in This Release Version

The following lists some of the new functions and changes included in the GSI release v3.5 versus the previous v3.4:

New observation types:

- All sky data assimilation option for Advanced Microwave Sounding Unit-A (AMSU-A)
- Assimilation of Geostationary Operational Environmental Satellite (GOES) clear air water vapor (CAWV) atmospheric motion vectors (AMV)
- Assimilation of Special Sensor Microwave Imager/Sounder (SSM/I/S) Defense Meteorological Satellite Program (DMSP) F19
- Initial capabilities for assimilating observations from the following instruments:
 - Global Change Observation Mission-W1 satellite (GCOM-W1) Advanced Microwave Scanning Radiometer 2 (AMSR2)
 - Global Precipitation Measurement (GPM) Microwave Imager (GMI)
 - Megha-Tropiques Sondeur Atmospherique du Profil D'Humidite Intertropicale par Radiometrie (SAPHIR)
 - Himawari Advanced Himawari Imager (AHI)
 - International Space Station Rapid Scatterometer (ISS-RapidScat)

Algorithm/application update:

- Added 4D (hybrid) EnVar option
- Added new AMV algorithm (requires a new BUFR table)
- Added QC for regional assimilation of GPS RO bending angle
- Updated observation thinning algorithms
- Added the capability of using blend global-regional coordinate with HWRF ensemble
- Added capability to output ensemble spread
- Updated to near surface sea temperature (NSST) capability
- Updated to RTMA capability

Libraries update:

- Switched to new version of the Community Radiative Transfer Model (CRTM) (v2.2.3)
- added capability to use the NCEP I/O library *nemio* for global GSI

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

1.3.2 Observations Used by This Version

GSI is being used by various applications on multiple scales. The types of observations GSI can assimilate vary from conventional to aerosol observations. Users should use observations

1. Overview

with caution to fit their specific applications. The GSI v3.5 can assimilate, but is not limited to, the following types of observations:

Conventional observations (including satellite retrievals):

- Radiosondes
- Pilot balloon (PIBAL) winds
- Synthetic tropical cyclone winds
- Wind profilers: USA, Jan Meteorological Agency (JMA)
- Conventional aircraft reports
- Aircraft to Satellite Data Relay (ASDAR) aircraft reports
- Meteorological Data Collection and Reporting System (MDCRS) aircraft reports
- Dropsondes
- Moderate Resolution Imaging Spectroradiometer (MODIS) IR and water vapor winds
- Geostationary Meteorological Satellite (GMS), JMA, and Meteosat cloud drift IR and visible winds
- European Organization for the Exploitation of Meteorological Satellites (EUMETSAT) and GOES water vapor cloud top winds
- GEOS hourly IR and cloud top wind
- Surface land observations
- Surface ship and buoy observation
- Special Sensor Microwave Imager (SSM/I) wind speeds
- Quick Scatterometer (QuikSCAT), the Advanced Scatterometer (ASCAT) and Oceansat-2 Scatterometer (OSCAT) wind speed and direction
- RapidScat observations
- SSM/I and Tropical Rainfall Measuring Mission (TRMM) Microwave Imager (TMI) precipitation estimates
- Velocity-Azimuth Display (VAD) Next Generation Weather Radar ((NEXRAD) winds
- Global Positioning System (GPS) precipitable water estimates
- Sea surface temperature (SST)
- Doppler wind Lidar
- Aviation routine weather report (METAR) cloud coverage
- Flight level and Stepped Frequency Microwave Radiometer (SFMR) High Density Observation (HDOB) from reconnaissance aircraft
- Tall tower wind

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

- SBUV: *NOAA-17, NOAA-18, NOAA-19*
- High Resolution Infrared Radiation Sounder (HIRS): *Meteorological Operational-A (MetOp-A), MetOp-B, NOAA-17, NOAA-19*
- GOES imager: *GOES-11, GOES-12*
- Atmospheric IR Sounder (AIRS): *aqua*
- AMSU-A: *MetOp-A, MetOp-B, NOAA-15, NOAA-18, NOAA-19, aqua*
- AMSU-B: *MetOp-B, NOAA-17*
- Microwave Humidity Sounder (MHS): *MetOp-A, MetOp-B, NOAA-18, NOAA-19*
- SSMI: *DMSP F14, F15, F19*
- SSMI/S: *DMSP F16*

1. Overview

- Advanced Microwave Scanning Radiometer for Earth Observing System (AMSR-E): *aqua*
- GOES Sounder (SNDR): *GOES-11, GOES-12, GOES-13*
- Infrared Atmospheric Sounding Interferometer (IASI): *MetOp-A, MetOp-B*
- Global Ozone Monitoring Experiment (GOME): *MetOp-A, MetOp-B*
- Ozone Monitoring Instrument (OMI): *aura*
- Spinning Enhanced Visible and Infrared Imager (SEVIRI): *Meteosat-8, Meteosat-9, Meteosat-10*
- Advanced Technology Microwave Sounder (ATMS): *Suomi NPP*
- Cross-track Infrared Sounder (CrIS): *Suomi NPP*
- GCOM-W1 AMSR2
- GPM GMI
- Megha-Tropiques SAPHIR
- Himawari AHI

Others:

- GPS Radio occultation (RO) refractivity and bending angle profiles
- Solar Backscatter Ultraviolet (SBUV) ozone profiles, Microwave Limb Sounder (MLS) (including NRT) ozone, and Ozone Monitoring Instrument (OMI) total ozone
- Doppler radar radial velocities
- Radar reflectivity Mosaic
- Tail Doppler Radar (TDR) radial velocity and super-observation
- Tropical Cyclone Vitals Database (TCVital)
- Particulate matter (PM) of 2.5-um diameter or less
- MODIS AOD (when using GSI-chem package)

Please note some of these above mentioned data are not yet fully tested and/or implemented for operations. Therefore, the current GSI code might not have the optimal setup for these data.



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 3.5. It builds and runs on most standard Linux platforms using either Intel, PGI, and Gnu compilers. Legacy build rules are provided for two platforms, the IBM AIX computers using the xlf compiler, and Intel based Macintosh computers using the PGI compiler. In both cases, the default build system must be modified to build on these platforms. See the community web page user support FAQ to get started.

This chapter describes how to build and install the DTC community GSI software on your local Linux computing resources. These instructions apply only to the DTC community GSI. While the community GSI source code is identical to the tag of the NCEP's GSI trunk code used for release, the community build system is different, allowing it to be more general 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

2. Software Installation

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 three of the compilers (Intel, PGI and Gnu) on the NCAR supercomputer Yellowstone. Section 2.6 covers the system requirements and settings (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 chapters provide background material for completeness. Advanced topics, such as customizing the build, porting to new platforms, and debugging can be found in the GSI Advanced User's Guide.

2.2 Obtaining and Setting Up the Source Code

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

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

The source code is available by first selecting the Download tab on the vertical menu located on the left column of the page, and then selecting the GSI/EnKF System submenu. New users must first register before downloading the source code. Returning users only need to enter their registration email address to log in. After accessing the download page, select the link to the `comGSIv3.5_EnKFv1.1.tar` tarball to download the most recent version of the source code (July 2016). Selecting the newest release of the community GSI is critical for having the most recent capabilities, versions of supplemental libraries, and bug fixes. Full support is only offered for the two most recent code releases.

To analyze satellite radiance observations, GSI requires use of CRTM coefficients. It is important to use **only** the version of CRTM coefficients provided by GSI website. Due to their large size, these are available as a separate tarfile. They can be downloaded by selecting the link to the tarball for the CRTM 2.2.3 Big Endian coefficients from the web page. For all compilers use the big endian byte order coefficients found in the first CRTM link.

The download page also contains links to the fixed files necessary for running global GSI:

- Global configuration (fix files to run Global GSI)

The community GSI version 3.5 comes in a tar file named `comGSIv3.5_EnKFv1.1.tar`. The tar file may be unpacked by using the UNIX commands:

```
gunzip comGSIv3.5_EnKFv1.1.tar.gz
tar -xvf comGSIv3.5_EnKFv1.1.tar
```

This creates the top level GSI directory `comGSIv3.5_EnKFv1.1/`. After downloading the source code, and prior to building, the user should check the known issues link on the

2. Software Installation

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, the 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 makefiles
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 the CRTM coefficients
include/	Include files created by the build system
lib/	Location for built supplemental libraries
run/	Directory for executable <code>gsi.exe</code> and sample run scripts
arch/	Build system support such as 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_2.2.3/	JCSDA community radiative transfer model
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)
w3ncv_v2.0.6	NCEP/NCV W3 library (date/time manipulation, GRIB)

The one nonstandard *library* not included with the source code are the WRF IO API's. These are obtained by linking to a build of the WRF code. Please note that the release version of WRF/EnKF has only been tested using the previous two release versions of WRF. Older versions of WRF may provide unpredictable results.

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

2. Software Installation

<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 Fortran compilers. This means that if WRF is built with the Intel Fortran compiler, then GSI must also be built with the Intel Fortran compiler. It is also recommended that both codes be built with the same annual version number of the compiler (.e.g. 12, 13, 14, 15, etc.).

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 your path variable.
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.

2. Software Installation

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 csh or tcsh
export WRF_DIR=/path_to_WRF_root_directory/ for ksh or bash
```

NETCDF The second environment variable specifies the local path to NetCDF library. The path location for **NETCDF** environment variable may be checked by using the command

```
echo $NETCDF
```

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

```
NETCDF: Undefined variable.
```

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

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

- Linux environments using Intel Fortran compiler.
- Building with Gfortran.
- On systems where the path variables are not properly set.
- On stripped down versions of the IBM AIX OS that lack the ESSL libraries

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

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

```
setenv LAPACK_PATH $MKLROOT
```

and for bash/ksh by

2. Software Installation

```
export LAPACK_PATH=$MKLRROOT
```

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

2.4.3 Configure and Compile

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

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

Change into the `comGSIv3.5_EnKFv1.1/` 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:

```
Will use NETCDF in dir: /glade/apps/opt/netcdf/4.3.0/intel/12.1.5
```

```
Will use WRF in dir: /glade/p/work/mhu/wrf/WRFV3.5
```

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

1. Linux x86_64, PGI compilers (pgf90 & pgcc) (dmpar,optimize)
2. Linux x86_64, PGI compilers (pgf90 & gcc) (dmpar,optimize)
3. Linux x86_64, PGI compilers (pgf90 & gcc) Supercomp (w/o -f90=SFC) (dmpar,optimize)
4. Linux x86_64, PGI compilers (pgf90 & pgcc) Supercomp (w/o -f90=SFC) (dmpar,optimize)
5. Linux x86_64, GNU compilers (gfortran & gcc) (dmpar,optimize)
6. Linux x86_64, Intel/gnu compiler (ifort & gcc) (dmpar,optimize)
7. Linux x86_64, Intel compiler (ifort & icc) (dmpar,optimize)
8. Linux x86_64, Intel compiler (ifort & icc), IBM POE (EXPERIMENTAL) (dmpar,optimize)
9. Linux x86_64, Intel compiler (ifort & icc), SGI MPT (EXPERIMENTAL) (dmpar,optimize)

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

The second thing to notice is that there are separate build targets for vendor supplied versions of MPI such as IBM POE and SGI MPT. This was added due to some computing hardware vendors creating non-standard `mpif90` wrappers for their vendor supplied version of MPI. If uncertain about which to choose, start by selecting the default option corresponding to the fortran compiler you wish to use (either 1,2,5,6, or 7). If that option fails with an error referencing a bad argument for `mpif90`, only then try the options listed for use with Supercomp, IBM POE, or SGI MPT.

2. Software Installation

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 in case it is necessary to diagnose build issues.

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 the Intel compiler, the PGI compiler, and the Gnu 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
```

2. Software Installation

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 Kernel 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 7. The alternative options (8 and 9) are needed for certain platforms that have a vendor supplied custom version of MPI. Try the default build options for MPI first, and only if it fails should the second option be used.
4. To compile the code, enter in a C-shell: `./compile >& compile.log`. If the build completes successfully, an executable named `gsi.exe` will be created in the `./run` directory.

2.5.2 PGI Build

Steps to build GSI on Yellowstone using the PGI compiler:

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

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

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

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

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

3. Similar to the Intel example, pick compiler options listed in a table. In this case, the desired compiler combination is either 3 or 4.
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.

2.5.3 GNU Build

Steps to build GSI on Yellowstone using the GNU compiler:

2. Software Installation

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

```
module load gnu/5.3.0
module load ncarcompilers ncarbinlibs netcdf lapack/3.2.1
```

These module commands have specified the compiler, mpi, 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 we will use the LAPACK library installed by the module. In a C-shell environment use:

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

3. Similar to the Intel example, pick compiler options listed in a table. In this case, the desired compiler combination is 5.
4. To compile the code, enter in a C-shell: `./compile >& compile.log`. If the build completes successfully, an executable named `gsi.exe` will be created in the `./run` directory.

2.6 System Requirements and External Libraries

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

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

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

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

2.6.1 Compilers Tested for Release

Version 3.5 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.

2. Software Installation

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 platforms, 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 only	ifort 16.0.1, 15.0.1, 13.0.1, 12.1.5, 12.1.4	icc
Intel & gcc	ifort 16.0.1, 15.0.1, 13.0.1, 12.1.5, 12.1.4	gcc 4.8.2, 4.4.7
PGI only	pgf90 16.1, 15.10, 15.7, 15.1, 14.10, 14.9, 14.7, 13.9, 13.3	pgcc
PGI & gcc	pgf90 16.1, 15.10, 15.7, 15.1, 14.10, 14.9, 14.7, 13.9, 13.3	gcc 4.8.2
GNU only	gfortran 6.3.0, 5.3.0	gcc 6.3.0, 5.3.0

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 specified paths in the configure file 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 Help Desk for assistance at

gsi-help@ucar.edu

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



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 3D EnVAR hybrid option, global or regional ensemble forecasts are also needed.

3.1.1 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

3. Running GSI

- e) GFS input fields in binary format or through NEMS I/O
- f) NEMS-NMMB input fields
- g) RTMA input files (2-dimensional binary format)
- h) WRF-Chem GOCART input fields with NetCDF format
- i) CMAQ binary file

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 (d). The DTC also supports the GSI in global and chemical applications with limited resources. The following backgrounds have been tested for the release:

1. ARW NetCDF (d) were tested with multiple cases
2. GFS (e) was tested with multiple NCEP cases
3. WRF-Chem NetCDF (h) was tested with a single case
4. NEMS-NMMB(f) was tested with a single case

3.1.2 Observations

GSI can analyze many types of observational data, including conventional data, satellite radiance observations, GPS Radio Occultations, 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 downloadable 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 User's Guide and examples on BUFR/PreBUFR processing:

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

DTC supports BUFR/PrepBUFR data 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 and 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

3. Running GSI

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.

3.1.3 Fixed Files (Statistics and Control Files)

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

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

Because most of those fixed files have hardwired names inside the GSI, a GSI run script needs to copy or link those files (right column in table 3.2) from `./fix` directory to GSI run directory with the file name required in GSI (left column in table 3.2). For example, if GSI runs with ARW background case, the following line should be in the run script:

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

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

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

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

3. Running GSI

Table 3.1: GSI observation file name, content, and examples

GSI Name	Content	Example file names
prepbuf	Conventional observations, including ps, t, q, pw, uv, spd, dw, sst	gdasl.t12z.prepbuf.nr
satwndbuf	satellite winds observations	gdasl.t12z.satwnd.tm00.buf_d
amsuabuf	AMSU-A 1b radiance (brightness temperatures) from satellites NOAA-15, 16, 17,18, 19 and METOP-A/B	gdasl.t12z.lbamua.tm00.buf_d
amsubbuf	AMSU-B 1b radiance (brightness temperatures) from satellites NOAA-15, 16,17	gdasl.t12z.lbamub.tm00.buf_d
radarbuf	Radar radial velocity Level 2.5 data	ndas.t12z.radwnd.tm12.buf_d
gpsrobuf	GPS radio occultation and bending angle observation	gdasl.t12z.gpsro.tm00.buf_d
ssmirrbuf	Precipitation rate observations from SSM/I	gdasl.t12z.spssmi.tm00.buf_d
tmirrbuf	Precipitation rate observations from TMI	gdasl.t12z.sptrmm.tm00.buf_d
sbuvbuf	SBUV/2 ozone observations from satellite NOAA-16, 17, 18, 19	gdasl.t12z.osbuv8.tm00.buf_d
hirs2buf	HIRS2 1b radiance from satellite NOAA-14	gdasl.t12z.lbhrs2.tm00.buf_d
hirs3buf	HIRS3 1b radiance observations from satellite NOAA-16, 17	gdasl.t12z.lbhrs3.tm00.buf_d
hirs4buf	HIRS4 1b radiance observation from satellite NOAA-18, 19 and METOP-A/B	gdasl.t12z.lbhrs4.tm00.buf_d
msubuf	MSU observation from satellite NOAA 14	gdasl.t12z.lbmsu.tm00.buf_d
airsbuf	AMSU-A and AIRS radiances from satellite AQUA	gdasl.t12z.airsev.tm00.buf_d
mhsbuf	Microwave Humidity Sounder observation from NOAA-18, 19 and METOP-A/B	gdasl.t12z.lbmhs.tm00.buf_d
ssmitbuf	SSMI observation from satellite fl3, fl4, fl5	gdasl.t12z.ssmi.tm00.buf_d
amsrebuf	AMSRE radiance from satellite AQUA	gdasl.t12z.amsre.tm00.buf_d
ssmisbuf	SSMIS radiances from satellite fl6	gdasl.t12z.ssmis.tm00.buf_d
gsndlbuf	GOES sounder radiance (sndrd1, sndrd2, sndrd3 sndrd4) from GOES-11, 12, 13, 14, 15.	gdasl.t12z.goesfv.tm00.buf_d
l2rwbuf	NEXRAD Level 2 radial velocity	ndas.t12z.nexrad.tm12.buf_d
gsndrbuf	GOES sounder radiance from GOES-11, 12	gdasl.t12z.goesnd.tm00.buf_d
gimgbuf	GOES imager radiance from GOE-11, 12	
omibuf	Ozone Monitoring Instrument (OMI) observation NASA Aura	gdasl.t12z.omi.tm00.buf_d
iasibuf	Infrared Atmospheric Sounding Interferometer sounder observations from METOP-A/B	gdasl.t12z.mtiasi.tm00.buf_d
gomebuf	The Global Ozone Monitoring Experiment (GOME) ozone observation from METOP-A/B	gdasl.t12z.gome.tm00.buf_d
mlsbuf	Aura MLS stratospheric ozone data from Aura	gdasl.t12z.mlsbuf.tm00.buf_d
tcvitl	Synthetic Tropic Cyclone-MSLP observation	gdasl.t12z.syndata.tcvitals.tm00
seviribuf	SEVIRI radiance from MET-08,09,10	gdasl.t12z.sevcsr.tm00.buf_d
atmsbuf	ATMS radiance from Suomi NPP	gdasl.t12z.atms.tm00.buf_d
crisbuf	CRIS radiance from Suomi NPP	gdasl.t12z.cris.tm00.buf_d
modisbuf	MODIS aerosol total column AOD observations from AQUA and TERRA	

3. Running GSI

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

Table 3.2: GSI fixed files, content, and examples

GSI Name	Content	Example file names
anavinfo	Information file to set control and analysis variables	anavinfo_arw_netcdf anavinfo_ndas_netcdf global_anavinfo.l64.txt
berror_stats	background error covariance	nam_nmmstat_na.gcv nam_glb_berror.f77.gcv global_berror.l64y386.f77
errtable	Observation error table	nam_errtable.r3dv prepobs_errtable.global
<i>Observation data control file (more detailed explanation in Section 4.3)</i>		
convinfo	Conventional observation information file	global_convinfo.txt nam_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
<i>Bias correction and Rejection list</i>		
satbias_angle	satellite scan angle dependent bias correction file	global_satangbias.txt
satbias_in	satellite mass bias correction coefficient file	sample.satbias
	combined satellite angle dependent and mass bias correction coefficient file	gdasl.t00z.abias.new
t_rejectlist, w_rejectlist,..	Rejection list for T, wind, et al. in RTMA	new_rtma_t_rejectlist new_rtma_w_rejectlist

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 `nam_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 set up 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 3 sub-directories to hold special group of fix files, which are introduced in table 3.3.

3. Running GSI

Table 3.3: List of sub-directories in fix directory

Directory name	Content
Little_Endian	Little Endian Background Error covariance (BE) files
Big_Endian	Big Endian BE files
global	Global BE files and ch4, co, co2, n2o history files

Please note released comGSIv3.5_EnKFv1.1 tar files dosen't include ./fix/global and ./fix/Little_Endian for space saving. Please download comGSIv3.5_EnKFv1.1_fix_global.tar.gz if you need to run global case, comGSIv3.5_EnKFv1.1_fix_Little_Endian.tar.gz if you need Little_endian BE files.

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 2.2.3. 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 FASTEM6.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 FASTEM6.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:

- comGSIv3.5_EnKFv1.1/run/run_gsi_regional.ksh for regional GSI
- comGSIv3.5_EnKFv1.1/run/run_gsi_global.ksh for global GSI (GFS)
- comGSIv3.5_EnKFv1.1/run/run_gsi_chem.ksh for chemical analysis

These scripts will be called to generate GSI namelists:

3. Running GSI

- `comGSIv3.5_EnKFv1.1/run/comgsi_namelist.sh` for regional GSI
- `comGSIv3.5_EnKFv1.1/run/comgsi_namelist_gfs.sh` for global GSI (GFS)
- `comGSIv3.5_EnKFv1.1/run/comgsi_namelist_chem.sh` for GSI chemical analysis

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

Note there is also a run script for regional EnKF (`run_enkf_wrf.ksh`), a run script for global EnKF (`run_enkf_global.ksh`) and the EnKF namelist script (`enkf_wrf_namelist.sh`) in the same directory, which will be introduced in the EnKF 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. Set the script that generates the GSI namelist.
5. Check the definitions of required variables.
6. Generate a run directory for GSI (sometimes called working or temporary directory).
7. Copy the GSI executable to the run directory.
8. Copy the background file to the run directory and create an index file listing the location and name of ensemble members if running the hybrid.
9. Link observations to the run directory.
10. Link fixed files (statistic, control, and coefficient files) to the run directory.
11. Generate namelist for GSI.
12. Run the GSI executable.
13. Post-process: save analysis results, generate diagnostic files, clean run directory.
14. Run GSI as observation operator for EnKF, only for `if_observer=Yes`.

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

3.2.2 Customization of the GSI Run Script

3.2.2.1 Setting Up the Machine Environment

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

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

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

Two queuing systems are listed below as examples:

Machine & queue system	Linux Cluster with LSF	Linux Cluster with PBS	Workstation
example	<pre>#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</pre>	<pre>#PBS -l procs=4 #PBS -n #PBS -o gsi.out #PBS -e gsi.err #PBS -N GSI #PBS -l walltime=00:20 #PBS -A ???????</pre>	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
```

3. Running GSI

```
#-----
GSIPROC=4
ARCH='LINUX_LSF'
# Supported configurations:
# IBM_LSF,
# LINUX, LINUX_LSF, LINUX_PBS,
# DARWIN_PGI
```

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

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

3.2.2.3 Setting Up an Analysis Case

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

```
#####
# case set up (users should change this part)
#####
#
# ANAL_TIME= analysis time (YYYYMMDDHH)
# WORK_ROOT= working directory, where GSI runs
# PREPBUF = path of PreBUFR conventional obs
# BK_FILE = path and name of background file
# OBS_ROOT = path of observations files
# FIX_ROOT = path of fix files
# GSI_EXE = path and name of the gsi executable
ANAL_TIME=2014061700
HH='echo $ANAL_TIME | cut -c9-10'
WORK_ROOT=comGSIv3.5_EnKFv1.1/run/testarw
OBS_ROOT=GSI_DTC/data/20140617/${ANAL_TIME}/obs
PREPBUF=${OBS_ROOT}/nam.t${HH}z.prepbufr.tm00.nr
BK_ROOT=GSI_DTC/data/20140617/${ANAL_TIME}/arw
BK_FILE=${BK_ROOT}/wrfinput_d01.${ANAL_TIME}
CRTM_ROOT=GSI_DTC/data/fix/CRTM_2.2.3
GSI_ROOT=comGSIv3.5_EnKFv1.1
FIX_ROOT=${GSI_ROOT}/fix
GSI_EXE=${GSI_ROOT}/run/gsi.exe
GSI_NAMELIST=${GSI_ROOT}/run/comgsi_namelist.sh
```

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

3. Running GSI

- 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). Appendix A.1 discusses the conversion tool `ssrc` to byte-swap observation files. Since the release version 3.2, GSI compiled with PGI and Intel can automatically handle the byte order issue in PrepBUFR and BUFR files. Users can directly link any order BUFR file if working with Intel and PGI platform.

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

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

Option `bk_core` indicates the specific dynamic core used to create the background files and is used to specify the core in the namelist. In this release, there is another `bk_core` option for NMMB in addition to WRF ARW and NMM, and also an option `if_observer=Yes` to run GSI as observation operator for EnKF. Option `bkcw_option` specifies the background error covariance to be used in the case. Two regional background error covariance matrices are provided with the release, one from NCEP global data assimilation (GDAS), and one from NAM data assimilation system (NDAS). Please check Section 4.8 for more details about GSI background error covariance. Option `if_clean` is to tell the run script if it needs to delete temporal intermediate files in the working directory after a GSI run is completed. Option `if_observer` is to tell the run script if it needs to run GSI as observation operator for EnKF.

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

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

3. Running GSI

3.2.3 Description of the Sample Regional Run Script to Run GSI

Listed below is an annotated regional run script 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)
#####

set -x
#
# GSIPROC = processor number used for GSI analysis
#-----
GSIPROC=8
ARCH='LINUX_LSF'

# Supported configurations:
# IBM_LSF,
# LINUX, LINUX_LSF, LINUX_PBS,
# DARWIN_PGI
#
#####
# case set up (users should change this part)
#####
#
# ANAL_TIME= analysis time (YYYYMMDDHH)
# WORK_ROOT= working directory, where GSI runs
# PREPBURF = path of PreBUFR conventional obs
# BK_FILE = path and name of background file
# OBS_ROOT = path of observations files
# FIX_ROOT = path of fix files
# GSI_EXE = path and name of the gsi executable
ANAL_TIME=2014061700
HH='echo $ANAL_TIME | cut -c9-10'
WORK_ROOT=run/testarw
OBS_ROOT=data/20140617/obs
PREPBURF=${OBS_ROOT}/nam.t${HH}z.prepbufr.tm00.nr
BK_ROOT=data/20140617/2014061700/arw
BK_FILE=${BK_ROOT}/wrfinput_d01.${ANAL_TIME}
CRTM_ROOT=data/CRTM_2.2.3
GSI_ROOT=code/comGSIv3.5_EnKFv1.1
FIX_ROOT=${GSI_ROOT}/fix
GSI_EXE=${GSI_ROOT}/run/gsi.exe
GSI_NAMELIST=${GSI_ROOT}/run/comgsi_namelist.sh

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

3. Running GSI

```
if_clean=clean
# if_observer = Yes : only used as observation operator for enkf
# no_member      number of ensemble members
# BK_FILE_mem    path and base for ensemble members
if_observer=No  # Yes, or, No -- case sensitive !
no_member=20
BK_FILE_mem=${BK_ROOT}/wrfarw.mem
```

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 set as Big_Endian because GSI compiled with Intel and PGI can read in Big_Endian background error file, BUFR file and CRTM coefficient files.

```
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
```

3. Running GSI

```
##### 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
```

3. Running GSI

```
echo "ERROR: ${GSI_EXE} does not exist!"
exit 1
fi

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

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

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

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

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

#
#####

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

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

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

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

# ln -s ${OBS_ROOT}/gdas1.t${HH}z.sptrmm.tm00.bufr_d tmirrbuf
# Link to the radiance data
ln -s ${OBS_ROOT}/gdas1.t${HH}z.1bamua.tm00.bufr_d amsuabuf
ln -s ${OBS_ROOT}/nam.t${HH}z.1bamub.tm00.bufr_d amsubbuf
ln -s ${OBS_ROOT}/nam.t${HH}z.1bhrs3.tm00.bufr_d hirs3buf
ln -s ${OBS_ROOT}/gdas1.t${HH}z.1bhrs4.tm00.bufr_d hirs4buf
ln -s ${OBS_ROOT}/nam.t${HH}z.1bmhs.tm00.bufr_d mhsbuf
ln -s ${OBS_ROOT}/nam.t${HH}z.gpsro.tm00.bufr_d gpsrobuf
```

3. Running GSI

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  
# speccoef = 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 precipitation rates  
# ozinfo = text file with information about assimilation of ozone data  
# errtable = text file with obs error for conventional data (regional only)  
# convinfo = text file with information about assimilation of conventional data  
# bufrtable= text file ONLY needed for single obs test (oneobstest=.true.)  
# bftab_sst= bufr table for sst ONLY needed for sst retrieval (retrieval=.true.)
```

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

```
if [ ${bkcv_option} = GLOBAL ] ; then  
  echo ' Use global background error covariance'  
  BERROR=${FIX_ROOT}/${BYTE_ORDER}/nam_glb_berror.f77.gcv  
  OBERROR=${FIX_ROOT}/prepobs_errtable.global  
  if [ ${bk_core} = NMM ] ; then  
    ANAVINFO=${FIX_ROOT}/anavinfo_ndas_netcdf_glbe  
  fi  
  if [ ${bk_core} = ARW ] ; then  
    ANAVINFO=${FIX_ROOT}/anavinfo_arw_netcdf_glbe  
  fi  
  if [ ${bk_core} = NMMB ] ; then  
    ANAVINFO=${FIX_ROOT}/anavinfo_nems_nmm_b_glb  
  fi  
else  
  echo ' Use NAM background error covariance'  
  BERROR=${FIX_ROOT}/${BYTE_ORDER}/nam_nmmstat_na.gcv  
  OBERROR=${FIX_ROOT}/nam_errtable.r3dv  
  if [ ${bk_core} = NMM ] ; then  
    ANAVINFO=${FIX_ROOT}/anavinfo_ndas_netcdf  
  fi  
  if [ ${bk_core} = ARW ] ; then  
    ANAVINFO=${FIX_ROOT}/anavinfo_arw_netcdf  
  fi  
  if [ ${bk_core} = NMMB ] ; then  
    ANAVINFO=${FIX_ROOT}/anavinfo_nems_nmm_b  
  fi  
fi
```

3. Running GSI

```
fi

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 $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=${CRTM_ROOT_ORDER}/FASTEM6.MWwater.EmisCoeff.bin
aercoef=${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 ./FASTEM6.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
bufhtable=${FIX_ROOT}/prepobs_prep.bufhtable
cp $bufhtable ./prepobs_prep.bufhtable

# for satellite bias correction
cp ${FIX_ROOT}/gdas1.t00z.abias.20150617 ./satbias_in
cp ${FIX_ROOT}/gdas1.t00z.abias_pc.20150617 ./satbias_pc
```

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

3. Running GSI

```
cp ${FIX_ROOT}/gdas1.t00z.abias.20150617 ./satbias_in
cp ${FIX_ROOT}/gdas1.t00z.abias_pc.20150617 ./satbias_pc
```

There are two options on how to perform the radiance bias correction. The first method is to do the angle dependent bias correction offline and do the mass bias correction inside the GSI analysis, therefore requiring two input files: `satbias_angle` corresponding to angle dependent bias correction file and `satbias_in` being the input file for mass bias correction. The second method is to combine the angle dependent and mass bias correction together and do it within the GSI analysis, requiring one combined input file `satbias_in`. Note that the input bias correction coefficients file `satbias_in` are different for the two options, therefore it is important to use the appropriate input file for each method. The sample input files for the first method are provided with this release `global_satangbias.txt` and `sample.satbias`. For using the second option - combined angle dependent and mass bias correction, a sample file `gdas1.t00z.abias_pc.20150617` is also provided. Users, as a starting point, might also download a GDAS `satbias` coefficient file from the NOMADS ftp site as the input file (starting spring 2015, the GDAS `satbias` files have adopted the new format):

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

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

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

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

# default is NAM
# as_op='1.0,1.0,0.5 ,0.7,0.7,0.5,1.0,1.0,'
vs_op='1.0,'
hzscl_op='0.373,0.746,1.50,'
if [ ${bkcv_option} = GLOBAL ] ; then
# as_op='0.6,0.6,0.75,0.75,0.75,0.75,1.0,1.0'
vs_op='0.7,'
hzscl_op='1.7,0.8,0.5,'
fi
if [ ${bk_core} = NMMB ] ; then
vs_op='0.6,'
fi

# default is NMM
bk_core_arw='.false.'
```


3. Running GSI

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

The following section specifies the number of outer loops and whether to save GSI read observations based on the setting of "if_observer".

```
if [ ${if_observer} = Yes ] ; then  
nummiter=0  
if_read_obs_save='.true.'  
if_read_obs_skip='.false.'  
else  
nummiter=2  
if_read_obs_save='.false.'  
if_read_obs_skip='.false.'  
fi
```

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

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

Note: EOF indicates the end of GSI namelist.

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

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

3. Running GSI

```
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 result.

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

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

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

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

3. Running GSI

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

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

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

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

The following scripts clean the temporal intermediate files

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

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

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

3. Running GSI

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

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

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

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

```
# Loop through each member  
loop="01"  
ensmem=1  
while [[ $ensmem -le $no_member ]];do  
  
  rm pe0*  
  
  print "\$ensmem is $ensmem"  
  ensmemid='printf %3.3i $ensmem'  
  
# get new background for each member  
  if [[ -f wrf_inout ]]; then  
    rm wrf_inout  
  fi  
  
  BK_FILE=${BK_FILE_mem}${ensmemid}  
  echo $BK_FILE  
  ln -s $BK_FILE wrf_inout  
  
# run GSI  
  echo ' Run GSI with' ${bk_core} 'for member ', ${ensmemid}  
  
  case $ARCH in  
    'IBM_LSF')  
      ${RUN_COMMAND} ./gsi.exe < gsiparm.anl > stdout_mem${ensmemid} 2>&1 ;;
```

3. Running GSI

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

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

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

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

The following lines generate the diagnostics files for each member.

```
# generate diag files

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

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

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

done

fi
```

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

```
exit 0
```

3.3 GSI Analysis Result Files in Run Directory

Once the GSI run script is set up, it is ready to be submitted just as other batch jobs. When completed, GSI will create a number of files in the run directory. Below is an example of the files generated in the run directory from one of the GSI test case runs. This case was run to perform a regional GSI analysis with a WRF ARW NetCDF background using conventional (prepbufr), radiance (AMSU-A, HIRS4, and MHS), and GPSRO data. The analysis time is

3. Running GSI

00Z 17 June 2014. Four processors were used. To make the run directory more readable, we turned on the clean option in the run script, which deleted all temporary intermediate files.

amsuabufr	diag_mhs_n19_ges.2014061700	fort.226
amsubbuf	errtable	fort.227
anavinfo	fit_p1.2014061700	fort.228
berror_stats	fit_q1.2014061700	fort.229
convinfo	fit_rad1.2014061700	fort.230
diag_amsua_metop-a_anl.2014061700	fit_t1.2014061700	gpsrobuf
diag_amsua_metop-a_ges.2014061700	fit_w1.2014061700	gsi.exe
diag_amsua_metop-b_anl.2014061700	fort.201	gsiparm.anl
diag_amsua_metop-b_ges.2014061700	fort.202	hirs4buf
diag_amsua_n15_anl.2014061700	fort.203	l2rwbufr
diag_amsua_n15_ges.2014061700	fort.204	list_run_directory
diag_amsua_n18_anl.2014061700	fort.205	mhsbuf
diag_amsua_n18_ges.2014061700	fort.206	ozinfo
diag_amsua_n19_anl.2014061700	fort.207	pcpbias_out
diag_amsua_n19_ges.2014061700	fort.208	pcpinfo
diag_conv_anl.2014061700	fort.209	prepbufr
diag_conv_ges.2014061700	fort.210	prepobs_prep.bufhtable
diag_hirs4_metop-a_anl.2014061700	fort.211	satbias_ang.out
diag_hirs4_metop-a_ges.2014061700	fort.212	satbias_in
diag_hirs4_metop-b_anl.2014061700	fort.213	satbias_out
diag_hirs4_metop-b_ges.2014061700	fort.214	satbias_out.int
diag_hirs4_n19_anl.2014061700	fort.215	satbias_pc
diag_hirs4_n19_ges.2014061700	fort.217	satbias_pc.out
diag_mhs_metop-a_anl.2014061700	fort.218	satinfo
diag_mhs_metop-a_ges.2014061700	fort.219	stdout
diag_mhs_metop-b_anl.2014061700	fort.220	stdout.anl.2014061700
diag_mhs_metop-b_ges.2014061700	fort.221	wrf_inout
diag_mhs_n18_anl.2014061700	fort.223	wrfanl.2014061700
diag_mhs_n18_ges.2014061700	fort.224	
diag_mhs_n19_anl.2014061700	fort.225	

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

- *stdout.anl.2014061700/stdout*: standard text output file, which is a link to stdout with the analysis time appended. This is the most commonly used file to check the GSI analysis processes as well as basic and important information about the analyses. We will explain the contents of stdout in Section 4.1 and users are encouraged to read this file in detail to become familiar with the order of GSI analysis processing.
- *wrfanl.2014061700/wrf_inout*: analysis results if GSI completes successfully it exists only if using WRF for background. This is a link to *wrf_inout* with the analysis time appended. The format is the same as the background file.
- *diag_conv_anl.(time)*: binary diagnostic files for conventional and GPS RO observations at the final analysis step (analysis departure for each observation).
- *diag_conv_ges.(time)*: binary diagnostic files for conventional and GPS RO observations before initial analysis step (background departure for each observation)
- *diag(instrument_satellite)_anl*: diagnostic files for satellite radiance observations at final analysis step.
- *diag(instrument_satellite)_ges*: diagnostic files for satellite radiance observations before initial analysis step.
- *gsiparm.anl*: GSI namelist, generated by the run script.
- *fit(variable).(time)*: links to fort.2?? with meaningful names (variable name plus analysis time). They are statistic results of observation departures from background and

3. Running GSI

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).
- **buf*: observation BUFR files linked to the run directory Please see Section 3.1 for details.
- *satbias_in*: the input coefficients of bias correction for satellite radiance observations.
- *satbias_out*: the output coefficients of bias correction for satellite radiance observations after the GSI run.
- *satbias_pc*: the input coefficients of bias correction for passive 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_(instrumen_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 table 3.5 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.)

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:

3.4.1 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:

3. Running GSI

Table 3.5: The list of GSI intermediate files

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 instru- ment_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

- `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, or reaches the convergence threshold, 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.

3.4.2 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.

3.4.3 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

3. Running GSI

the background. If false, perform a global GSI analysis. If either `wrf_nmm_regional` or `wrf_mass_regional` are true, it will be set to true.

- `wrf_nmm_regional`: if true, background comes from WRF NMM. When using other background fields, set it to false.
- `wrf_mass_regional`: if true, background comes from WRF ARW. When using other background fields, set it to false.
- `nems_nmmb_regional`: if true, background comes from NMMB. When using other background fields, set it to false.
- `netcdf`: if true, WRF files are in NetCDF format, otherwise WRF files are in binary format. This option only works for performing a regional GSI analysis.

3.4.4 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.

3.4.5 Set Up the GSI Recognized Observation Files

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

```
OBS_INPUT::
! dfile      dtype      dplat      dsis      dval      dthin dsfcalc
  prepbufr   ps          null       ps        1.0      0      0
  prepbufr   t          null       t         1.0      0      0
  prepbufr   q          null       q         1.0      0      0
  prepbufr   pw         null       pw        1.0      0      0
  satwndbufr uv         null       uv        1.0      0      0
  prepbufr   uv         null       uv        1.0      0      0
  prepbufr   spd        null       spd       1.0      0      0
  prepbufr   dw         null       dw        1.0      0      0
  radarbufr  rw         null       rw        1.0      0      0
  prepbufr   sst        null       sst       1.0      0      0
  gpsrobufr  gps_ref    null       gps       1.0      0      0
  ssmirrbufr pcp_ssmi   dmsp      pcp_ssmi  1.0     -1      0
```

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

3. Running GSI

- `dtype`: analysis variable name that GSI can read in and handle. Please note this name should be consistent with that used in the GSI code.
- `dplat`: sets up the observation platform for a certain observation, which will be read in from the file `dfile`.
- `dsis`: sets up data name (including both data type and platform name) used inside GSI.

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

3.4.6 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.

3.4.7 Set Up Data Thinning

1) Radiance data thinning

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

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

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

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

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

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

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

3. Running GSI

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

2) Conventional data thinning

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

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

3.4.8 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

3.4.9 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.



4

GSI Diagnostics and Tuning

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

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

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

4.1 Understanding Standard Output (*stdout*)

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

4. GSI Diagnostics and Tuning

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:
 - Read in configuration (namelist)
 - Read in background
 - Read in observations
 - Partition domain and data for parallel analysis
 - Read in constant fields (fixed files)
2. Calculate observation innovations
3. Optimal iteration (analysis)
4. Save analysis result

In this section, the detailed structure and content of **stdout** are explained using the v3.5 online example case: 2014061700. To keep the output concise and make it more readable, most repeated content was deleted (shown by the dotted line). For the same reason, the accuracy of some numbers has been reduced to avoid line breaks in **stdout**.

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

```
* . . . . .  
PROGRAM GSI_ANL HAS BEGUN. COMPILED 1999232.55   ORG: NP23  
STARTING DATE-TIME  JUL 02,2016  20:36:21.760 184 SAT  2457572
```

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

```
gsi_metguess_mod*init_: 2D-MET STATE VARIABLES:  
ps  
z  
gsi_metguess_mod*init_: 3D-MET STATE VARIABLES:  
u  
v  
div  
vor  
tv  
q  
oz  
cw  
gsi_metguess_mod*init_: ALL MET STATE VARIABLES:  
u  
v  
div  
vor  
tv  
q  
oz  
cw  
ps  
z  
state_vectors*init_anasv: 2D-STATE VARIABLES ps  
sst  
state_vectors*init_anasv: 3D-STATE VARIABLES u  
v          tv  
tsen      q  
oz        cw
```

4. GSI Diagnostics and Tuning

```
prse
state_vectors*init_anasv: ALL STATE VARIABLES u
v                tv
tsen             q
oz              cw
prse            ps
sst
control_vectors*init_anacv: 2D-CONTROL VARIABLES ARE
ps              sst
control_vectors*init_anacv: 3D-CONTROL VARIABLES ARE
sf             vp
t              q
oz            cw
control_vectors*init_anacv: MOTLEY CONTROL VARIABLES
stl           sti
control_vectors*init_anacv: ALL CONTROL VARIABLES
sf            vp
ps            t
q             oz
sst          cw
stl          sti
```

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

```
GSI_4DVAR: nobs_bins =          1
SETUP_4DVAR: l4dvar= F
SETUP_4DVAR: l4densvar= F
SETUP_4DVAR: winlen=  3.000000000000000
SETUP_4DVAR: winoff=  3.000000000000000
SETUP_4DVAR: hr_obsbin=  3.000000000000000
SETUP_4DVAR: nobs_bins=          1
SETUP_4DVAR: ntlevs_ens=          1
SETUP_4DVAR: nsubwin,nhr_subwin=          1          3
SETUP_4DVAR: lsqrtb= F
SETUP_4DVAR: lbicg= F
SETUP_4DVAR: lcongrad= F
SETUP_4DVAR: lbfgsmin= F
SETUP_4DVAR: ltlint= F
SETUP_4DVAR: ladtest,ladtest_obs,lgrtest= F F F
SETUP_4DVAR: iwrtinc=          -1
SETUP_4DVAR: lanczosave= F
SETUP_4DVAR: ltcost= F
SETUP_4DVAR: jsiga=          -1
SETUP_4DVAR: nwrvecs=          -1
SETUP_4DVAR: iorthomax=          0
SETUP_4DVAR: liauon= F
SETUP_4DVAR: ljc4tlevs= F
SETUP_4DVAR: ibin_anl=          1
in gsimod: use_gfs_stratosphere,nems_nmmb_regional,wrf_nmm_regional= F F F
GSIMOD: ***WARNING*** set l_cloud_analysis=false
INIT_OBSMOD_VARS: reset time window for one or more OBS_INPUT entries to
  1.500000000000000
INIT_OBSMOD_VARS: ndat_times,ndat_types,ndat=          1          81
  81
INIT_OBSMOD_VARS: nhr_assimilation=          3
GSIMOD: ***WARNING*** reset oberrflg= T
calling gsisub with following input parameters:

&SETUP
GENCODE = 78.00000000000000 ,
FACTQMIN = 0.000000000000000E+000,
FACTQMAX = 0.000000000000000E+000,
CLIP_SUPERSATURATION = F,
FACTV = 1.000000000000000 ,
```

4. GSI Diagnostics and Tuning

```
FACTL = 1.0000000000000000 ,
FACTP = 1.0000000000000000 ,
FACTG = 1.0000000000000000 ,
FACTW10M = 1.0000000000000000 ,
FACTHOWV = 1.0000000000000000 ,
R_OPTION = F,
DELTIM = 1200.000000000000 ,
DTPHYS = 3600.000000000000 ,
BIASCOR = -1.0000000000000000 ,
BCOPTION = 1,
DIURNALBC = 0.0000000000000000E+000,
NITER = 0, 2*50, 48*0,
NITER_NO_QC = 51*1000000,
MITER = 2,
QOPTION = 2,
CWOPTION = 0,
NHR_ASSIMILATION = 3,
MIN_OFFSET = 180,
PSEUDO_Q2 = F,
IOUT_ITER = 220,
NPREDP = 6,
```

```
/
&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 option FGAT (First Guess at Appropriate Time), however we only have provided one in this test case. Therefore, there is error information at the beginning of the reading background portion:

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

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

Next, the background fields for the analysis are read in and the maximum, minimum and median values of the fields at each vertical level are displayed. Here, only part of the variables ZNU and T are shown, and all other variables read by the GSI are listed only as the variable name in the NetCDF file (`rmse_var = T`). 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.

4. GSI Diagnostics and Tuning

```

dh1 = 3
iy,m,d,h,m,s= 2014 6 17 0 0
0
dh1 = 3
rmse_var = SMOIS
ndim1 = 3
ordering = XYZ
staggering = N/A
start_index = 1 1 1 0
end_index = 332 215 4 0
WrfType = 104
ierr = 0
rmse_var = T ndim1 = 3 dh1 = 3
WrfType = 104 ierr = 0
ordering = XYZ staggering = N/A
start_index = 1 1 1 0
end_index = 332 215 50 0
nlon,lat,sig_regional= 332 215 50
rmse_var = P_TOP ndim1= 0
WrfType = 104 WRF_REAL= 104 ierr = 0
ordering = 0 staggering = N/A
start_index = 1 1 1 0
end_index = 332 215 50 0
p_top= 2000.000
...
...
rmse_var = ZNU ndim1= 1
WrfType = 104 WRF_REAL= 104 ierr = 0
ordering = Z staggering = N/A
start_index = 1 1 1 0
end_index = 50 215 50 0
k,znu(k)= 1 0.9990000
k,znu(k)= 2 0.9960001
k,znu(k)= 3 0.9905000
...
...
k,znu(k)= 49 7.1999999E-03
k,znu(k)= 50 2.3500000E-03
rmse_var = ZNW ndim1= 1
...
rmse_var = RDX ndim1= 0
...
rmse_var = RDY ndim1= 0
...
rmse_var = MAPFAC_M ndim1= 2
...
rmse_var = XLAT ndim1= 2
...
rmse_var = XLONG ndim1= 2
...
rmse_var = MUB ndim1= 2
...
rmse_var = MU ndim1= 2
...
rmse_var = PHB ndim1= 3
...
rmse_var = T ndim1= 3
WrfType = 104 WRF_REAL= 104 ierr = 0
ordering = XYZ staggering = N/A
start_index = 1 1 1 0
end_index = 332 215 50 0
k,max,min,mid T= 1 321.5280 270.7682 309.0504
k,max,min,mid T= 2 321.6272 270.9064 309.1002
k,max,min,mid T= 3 321.4596 271.1610 309.1918
k,max,min,mid T= 4 321.2505 271.6038 309.3501
k,max,min,mid T= 5 321.6713 272.2668 309.4191
...
...
k,max,min,mid T= 48 632.2557 567.8249 596.6701
k,max,min,mid T= 49 659.2219 604.4777 630.4330
k,max,min,mid T= 50 689.7565 646.8995 668.5146
rmse_var = QVAPOR ndim1= 3
...

```


4. GSI Diagnostics and Tuning

```
rmse_var = U ndim1=      3
...
rmse_var = V ndim1=      3
...
rmse_var = XLAND ndim1=   2
...
rmse_var = SEAICE ndim1=  2
...
rmse_var = SST ndim1=     2
...
rmse_var = IVGTYP ndim1=  2
...
rmse_var = ISLTYP ndim1=  2
...
rmse_var = VEGFRA ndim1=  2
...
rmse_var = SNOW ndim1=    2
...
rmse_var = U10 ndim1=     2
...
rmse_var = V10 ndim1=     2
...
rmse_var = SMOIS ndim1=   3
...
rmse_var = TSLB ndim1=    3
...
rmse_var = TSK ndim1=     2
...
rmse_var = Q2 ndim1=      2
...
rmse_var = QCLOUD ndim1=  3
...
rmse_var = QRAIN ndim1=   3
...
rmse_var = QSNOW ndim1=   3
...
rmse_var = QICE ndim1=    3
...
rmse_var = QGRAUP ndim1=  3
...
rmse_var = QNRAIN ndim1=  3
...
rmse_var = RAD_TTEN_DFI ndim1= 3
...
```

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

```
rmse_var = QSNOW ndim1=      3
WrfType =          104 WRF_REAL=          104 ierr =          -1021
ordering = XYZ staggering = N/A
start_index =          1          1          1          0
end_index =          332          215          50          0
NetCDF error: NetCDF: Variable not found
NetCDF error: NetCDF: Variable not found
NetCDF error in wrf_io.F90, line          2842 Varname QSNOW
NetCDF error in wrf_io.F90, line          2842 Varname QSNOW
```

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

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

4. GSI Diagnostics and Tuning

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

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

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

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

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

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

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

```
[000]gsisub(): : starting ...
READ_wrf_mass_FILES: analysis date,minutes      2014      6
17          0          0      19175040
READ_wrf_mass_FILES: sigma guess file, nming2  0.0000000000000000E+000
2014        6          17          0      19175040
READ_wrf_mass_FILES: sigma fcst files used in analysis :      3
3.0000000000000000          1
READ_wrf_mass_FILES: surface fcst files used in analysis:      3
3.0000000000000000          1
GESINFO: Guess date is          0          6          17          2014
0.0000000000000000E+000
GESINFO: Analysis date is      2014          6          17          0
0 2014061700 3.0000000000000000
using restart file date =      2014          6          17          0
```

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

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

Read in information from fix file *scaninfo* (see table 3.2) and *pcpinfo* (see table 3.2).

4. GSI Diagnostics and Tuning

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

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

```
READ_CONVINFO: tcp      112  0  1  3.00000      0  0  0  75.0000  5.00000  1.00000  75.0000
0.00000      0  0.00000  0.00000  0  0.00000  0.00000  2
READ_CONVINFO: ps      120  0  1  3.00000      0  0  0  4.00000  3.00000  1.00000  4.00000
0.300000E-03  0  0.00000  0.00000  0  0.00000  0.00000  2
...
READ_CONVINFO: t       120  0  1  3.00000      0  0  0  8.00000  5.60000  1.30000  8.00000
0.100000E-05  0  0.00000  0.00000  0  0.00000  0.00000  2
READ_CONVINFO: t       126  0 -1  3.00000      0  0  0  8.00000  5.60000  1.30000  8.00000
0.100000E-02  0  0.00000  0.00000  0  0.00000  0.00000  2
...
READ_CONVINFO: gps     729  0 -1  3.00000      0  0  0  10.0000  10.0000  1.00000  10.0000
0.00000      0  0.00000  0.00000  0  0.00000  0.00000  2
READ_CONVINFO: gps     44  0 -1  3.00000      0  0  0  10.0000  10.0000  1.00000  10.0000
0.00000      0  0.00000  0.00000  0  0.00000  0.00000  2
```

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

```
glbsoi: starting ...
gsi_metguess_mod*create_: alloc() for met-guess done
guess_grids*create_chemges_grids: trouble getting number of chem/gases
  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 observation observer is successfully initialized and inquire about the control vectors (space for analysis variables).

```
observer_init: successfully initialized
control_vectors: length= 5613648
control_vectors: currently allocated= 0
control_vectors: maximum allocated= 0
control_vectors: number of allocates= 0
control_vectors: number of deallocates= 0
control_vectors: Estimated max memory used= 0.0 Mb
```

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

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

4. GSI Diagnostics and Tuning

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 checks the file status on whether the observation time matches the analysis time and how namelist option *offtime_data* is set (can be used to turn off the time consistent check between observation and analysis time). This step also checks for consistency between the satellite radiance data types in the BUFR files and the usage setups in the *satinfo* files. The following shows stdout information from this step:

```

read_obs_check: bufr file date is 2014061700 prepbufr ps
read_obs_check: bufr file uv not available satwndbufr
read_obs_check: bufr file rw not available radarbufr
read_obs_check: bufr file pcp_tmi trmm not available tmirrbufr
read_obs_check: bufr file hirs3 n17 not available hirs3bufr
read_obs_check: bufr file goes_img g11 not available gimgrbufr
read_obs_check: bufr file date is 2014061700 prepbufr q
read_obs_check: bufr file date is 2014061700 prepbufr t
read_obs_check: bufr file date is 2014061700 amsuabufr amsua n18
read_obs_check: bufr file amsua n18 not available amsuabufrears

...
...

read_obs_check: bufr file sndrd3 g15 not available gsnd1bufr
read_obs_check: bufr file omi aura not available omibufr
read_obs_check: bufr file seviri m09 not available seviribufr
read_obs_check: bufr file atms npp not available atmsbufr
read_obs_check: bufr file date is 2014061700 prepbufr sst

...
...

read_obs_check: bufr file seviri m08 not available seviribufr
read_obs_check: bufr file gome metop-b not available gomebufr
read_obs_check: bufr file uv not available oscatbufr
data type mta_cld not used in info file -- do not read file
prepbufr
data type gos_ctp not used in info file -- do not read file
prepbufr
data type rad_ref not used in info file -- do not read file
refInGSI
data type lghtn not used in info file -- do not read file
lghtInGSI
data type larcld not used in info file -- do not read file
larcInGSI

```

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

number of	extra	processors						
		1						
READ_OBS:	read	33 mhs	mhs_n18	using ntasks=	2	0	131	0
READ_OBS:	read	34 mhs	mhs_n19	using ntasks=	2	2	153	0
READ_OBS:	read	35 mhs	mhs_metop-a	using ntasks=	2	0	563	0
READ_OBS:	read	36 mhs	mhs_metop-b	using ntasks=	2	2	2	0
READ_OBS:	read	1 ps	ps	using ntasks=	1	0	0	0
READ_OBS:	read	2 t	t	using ntasks=	1	1	0	0
READ_OBS:	read	3 q	q	using ntasks=	1	2	0	0
READ_OBS:	read	4 pw	pw	using ntasks=	1	3	839	0
READ_OBS:	read	6 uv	uv	using ntasks=	1	0	0	0
READ_OBS:	read	10 sst	sst	using ntasks=	1	1	504	0
READ_OBS:	read	11 gps_ref	gps	using ntasks=	1	2	0	0

4. GSI Diagnostics and Tuning

```

READ_OBS: read 19 hirs4      hirs4_metop-a      using ntasks= 2 3 277 0
READ_OBS: read 21 hirs4      hirs4_n19           using ntasks= 1 0 75 0
READ_OBS: read 22 hirs4      hirs4_metop-b      using ntasks= 1 1 2 0
READ_OBS: read 26 amsua      amsua_n15           using ntasks= 1 2 27 0
READ_OBS: read 27 amsua      amsua_n18           using ntasks= 1 3 45 0
READ_OBS: read 28 amsua      amsua_n19           using ntasks= 1 0 47 0
READ_OBS: read 29 amsua      amsua_metop-a      using ntasks= 1 1 124 0
READ_OBS: read 30 amsua      amsua_metop-b      using ntasks= 1 2 2 0

```

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

```

GETSFC: enter with nlat_sfc,nlon_sfc=          0          0 and nlat,nlon=
          215          332
GETSFC: set nlat_sfc,nlon_sfc=          215          332
=====
Status  Var          Mean          Min          Max
sfcges2 FC10      1.000000000000E+00  1.000000000000E+00  1.000000000000E+00
sfcges2 SNOW      8.137817211798E-02  0.000000000000E+00  9.510296630859E+01
sfcges2 VFRC      1.701514588514E-01  0.000000000000E+00  9.899999499321E-01
sfcges2 SRGH      5.003234230527E-02  5.000000074506E-02  5.000000074506E-02
sfcges2 STMP      2.936729335948E+02  2.643117675781E+02  3.229424743652E+02
sfcges2 SMST      7.664003944557E-01  6.047149747610E-02  1.000000000000E+00
sfcges2 SST       2.942266741384E+02  2.688000183105E+02  3.240092468262E+02
sfcges2 VTYP      1.463281031101E+01  1.000000000000E+00  2.400000000000E+01
sfcges2 ISLI      3.405295601009E-01  0.000000000000E+00  2.000000000000E+00
sfcges2 STYP      1.137135051835E+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=mhsbufr      type=mhs      sis=mhs_n18      nread= 248485  ithin= 2
rmesh= 60.000000 isfcalc= 0 ndata= 26765 ntask= 2
READ_BUFRTOVS      : file=mhsbufr      type=mhs      sis=mhs_n19      nread= 60900  ithin= 2
rmesh= 60.000000 isfcalc= 0 ndata= 6725 ntask= 2
READ_BUFRTOVS      : file=mhsbufr      type=mhs      sis=mhs_metop-a  nread= 142555 ithin= 2
rmesh= 60.000000 isfcalc= 0 ndata= 15145 ntask= 2
READ_BUFRTOVS      : file=mhsbufr      type=mhs      sis=mhs_metop-b  nread= 113590 ithin= 2
rmesh= 60.000000 isfcalc= 0 ndata= 12185 ntask= 2
READ_PREPBUFR: messages/reports = 142 / 20925 ntreadd =
1
new vad flag:: F
READ_PREPBUFR      : file=prepbufr      type=pw      sis=pw      nread= 252  ithin= 0  rmesh=
120.000000 isfcalc= 0 ndata= 252 ntask= 1
READ_PREPBUFR: messages/reports = 682 / 67083 ntreadd =
1
READ_PREPBUFR: time offset is 3.0000000000000000 hours.
new vad flag:: F
READ_PREPBUFR: messages/reports = 682 / 67083 ntreadd =
1
mesonetuselist: listexist,nprov= F 0
w_rejectlist: wlistexist,nwrjs= F 0
t_rejectlist: tlistexist,ntrjs= F 0
...
READ_PREPBUFR      : file=prepbufr      type=ps      sis=ps      nread= 23868  ithin= 0  rmesh=
120.000000 isfcalc= 0 ndata= 15819 ntask= 1
READ_PREPBUFR      : file=prepbufr      type=t      sis=t      nread= 26296  ithin= 0  rmesh=
120.000000 isfcalc= 0 ndata= 25686 ntask= 1
READ_PREPBUFR      : file=prepbufr      type=sst     sis=sst     nread= 0  ithin= 0  rmesh=
120.000000 isfcalc= 0 ndata= 0 ntask= 1
READ_PREPBUFR      : file=prepbufr      type=q      sis=q      nread= 24461  ithin= 0  rmesh=
120.000000 isfcalc= 0 ndata= 20989 ntask= 1
..
READ_BUFRTOVS      : file=hirs4bufr     type=hirs4   sis=hirs4_n19   nread= 55613  ithin= 2  rmesh=
60.000000 isfcalc= 0 ndata= 23408 ntask= 1
READ_BUFRTOVS      : file=amsuabufr     type=amsua   sis=amsua_n19   nread= 20370  ithin= 2
rmesh= 60.000000 isfcalc= 0 ndata= 16912 ntask= 1

```

Using the above output information, many details on the observations can be obtained. For example, the last line indicates that subroutine *READ_BUFRTOVS* was called to read

4. GSI Diagnostics and Tuning

in NOAA-19 AMSU-A (sis=amsua_n19) from the BUFR file *amsuabufr* (file=amsuabufr). Furthermore, this kind of data has 20370 observations in the file (nread=20370) and 16912 in analysis domain and time-window (ndata=16912). 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		1429	3190	4655	6774
OBS_PARA: t		2564	5200	7057	11128
OBS_PARA: q		2346	4626	6148	8128
OBS_PARA: pw		65	80	63	49
OBS_PARA: uv		3358	6453	8091	11998
OBS_PARA: gps_ref		1799	1368	2664	3520
OBS_PARA: hirs4	metop-a	0	0	1146	1661
OBS_PARA: hirs4	n19	213	1020	0	0
OBS_PARA: hirs4	metop-b	0	0	85	555
OBS_PARA: amsua	n15	1458	2026	830	234
OBS_PARA: amsua	n18	2223	2318	108	0
OBS_PARA: amsua	n19	176	960	0	0
OBS_PARA: amsua	metop-a	0	0	1077	1559
OBS_PARA: amsua	metop-b	0	0	265	1829
OBS_PARA: mhs	n18	2550	2695	160	0
OBS_PARA: mhs	n19	246	1103	0	0
OBS_PARA: mhs	metop-a	0	0	1237	1809
OBS_PARA: mhs	metop-b	0	0	321	2128

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:

4. GSI Diagnostics and Tuning

```

SETUPALL:,obstype,isis,nreal,nchanl= ps      ps      20      0
SETUPALL:,obstype,isis,nreal,nchanl= t      t      25      0
SETUPALL:,obstype,isis,nreal,nchanl= q      q      26      0
SETUPALL:,obstype,isis,nreal,nchanl= pw     pw     20      0
SETUPALL:,obstype,isis,nreal,nchanl= uv     uv     25      0
SETUPALL:,obstype,isis,nreal,nchanl= gps_ref gps    16      0
SETUPALL:,obstype,isis,nreal,nchanl= hirs4  hirs4_n19 33      19
      0 setuprad: passive obs      21 hirs4_n19
crtm_interface*init_crtm: crt_m_init() on path "./"
SpcCoeff_ReadFile(Binary)(INFORMATION) : FILE: ./hirs4_n19.SpcCoeff.bin; ^M
SpcCoeff RELEASE.VERSION: 8.03^M
  N_CHANNELS=19
Read_ODPS_Binary(INFORMATION) : FILE: ./hirs4_n19.TauCoeff.bin; ^M
  ODPS RELEASE.VERSION: 2.01  N_LAYERS=100  N_COMPONENTS=5  N_ABSORBERS=3  N_CHANNELS=19  N_COEFFS=82000
SEcategory_ReadFile(INFORMATION) : FILE: ./NPOESS.IRland.EmisCoeff.bin; ^M
SEcategory RELEASE.VERSION: 3.01^M
  CLASSIFICATION: NPOESS,  N_FREQUENCIES=20  N_SURFACE_TYPES=20
SETUPALL:,obstype,isis,nreal,nchanl= hirs4  hirs4_metop-a      33      19
crtm_interface*init_crtm: crt_m_init() on path "./"
...
...

```

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

```

...
...
FitCoeff_ReadFile(INFORMATION) : FILE: ./FASTEM6.MWwater.EmisCoeff.bin; ^M
FitCoeff RELEASE.VERSION : 1.6; DIMENSIONS= 3, 6, 2
MWwaterCoeff_ReadFile(INFORMATION) : FILE: ./FASTEM6.MWwater.EmisCoeff.bin; ^M
MWwaterCoeff RELEASE.VERSION: 1.6
SETUPRAD: write header record for mhs_n19      12      30
      8      0      0      22      4      30303
to file pe0000.mhs_n19_01 2014061700

```

The inner iteration of the first outer loop is discussed in the example below. In this example, the maximum number of iterations is 50.

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

```

GLBSOI: START pcgsoi jiter= 1
pcgsoi: gnorm(1:2),b= 2.767403469782257162E+03 2.767403469782257162E+03 0.0000000000000000E+00
Begin J table inner/outer loop 0 1
  J term J
surface pressure 5.7012207042385944E+03
temperature 6.4242087278840627E+03
wind 1.6782607330525603E+04
moisture 3.5878183830232451E+03
gps 7.8814883785376896E+03
radiance 3.4334884315701471E+04
-----
J Global 7.4712227839910673E+04
End Jo table inner/outer loop 0 1
Initial cost function = 7.471222783991067263E+04
Initial gradient norm = 5.260611627731377382E+01
cost,grad,step,b,step? = 1 0 7.471222783991067263E+04 5.260611627731377382E+01 1.717817994849075269E+00 0.0000000000000000E+00 good
pcgsoi: gnorm(1:2),b= 1.754232612149755596E+03 1.754232612149755141E+03 6.338911659627933792E-01
cost,grad,step,b,step? = 1 1 6.995833236051093263E+04 4.188356016565158058E+01 4.106937422100393142E+00 6.338911659627933792E-01 good
pcgsoi: gnorm(1:2),b= 1.216588309725912268E+03 1.216588309725912268E+03 6.935159575188962755E-01
cost,grad,step,b,step? = 1 2 6.275380879860417917E+04 3.487962599750622417E+01 2.174716042085542700E+00 6.935159575188962755E-01 good
pcgsoi: gnorm(1:2),b= 1.156766558917323891E+03 1.156766558917324346E+03 9.508282708864222998E-01
cost,grad,step,b,step? = 1 3 6.010807468482949480E+04 3.401127105706759579E+01 2.916832102067935306E+00 9.508282708864222998E-01 good
pcgsoi: gnorm(1:2),b= 6.945724726018979709E+02 6.945724726018985393E+02 6.004430775142600707E-01
...
...
cost,grad,step,b,step? = 1 49 4.142785387197384262E+04 1.680503228207865574E+00 2.338314294416948602E+00 1.076393393015242506E+00 good
pcgsoi: gnorm(1:2),b= 1.980029522220628557E+00 1.980029522219752591E+00 7.011209809087933786E-01
cost,grad,step,b,step? = 1 50 4.142125025938593171E+04 1.407135218172236746E+00 5.458012252072157899E+00 7.011209809087933786E-01 good
update_guess: successfully complete

```

4. GSI Diagnostics and Tuning

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

```

=====
Status  Var          Mean          Min          Max
analysis U      3.027810174754E+00 -4.616646796505E+01  6.874148210358E+01
analysis V     -2.783966384966E-02 -6.673607446514E+01  6.206906140999E+01
analysis TV     2.466648731614E+02  1.909849532362E+02  3.159577451606E+02
analysis Q      2.789588139811E-03  1.000000000000E-07  2.260955460480E-02
analysis TSEN   2.461750146062E+02  1.909846857229E+02  3.153599236074E+02
analysis OZ     1.000000000000E-15  1.000000000000E-15  1.000000000000E-15
analysis CW     0.000000000000E+00  0.000000000000E+00  0.000000000000E+00
analysis DIV    0.000000000000E+00  0.000000000000E+00  0.000000000000E+00
analysis VDR    0.000000000000E+00  0.000000000000E+00  0.000000000000E+00
analysis PRSL   4.154470108570E+01  2.152367800892E+00  1.028272918117E+02
analysis PS     9.910751025141E+01  6.684714489139E+01  1.029767184368E+02
analysis SST    2.942451749464E+02  2.688000183105E+02  3.240092468262E+02
analysis radb   6.939468963354E-02  -1.373884240000E+02  1.230549030000E+02
analysis pcpb   0.000000000000E+00  0.000000000000E+00  0.000000000000E+00
analysis aftb   0.000000000000E+00  0.000000000000E+00  0.000000000000E+00
=====
increment u          6.699567621553E-04  -1.360159370531E+01  7.008598936474E+00
increment v          1.598741317770E-03  -9.984198525101E+00  8.688133965521E+00
increment tv         -5.436012894801E-04  -2.969908758852E+00  4.753382796517E+00
increment tsen       4.740094380224E-04  -2.966625119631E+00  4.955252532399E+00
increment q          -5.666454694731E-06  -4.783507458114E-03  4.607810408495E-03
increment oz         0.000000000000E+00  0.000000000000E+00  0.000000000000E+00
increment cw         0.000000000000E+00  0.000000000000E+00  0.000000000000E+00
increment prse       4.684195247526E-03  -8.941125614721E-02  1.065972973935E-01
increment ps         1.145022012350E-02  -8.941125614721E-02  1.065972973935E-01
increment sst       2.337322205329E-02  -5.017646098146E-01  1.017498439243E+00
=====

```

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      20      0
SETUPALL:,obstype,isis,nreal,nchanl= t       t       25      0
SETUPALL:,obstype,isis,nreal,nchanl= q       q       26      0
...

```

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

```

SETUPALL:,obstype,isis,nreal,nchanl= ps      ps      20      0
SETUPALL:,obstype,isis,nreal,nchanl= t       t       25      0
SETUPALL:,obstype,isis,nreal,nchanl= q       q       26      0
SETUPALL:,obstype,isis,nreal,nchanl= pw      pw      20      0
SETUPALL:,obstype,isis,nreal,nchanl= uv      uv      25      0
SETUPALL:,obstype,isis,nreal,nchanl= gps_ref  gps     16      0
SETUPALL:,obstype,isis,nreal,nchanl= hirs4    hirs4_n19  33      19
      0 setuprad: passive obs      21 hirs4_n19
SETUPALL:,obstype,isis,nreal,nchanl= hirs4    hirs4_metop-a  33      19
SETUPALL:,obstype,isis,nreal,nchanl= amsua    amsua_n15  33      15
SETUPALL:,obstype,isis,nreal,nchanl= amsua    amsua_n18  33      15
SETUPALL:,obstype,isis,nreal,nchanl= hirs4    hirs4_metop-b  33      19
SETUPALL:,obstype,isis,nreal,nchanl= amsua    amsua_metop-a  33      15
SETUPALL:,obstype,isis,nreal,nchanl= amsua    amsua_n19  33      15
SETUPALL:,obstype,isis,nreal,nchanl= amsua    amsua_metop-b  33      15
SETUPALL:,obstype,isis,nreal,nchanl= mhs      mhs_n18    33      5
SETUPALL:,obstype,isis,nreal,nchanl= mhs      mhs_metop-a  33      5
SETUPALL:,obstype,isis,nreal,nchanl= mhs      mhs_metop-b  33      5
SETUPALL:,obstype,isis,nreal,nchanl= mhs      mhs_n19    33      5

```


4. GSI Diagnostics and Tuning

The output from the inner iterations in the second outer loop is shown below. In this example, the maximum number of iterations is 50.

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

```

GLBIOI: START pcgsoi jiter=          2
pcgsoi: gnorm(1:2),b= 9.125529304049867960E+02 9.125529304049867960E+02 0.0000000000000000E+00
Begin J table inner/outer loop      0          2
      J term                          J
background                          5.1520678195203345E+03
surface pressure                     4.1180289830866377E+03
temperature                          4.3079774551559522E+03
wind                                 1.0714401927194920E+04
moisture                             1.3696062777723114E+03
gps                                  3.1175680587783127E+03
radiance                             2.4255186183427002E+04
-----
J Global                             5.3034836704935471E+04
End Jo table inner/outer loop      0          2
Initial cost function = 5.303483670493547106E+04
Initial gradient norm = 3.020849103157896565E+01
cost,grad,step,b,step? = 2 0 5.303483670493547106E+04 3.020849103157896565E+01 1.417696886759607366E+00 0.0000000000000000E+00 good
pcgsoi: gnorm(1:2),b= 3.752399307247351885E+02 3.752399307247339380E+02 4.111979899710630493E-01
cost,grad,step,b,step? = 2 1 5.174111325649695937E+04 1.937111072511680021E+01 5.251908998246061167E+00 4.111979899710630493E-01 good
pcgsoi: gnorm(1:2),b= 2.403892651142143109E+02 2.403892651142157320E+02 6.406281566301591512E-01
cost,grad,step,b,step? = 2 2 4.977038728782249382E+04 1.550449177219989672E+01 3.447718424869901543E+00 6.406281566301591512E-01 good
pcgsoi: gnorm(1:2),b= 3.513995903418547186E+02 3.513995903418544913E+02 1.461794020522907633E+00
cost,grad,step,b,step? = 2 3 4.894159278934728354E+04 1.874565523906419173E+01 1.888694950411589302E+00 1.461794020522907633E+00 good
...
pcgsoi: gnorm(1:2),b= 4.240269047847287087E-01 4.240269047846388362E-01 8.928029496575288215E-01
cost,grad,step,b,step? = 2 49 4.549093632176067331E+04 6.511734828636134287E-01 3.659140330083644699E+00 8.928029496575288215E-01 good
pcgsoi: gnorm(1:2),b= 3.162643690270069974E-01 3.162643690267226138E-01 7.458592024656659492E-01
cost,grad,step,b,step? = 2 50 4.548938474781232799E+04 5.623738694383008108E-01 3.375651104905403432E+00 7.458592024656659492E-01 good
update_guess: successfully complete

```

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

```

=====
Status  Var          Mean              Min              Max
analysis U      3.031191508676E+00 -4.617266089077E+01 6.889816661368E+01
analysis V     -2.943556460524E-02 -6.653467898266E+01 6.166408410383E+01
analysis TV     2.467118072154E+02 1.909576489123E+02 3.159594232825E+02
analysis Q     2.792097480151E-03 1.000000000000E-07 2.263794691793E-02
analysis TSEN  2.462214978004E+02 1.909573814527E+02 3.153808896427E+02
analysis OZ    1.000000000007E-15 1.000000000000E-15 1.000000000000E-15
analysis CW    0.000000000000E+00 0.000000000000E+00 0.000000000000E+00
analysis DIV   0.000000000000E+00 0.000000000000E+00 0.000000000000E+00
analysis VOR  0.000000000000E+00 0.000000000000E+00 0.000000000000E+00
analysis PRSL  4.154936446270E+01 2.152390788296E+00 1.028757430184E+02
analysis PS    9.910817481416E+01 6.684789651716E+01 1.029802303066E+02
analysis SST   2.942451749464E+02 2.688000183105E+02 3.240092468262E+02
analysis radb  6.938254104671E-02 -1.373884240000E+02 1.230549030000E+02
analysis pcpb  0.000000000000E+00 0.000000000000E+00 0.000000000000E+00
analysis aftb  0.000000000000E+00 0.000000000000E+00 0.000000000000E+00
=====
increment u      3.381333921427E-03 -1.617060550844E+00 3.067216672606E+00
increment v     -1.595900755559E-03 -2.932441477540E+00 1.626847973396E+00
increment tv     4.693405394733E-02 -9.818098783002E-01 2.215784626624E+00
increment tsen  4.648264962860E-02 -9.818069362856E-01 2.215777805989E+00
increment q     2.508348182621E-06 -2.410389935149E-03 1.663695364258E-03
increment oz    0.000000000000E+00 0.000000000000E+00 0.000000000000E+00
increment cw    0.000000000000E+00 0.000000000000E+00 0.000000000000E+00
increment prse  2.718674106775E-04 -5.539352084836E-02 4.239997318097E-02
increment ps    6.645627545749E-04 -5.539352084836E-02 4.239997318097E-02
increment sst  -2.077555444927E-03 -3.656869699698E-01 7.433384932582E-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 = T). The maximum and minimum values are useful information for a quick check of the reasonableness of the analysis:

4. GSI Diagnostics and Tuning

```

at 2 in wrwrfmassa
update sigf03
at 3 in wrwrfmassa
at 6 in wrwrfmassa
at 10.11 in wrwrfmassa,max,min(temp1)= 2.1931874E-02 1.3461057E-03
at 10.12 in wrwrfmassa,max,min(tempa)= 0.0000000E+00 0.0000000E+00
at 10.13 in wrwrfmassa,max,min(tempa)= 0.0000000E+00 -2.1931874E-02
at 10.14 in wrwrfmassa,max,min(temp1)= 0.0000000E+00 0.0000000E+00
iy,m,d,h,m,s=      2014      6      17      0      0
0
nlon,lat,sig_regional=      332      215      50
rmse_var=P_TOP
ordering=0
WrfType,WRF_REAL=      104      104
ndim1=      0
staggering= N/A
start_index=      1      1      1      0
end_index1=      332      215      50      0
p_top= 2000.000
rmse_var=MUB
ordering=XY
WrfType,WRF_REAL=      104      104
ndim1=      2
staggering= N/A
start_index=      1      1      1      0
end_index1=      332      215      50      0
max,min MUB= 98672.59 63425.52
max,min psfc= 102799.7 66795.70
max,min MU= 2799.734 -1187.844
rmse_var=MU
ordering=XY
WrfType,WRF_REAL=      104      104
ndim1=      2
staggering= N/A
start_index=      1      1      1      0
end_index1=      332      215      50      0
k,max,min,mid T=      1 321.6379 270.7839 309.3401
k,max,min,mid T=      2 321.7433 270.9335 309.3846
k,max,min,mid T=      3 321.4780 271.1794 309.3658
...

k,max,min,mid T=      49 662.7022 604.6494 637.0358
k,max,min,mid T=      50 693.4219 647.1161 675.5701

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 setuprhall 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      20      0
SETUPALL:,obstype,isis,nreal,nchanl= t      t      25      0
SETUPALL:,obstype,isis,nreal,nchanl= q      q      26      0
SETUPALL:,obstype,isis,nreal,nchanl= pw     pw     20      0

```

4. GSI Diagnostics and Tuning

```
SETUPALL:,obstype,isis,nreal,nchanl= uv uv 25 0
SETUPALL:,obstype,isis,nreal,nchanl= gps_ref gps 16 0
SETUPALL:,obstype,isis,nreal,nchanl= hirs4 hirs4_n19 33 19
0 setuprad: passive obs 21 hirs4_n19
SETUPRAD: write header record for hirs4_n19 12 30
8 0 0 22 4 30303
to file pe0000.hirs4_n19_03 2014061700
SETUPALL:,obstype,isis,nreal,nchanl= hirs4 hirs4_metop-a 33 19
SETUPRAD: write header record for hirs4_metop-a 12 30
8 0 0 22 4 30303
```

Deallocate the data arrays and finalize the GSI run:

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

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

```
ENDING DATE-TIME JUL 02,2016 20:43:40.422 184 SAT 2457572
PROGRAM GSI_ANL HAS ENDED.
* . . . . .
*****RESOURCE STATISTICS*****
The total amount of wall time = 438.663534
The total amount of time in user mode = 427.578998
The total amount of time in sys mode = 9.457562
The maximum resident set size (KB) = 2020132
Number of page faults without I/O activity = 312762
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 = 7641
Number of InVoluntary Context Switches = 851
*****END OF RESOURCE STATISTICS*****
```

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

4.2 Single Observation Test

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

4.2.1 Setup a Single Observation Test

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

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

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

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

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

Note:

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

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

4.2.2 Examples of Single Observation Tests for GSI

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

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 Figure 4.1 was generated using NCL scripts, which can be found in the `util/Analysis_Uilities/plots_ncl` directory, introduced in Section A.4 .

4. GSI Diagnostics and Tuning

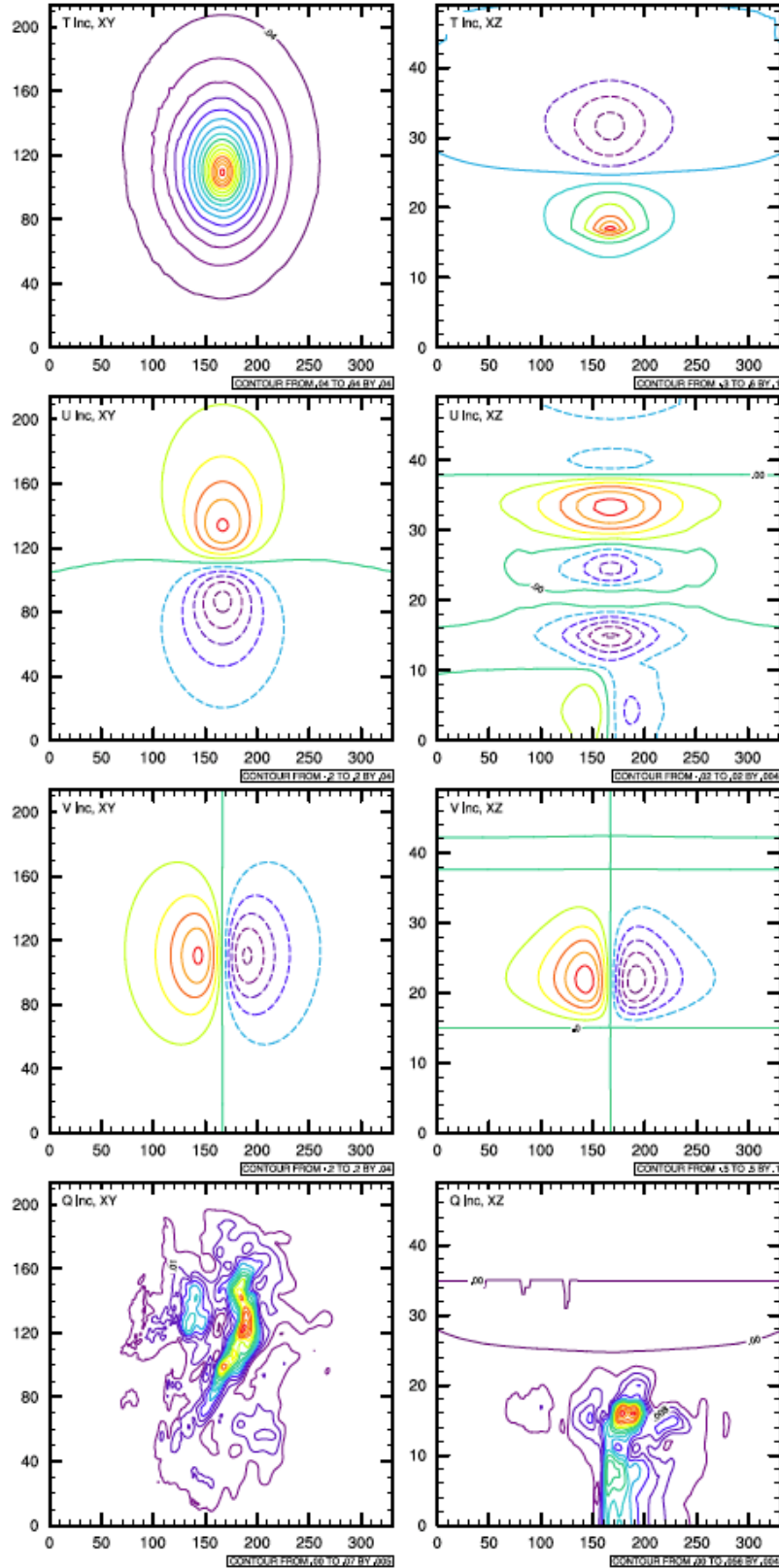


Figure 4.1: Horizontal cross sections (left column) and vertical cross sections (right column) of analysis increment of T, U, V, and Q from a single T observation

4.3 Control Data Usage

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

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

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

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

All BUFR/PrepBUFR observation files need to be linked to the working directory with GSI recognized names before can be used by GSI analysis. The run script (*run_gsi_regional.ksh*) makes these links after locating the working directory. Turning on or off these links can control the use of all the data contained in the BUFR files. Table 4.1 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:

Table 4.1: list of all default observation file names recognized by GSI

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.	<i>gdas1.t12z.prepbufr</i>
satwndbufr	satellite winds	<i>gdas1.t12z.satwnd.tm00.bufr_d</i>
amsuabufr	AMSU-A 1b radiance (brightness temperatures) from satellites NOAA-15, 16, 17,18, 19 and METOP-A/B	<i>gdas1.t12z.1bamua.tm00.bufr_d</i>

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

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

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

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

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

4. GSI Diagnostics and Tuning

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

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

This setup tells GSI that conventional observation variables *ps*, *t*, and *q* should be read in from the file *prepbuf_r* and AMSU-A radiances from NOAA-15 and -18 satellites should be read in from the file *amsuabuf_r*. 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:

```
amsuabuf_r     amsua           n15          amsua_n15      10.0         2      0
```

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

The observation filename in *dfile* can be different from the sample script (*comgsi_namelist.ksh*). If the filename in *dfile* has been changed, the link from the BUFR files to the GSI recognized name in the run script also needs to be changed correspondingly. For example, if we change the *dfile* for *amsuabuf_r* file for NOAA-15 to be *amsuabuf_r_n15*

```
amsuabuf_r_n15  amsua           n15          amsua_n15      10.0         2      0
amsuabuf_r      amsua           n18          amsua_n18      10.0         2      0
```

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

```
# Link to the radiance data
ln -s ${OBS_ROOT}/le_gdas1.t00z.1bamua.tm00.bufr_d amsuabuf_r
ln -s ${OBS_ROOT}/le_gdas1.t00z.1bamua.tm00.bufr_d amsuabuf_r_n15
```

The GSI will read NOAA-18 AMSU-A observations from file *amsuabuf_r* and NOAA-15 AMSU-A observations from file *amsuabuf_r_n15* based on the above changes to the run scripts and namelist. In this example, both *amsuabuf_r* and *amsuabuf_r_n15* are linked to the same BUFR file and NOAA-15 AMSU-A and NOAA-18 AMSU-A observations are still read in from the same BUFR file. If *amsuabuf_r* and *amsuabuf_r_n15* link to different BUFR files, then NOAA-15 AMSU-A and NOAA-18 AMSU-A will be read in from different BUFR files. Clearly, the changeable filename in *dfile* gives GSI more capability to handle multiple data resources.

4. GSI Diagnostics and Tuning

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 4.2 is a list of info files and their function:

Table 4.2: 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 <i>rw</i> and 2.5 <i>srw</i>), gps, <i>pm2_5</i>
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	nmitter	gross	ermax	ermin	var_b	var_pg	ithin	rmesh	pmesh	npred	pmot	ptime
tcp	112	0	1	3.0	0	0	0	75.0	5.0	1.0	75.0	0.000000	0	0.	0.	0	0.	0.
ps	120	0	1	3.0	0	0	0	4.0	3.0	1.0	4.0	0.000300	0	0.	0.	0	0.	0.
ps	132	0	-1	3.0	0	0	0	4.0	3.0	1.0	4.0	0.000300	0	0.	0.	0	0.	0.
ps	180	0	1	3.0	0	0	0	4.0	3.0	1.0	4.0	0.000300	0	0.	0.	0	0.	0.
ps	180	01	1	3.0	0	0	0	4.0	3.0	1.0	4.0	0.000300	0	0.	0.	0	0.	0.
ps	181	0	1	3.0	0	0	0	3.6	3.0	1.0	3.6	0.000300	0	0.	0.	0	0.	0.
ps	182	0	1	3.0	0	0	0	4.0	3.0	1.0	4.0	0.000300	0	0.	0.	0	0.	0.
ps	183	0	-1	3.0	0	0	0	4.0	3.0	1.0	4.0	0.000300	0	0.	0.	0	0.	0.
ps	187	0	1	3.0	0	0	0	4.0	3.0	1.0	4.0	0.000300	0	0.	0.	0	0.	0.
t	120	0	1	3.0	0	0	0	8.0	5.6	1.3	8.0	0.000001	0	0.	0.	0	0.	0.
t	126	0	-1	3.0	0	0	0	8.0	5.6	1.3	8.0	0.001000	0	0.	0.	0	0.	0.
t	130	0	1	3.0	0	0	0	7.0	5.6	1.3	7.0	0.001000	0	0.	0.	0	0.	0.
t	131	0	1	3.0	0	0	0	7.0	5.6	1.3	7.0	0.001000	0	0.	0.	0	0.	0.
t	132	0	1	3.0	0	0	0	7.0	5.6	1.3	7.0	0.001000	0	0.	0.	0	0.	0.
t	133	0	1	3.0	0	0	0	7.0	5.6	1.3	7.0	0.004000	0	0.	0.	0	0.	0.
t	134	0	-1	3.0	0	0	0	7.0	5.6	1.3	7.0	0.004000	0	0.	0.	0	0.	0.
t	135	0	-1	3.0	0	0	0	7.0	5.6	1.3	7.0	0.004000	0	0.	0.	0	0.	0.
t	180	0	1	3.0	0	0	0	7.0	5.6	1.3	7.0	0.004000	0	0.	0.	0	0.	0.
t	180	01	1	3.0	0	0	0	7.0	5.6	1.3	7.0	0.004000	0	0.	0.	0	0.	0.
t	181	0	-1	3.0	0	0	0	7.0	5.6	1.3	7.0	0.004000	0	0.	0.	0	0.	0.
t	182	0	1	3.0	0	0	0	7.0	5.6	1.3	7.0	0.004000	0	0.	0.	0	0.	0.

The meaning of each column is explained in the header of the file and is listed in Table 4.3.

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.

4. GSI Diagnostics and Tuning

Table 4.3: list of the content for each convinfo column

Column Name	Content of the column
otype	observation variables (t, uv, q, etc.)
type	prepbufr observation type (if available)
sub	prepbufr subtype (not yet available)
iuse	flag if to use/not use / monitor data; =1, use data, the data type will be read and used in the analysis after quality control; =0, read in and process data, use for quality control, but do NOT assimilate; =-1, monitor data. This data type will be read in and monitored but not be used in the GSI analysis.
twindow	time window (+/- hours) for data used in the analysis
numgrp	cross validation parameter - number of groups
ngroup	cross validation parameter - group to remove from data use
nmiter	cross validation parameter - external iteration to introduce removed data
gross	gross error parameter - gross error
ermax	gross error parameter - maximum error
ermin	gross error parameter - minimum error
var_b	variational quality control parameter - b parameter
var_pg	variational quality control parameter - pg parameter
ithin	Flag to turn on thinning (0, no thinning, 1 - thinning)
rmesh	size of horizontal thinning mesh (in kilometers)
pmesh	size of vertical thinning mesh
npred	Number of bias correction predictors
pmot	the option to keep thinned data as monitored, 0: not to keep, other values: to keep
ptime	time interval for thinning, 0, no temporal thinning, other values define time interval (less than 6)

4. GSI Diagnostics and Tuning

- 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 4.4.

```

!sensor/instr/sat      chan iuse  error  error_cld  ermax  var_b  var_pg  cld_det
amsua_n15              1   1    3.000  20.000    4.500  10.000  0.000   -2
amsua_n15              2   1    2.200  18.000    4.500  10.000  0.000   -2
amsua_n15              3   1    2.000  12.000    4.500  10.000  0.000   -2
amsua_n15              4   1    0.600   3.000    2.500  10.000  0.000   -2
amsua_n15              5   1    0.300   0.500    2.000  10.000  0.000   -2
amsua_n15              6  -1    0.230   0.300    2.000  10.000  0.000   -2
amsua_n15              7   1    0.250   0.250    2.000  10.000  0.000   -2
amsua_n15              8   1    0.275   0.275    2.000  10.000  0.000   -2
amsua_n15              9   1    0.340   0.340    2.000  10.000  0.000   -2
amsua_n15             10   1    0.400   0.400    2.000  10.000  0.000   -2
amsua_n15             11  -1    0.600   0.600    2.500  10.000  0.000   -2
amsua_n15             12   1    1.000   1.000    3.500  10.000  0.000   -2
amsua_n15             13   1    1.500   1.500    4.500  10.000  0.000   -2
amsua_n15             14  -1    2.000   2.000    4.500  10.000  0.000   -2
amsua_n15             15   1    3.500  15.000    4.500  10.000  0.000   -2
hirs3_n17              1  -1    2.000   0.000    4.500  10.000  0.000   -1
hirs3_n17              2  -1    0.600   0.000    2.500  10.000  0.000    1
hirs3_n17              3  -1    0.530   0.000    2.500  10.000  0.000    1

```

Table 4.4: list of the content for each satinfo column

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

4.4 Domain Partition for Parallelization and Observation Distribution

The standard output file (*stdout*) has an information block that shows the distribution of different kinds of observations in each sub-domain. This block follows the observation input section. The following is the observation distribution of the case shown in Section 4.1. From the case introduction, we know the prepbufr (conventional data), radiance BUFR files, and GPS BUFR files were used. In this list, the conventional observations (ps, t, q, pw, and uv), GPSRO (gps_ref), and radiance data (amusa, hirs4, and mhs from Metop-a, Metop-b, NOAA 15 and 18) were distributed among 4 sub-domains:

4. GSI Diagnostics and Tuning

OBS_PARA: ps		1429	3190	4655	6774
OBS_PARA: t		2564	5200	7057	11128
OBS_PARA: q		2346	4626	6148	8128
OBS_PARA: pw		65	80	63	49
OBS_PARA: uv		3358	6453	8091	11998
OBS_PARA: gps_ref		1799	1368	2664	3520
OBS_PARA: hirs4	metop-a	0	0	1146	1661
OBS_PARA: hirs4	n19	213	1020	0	0
OBS_PARA: hirs4	metop-b	0	0	85	555
OBS_PARA: amsua	n15	1458	2026	830	234
OBS_PARA: amsua	n18	2223	2318	108	0
OBS_PARA: amsua	n19	176	960	0	0
OBS_PARA: amsua	metop-a	0	0	1077	1559
OBS_PARA: amsua	metop-b	0	0	265	1829
OBS_PARA: mhs	n18	2550	2695	160	0
OBS_PARA: mhs	n19	246	1103	0	0
OBS_PARA: mhs	metop-a	0	0	1237	1809
OBS_PARA: mhs	metop-b	0	0	321	2128

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

4.5 Observation Innovation Statistics

The GSI analysis gives a group of files named *fort.2** to summarize observations fitting to the current solution in each outer loop (except for *fort.220*, see explanation on *fort.220* in next section). The content of some of these files is listed in Table 4.5:

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

4.5.1 Conventional observations

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

```
current fit of surface pressure data, ranges in mb
-----
pressure levels (hPa)= 0.0 2000.0
  it  obs  type stype  count  bias  rms  cpen  qcpen
o-g 01  ps   120 0000   101  0.1813  0.6089  0.4711  0.4711
o-g 01  ps   180 0000   601  0.0378  0.6354  0.7171  0.7171
o-g 01  ps   180 0001  1428  0.1857  0.8555  0.7164  0.7164
o-g 01  ps   181 0000   610  0.1959  0.8991  0.7387  0.7387
o-g 01  ps   182 0000    1  0.9173  0.9173  2.8976  2.8976
o-g 01  ps   187 0000  11149  0.1999  0.7877  0.3360  0.3360
o-g 01  all  13890  0.1912  0.7931  0.4105  0.4105
o-g 01  ps rej 120 0000    1 -3.5799  3.5799  0.0000  0.0000
o-g 01  ps rej 180 0001    7 -0.3349  4.9273  0.0000  0.0000
o-g 01  ps rej 181 0000   47  1.1146  54.8539  0.0000  0.0000
o-g 01  ps rej 183 0000    3 10.4797  10.4836  0.0000  0.0000
o-g 01  ps rej 187 0000   52  4.2528  7.4112  0.0000  0.0000
o-g 01  rej all  110  2.7186  36.2804  0.0000  0.0000
o-g 01  ps mon 132 0000    1  0.9173  0.9173  0.2104  0.2104
o-g 01  ps mon 180 0000  113  0.0447  0.5158  0.8709  0.8709
o-g 01  ps mon 180 0001   24  0.2122  0.4050  0.3559  0.3559
o-g 01  ps mon 181 0000  207  0.0910  0.9492  1.2514  1.2514
```

4. GSI Diagnostics and Tuning

Table 4.5: List of the content and units for each fort files

File Name	Variables in file	Ranges/units
<i>fort.201</i> <i>or</i> <i>fit_p1.analysis_time</i>	fit of surface pressure data	mb
<i>fort.202</i> <i>or</i> <i>fit_w1.analysis_time</i>	fit of u, v wind data	m/s
<i>fort.203</i> <i>or</i> <i>fit_t1.analysis_time</i>	fit of temperature data	K
<i>fort.204</i> <i>or</i> <i>fit_q1.analysis_time</i>	fit of moisture data	percent of guess qsaturation
<i>fort.205</i>	fit of precipitation water data	mm
<i>fort.206</i>	fit of ozone observations from sbuv6_n14 (, _n16, _n17, _n18), sbuv8_n16 (, _n17, _n18, _n19), omi_aura, gome_metop-a/b, mls_aura	
<i>fort.207</i> <i>or</i> <i>fit_rad1.analysis_time</i>	fit of satellite radiance data, such as: am- sua_n15(, n16, n17, n18, metop-a, aqua, n19), amsub_n17, hirs3_n17, hirs4_n19 (, metop-a), etc	
<i>fort.208</i>	fit of precpitation rate (pcp_ssmi, pcp_tmi)	
<i>fort.209</i>	fit of radar radial wind (rw)	
<i>fort.210</i>	fit of lidar wind (dw)	
<i>fort.211</i>	fit of radar superob wind data (srw)	
<i>fort.212</i>	fit of GPS data (refractivity or bending angle)	fractional differ- ence
<i>fort.213</i>	fit of conventional sst data	C
<i>fort.214</i>	Tropical cyclone central pressure	
<i>fort.215</i>	Lagrangian tracer data	
<i>Fort.217</i>	Fit of aerosol product (aod)	
Fort.218	Fit of wind gust	
Fort.219	Fit of visibility	

4. GSI Diagnostics and Tuning

o-g 01	ps mon 183 0000	1386	0.3974	1.0861	0.0000	0.0000
o-g 01	ps mon 187 0000	88	-0.0100	0.5290	0.7534	0.7534
o-g 01	mon all	1819	0.3188	1.0169	0.2378	0.2378

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

it	obs	type	styp	ptop pbot	1000.0 1200.0	900.0 1000.0	800.0 900.0	600.0 800.0	100.0 150.0	50.0 100.0	0.0 2000.0
o-g 01	uv	220	0000	count	44	223	223	478	437	519	4231
o-g 01	uv	220	0000	bias	0.26	0.51	0.04	0.35	0.99	0.97	0.59
o-g 01	uv	220	0000	rms	2.21	2.48	2.51	2.84	5.88	4.79	4.18
o-g 01	uv	220	0000	cpen	0.31	0.38	0.44	0.61	1.64	1.18	0.90
o-g 01	uv	220	0000	qcpen	0.31	0.38	0.44	0.60	1.63	1.18	0.89
o-g 01	uv	223	0000	count	0	6	16	88	32	8	331
o-g 01	uv	223	0000	bias	0.00	0.23	-0.29	0.85	-0.76	-4.89	0.27
o-g 01	uv	223	0000	rms	0.00	1.28	1.32	2.86	5.21	6.63	3.76
o-g 01	uv	223	0000	cpen	0.00	0.06	0.07	0.45	0.88	1.31	0.73
...											
o-g 01		all		count	1799	1726	2001	3050	469	527	13124
o-g 01		all		bias	0.05	0.91	0.78	0.63	0.87	0.88	0.62
o-g 01		all		rms	2.46	2.58	2.65	3.21	5.83	4.82	3.54
o-g 01		all		cpen	0.24	0.24	0.30	0.45	1.58	1.19	0.52
o-g 01		all		qcpen	0.23	0.24	0.30	0.44	1.58	1.19	0.52
o-g 01	uv rej	220	0000	count	0	0	0	0	0	0	800
o-g 01	uv rej	220	0000	bias	0.00	0.00	0.00	0.00	0.00	0.00	3.11
o-g 01	uv rej	220	0000	rms	0.00	0.00	0.00	0.00	0.00	0.00	6.41
...											
o-g 01	rej	all		count	23	108	21	41	2	0	1008
o-g 01	rej	all		bias	44.72	6.40	-0.30	-7.38	11.46	0.00	3.84
o-g 01	rej	all		rms	56.08	16.60	6.87	10.47	43.15	0.00	12.28
o-g 01	rej	all		cpen	0.00	0.00	0.00	0.00	0.00	0.00	0.00
o-g 01	rej	all		qcpen	0.00	0.00	0.00	0.00	0.00	0.00	0.00
o-g 01	uv mon	220	0000	count	0	4	0	1	8	9	29
o-g 01	uv mon	220	0000	bias	0.00	1.07	0.00	6.04	-1.03	0.23	-2.21
o-g 01	uv mon	220	0000	rms	0.00	20.49	0.00	17.78	9.06	8.51	12.52
o-g 01	uv mon	220	0000	cpen	0.00	0.00	0.00	0.00	0.00	0.00	0.00
o-g 01	uv mon	220	0000	qcpen	0.00	0.00	0.00	0.00	0.00	0.00	0.00
...											
o-g 01	mon	all		count	3573	7736	1004	563	8	9	13052
o-g 01	mon	all		bias	-0.01	0.40	0.05	-0.30	-1.03	0.23	0.23
o-g 01	mon	all		rms	2.55	3.25	4.67	5.98	9.06	8.51	3.60
o-g 01	mon	all		cpen	0.81	1.21	1.67	1.28	0.00	0.00	1.12
o-g 01	mon	all		qcpen	0.80	1.15	1.53	1.16	0.00	0.00	1.07
...											

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

The table 4.6 lists the meaning of each item in file *fort.201-213* except file *fort.207*:

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.

4. GSI Diagnostics and Tuning

Table 4.6: list of each item in file fort.201-213 (except fort.207)

Name	Explanation
<i>it</i>	outer loop number = 01: observation - background = 02: observation - analysis (after 1st outer loop) = 03: observation - analysis (after 2nd outer loop)
<i>obs</i>	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
<i>type</i>	prepbufr observation type (see BUFR User's Guide for details)
<i>styp</i>	prepbufr observation subtype (not used now)
<i>ptop</i>	for multiple level data: pressure at the top of the layer
<i>pbot</i>	for multiple level data: pressure at the bottom of the layer
<i>count</i>	The number of observations summarized under observation types and vertical layers
<i>bias</i>	Bias of observation departure for each outer loop (<i>it</i>)
<i>rms</i>	Root Mean Square of observation departure for each outer loop (<i>it</i>)
<i>cpen</i>	Observation part of penalty (cost function)
<i>qcpen</i>	nonlinear qc penalty

4. GSI Diagnostics and Tuning

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

it	obs	type	styp	ptop pbot	1000.0 1200.0	900.0 1000.0	800.0 900.0	600.0 800.0	100.0 150.0	50.0 100.0	0.0 2000.0
o-g 01	uv	220	0000	count	44	223	223	478	437	519	4231
o-g 01	uv	220	0000	bias	0.26	0.51	0.04	0.35	0.99	0.97	0.59
o-g 01	uv	220	0000	rms	2.21	2.48	2.51	2.84	5.88	4.79	4.18
o-g 01	uv	220	0000	cpen	0.31	0.38	0.44	0.61	1.64	1.18	0.90
o-g 01	uv	220	0000	qcpen	0.31	0.38	0.44	0.60	1.63	1.18	0.89
...											
o-g 01	uv	rej	220	0000	count	0	0	0	0	0	800
o-g 01	uv	rej	220	0000	bias	0.00	0.00	0.00	0.00	0.00	3.11
o-g 01	uv	rej	220	0000	rms	0.00	0.00	0.00	0.00	0.00	6.41
o-g 01	uv	rej	220	0000	cpen	0.00	0.00	0.00	0.00	0.00	0.00
o-g 01	uv	rej	220	0000	qcpen	0.00	0.00	0.00	0.00	0.00	0.00
...											
o-g 01	uv	mon	220	0000	count	0	4	0	1	8	29
o-g 01	uv	mon	220	0000	bias	0.00	1.07	0.00	6.04	-1.03	0.23
o-g 01	uv	mon	220	0000	rms	0.00	20.49	0.00	17.78	9.06	8.51
o-g 01	uv	mon	220	0000	cpen	0.00	0.00	0.00	0.00	0.00	0.00
o-g 01	uv	mon	220	0000	qcpen	0.00	0.00	0.00	0.00	0.00	0.00
...											
it	obs	type	styp	ptop pbot	1000.0 1200.0	900.0 1000.0	800.0 900.0	600.0 800.0	100.0 150.0	50.0 100.0	0.0 2000.0
o-g 02	uv	220	0000	count	44	223	223	478	437	519	4231
o-g 02	uv	220	0000	bias	0.21	0.05	-0.43	0.03	0.69	0.96	0.39
o-g 02	uv	220	0000	rms	2.13	2.26	2.29	2.56	5.19	4.55	3.73
o-g 02	uv	220	0000	cpen	0.32	0.31	0.37	0.50	1.27	1.07	0.71
o-g 02	uv	220	0000	qcpen	0.32	0.31	0.37	0.50	1.27	1.07	0.71
...											
o-g 02	uv	rej	220	0000	count	0	0	0	0	0	800
o-g 02	uv	rej	220	0000	bias	0.00	0.00	0.00	0.00	0.00	2.96
o-g 02	uv	rej	220	0000	rms	0.00	0.00	0.00	0.00	0.00	6.33
o-g 02	uv	rej	220	0000	cpen	0.00	0.00	0.00	0.00	0.00	0.00
o-g 02	uv	rej	220	0000	qcpen	0.00	0.00	0.00	0.00	0.00	0.00
...											
o-g 02	uv	mon	220	0000	count	0	4	0	1	8	29
o-g 02	uv	mon	220	0000	bias	0.00	2.16	0.00	5.80	-1.00	0.23
o-g 02	uv	mon	220	0000	rms	0.00	18.44	0.00	12.60	10.31	8.58
o-g 02	uv	mon	220	0000	cpen	0.00	0.00	0.00	0.00	0.00	0.00
o-g 02	uv	mon	220	0000	qcpen	0.00	0.00	0.00	0.00	0.00	0.00
...											
it	obs	type	styp	ptop pbot	1000.0 1200.0	900.0 1000.0	800.0 900.0	600.0 800.0	100.0 150.0	50.0 100.0	0.0 2000.0
o-g 03	uv	220	0000	count	44	223	223	478	437	519	4231
o-g 03	uv	220	0000	bias	0.28	0.08	-0.32	0.11	0.74	1.00	0.43
o-g 03	uv	220	0000	rms	2.11	2.17	2.20	2.38	5.00	4.46	3.60

4. GSI Diagnostics and Tuning

```

o-g 03      uv      220 0000  cpen    0.31  0.29  0.34  0.43  1.18  1.03  0.65
o-g 03      uv      220 0000  qcpen    0.31  0.29  0.34  0.43  1.18  1.03  0.65
...
o-g 03      uv rej 220 0000  count      0      0      0      0      0      0      800
o-g 03      uv rej 220 0000  bias      0.00  0.00  0.00  0.00  0.00  0.00  2.98
o-g 03      uv rej 220 0000  rms      0.00  0.00  0.00  0.00  0.00  0.00  6.35
o-g 03      uv rej 220 0000  cpen      0.00  0.00  0.00  0.00  0.00  0.00  0.00
o-g 03      uv rej 220 0000  qcpen      0.00  0.00  0.00  0.00  0.00  0.00  0.00
...
o-g 03      uv mon 220 0000  count      0      4      0      1      8      9      29
o-g 03      uv mon 220 0000  bias      0.00  1.86  0.00  6.09 -0.98  0.07 -0.94
o-g 03      uv mon 220 0000  rms      0.00 18.76  0.00 12.59 10.34  8.69 11.01
o-g 03      uv mon 220 0000  cpen      0.00  0.00  0.00  0.00  0.00  0.00  0.00
o-g 03      uv mon 220 0000  qcpen      0.00  0.00  0.00  0.00  0.00  0.00  0.00

```

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

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

```

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

```

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

4.5.2 Satellite Radiance

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

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

```

RADINFO_READ:  jpch_rad=  2723
  1 amsua_n15      chan=   1 var=   3.000 varch_cld= 20.000 use=   1 ermax=   4.500 b_rad=   10.00 pg_rad=   0.00 icld_det=-2
  2 amsua_n15      chan=   2 var=   2.200 varch_cld= 18.000 use=   1 ermax=   4.500 b_rad=   10.00 pg_rad=   0.00 icld_det=-2
  3 amsua_n15      chan=   3 var=   2.000 varch_cld= 12.000 use=   1 ermax=   4.500 b_rad=   10.00 pg_rad=   0.00 icld_det=-2
  4 amsua_n15      chan=   4 var=   0.600 varch_cld=   3.000 use=   1 ermax=   2.500 b_rad=   10.00 pg_rad=   0.00 icld_det=-2
  5 amsua_n15      chan=   5 var=   0.300 varch_cld=   0.500 use=   1 ermax=   2.000 b_rad=   10.00 pg_rad=   0.00 icld_det=-2
...

```


4. GSI Diagnostics and Tuning

This shows there are 2723 channels listed in the *satinfo* file and the 2723 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 bias correction, after reading the *satbias_in* file:

```
RADINFO_READ: ***WARNING instrument/channel ahi_himawari8          16
not found in satbias_pc file - set to zero
RADINFO_READ: guess air mass bias correction coefficients below
1      amsua_n15  -1.965181  0.000000  39.068176  -0.923195  0.408460  0.000000  0.000000  0.001735  3.689709
          2.367472  8.862940  -1.848292
2      amsua_n15  -7.415403  0.000000  80.138019  -0.818472  -1.013216  0.000000  0.000000  0.014279  2.638804
          8.315096  22.041458  -1.564773
...
```

Each channel has 12 coefficients listed in a line. Therefore, there are 2723 lines of radiance 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	ivar1	nlgross
metop-a	hirs4	371.46818888	385	143	0	25	0	1452	0
	qcpenalty	qc1	qc2	qc3	qc4	qc5	qc6	qc7	
		371.46818888	0	0	1651	3330	0	0	0

sat	type	penalty	nobs	iland	isnoice	icoast	ireduce	ivar1	nlgross
metop-b	hirs4	0.00000000	34	25	0	0	0	139	0
	qcpenalty	qc1	qc2	qc3	qc4	qc5	qc6	qc7	
		0.00000000	0	0	97	279	0	0	0

6

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

...

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

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

- Summaries for various statistics as a function of channel

1	2	3	4	5	6	7	8	9	10	11
1	1	amsua_n15	915	247	3.000	0.6623748	1.2673027	0.2414967	2.4209147	2.0627099
2	2	amsua_n15	902	261	2.000	0.5209886	1.1773765	0.2626527	2.2705374	1.9414234
3	3	amsua_n15	1114	48	2.000	1.2349467	-1.5090660	0.3421312	1.9416738	1.2218089
4	4	amsua_n15	1162	0	0.600	-0.2114506	-0.3195250	0.5286580	0.6271317	0.5396275
5	5	amsua_n15	1162	3	0.300	-0.0888496	-0.1685199	0.4352839	0.2734131	0.2153040
6	6	amsua_n15	2093	3	-0.230	-1.8141034	-0.0695503	0.8415303	0.2646033	0.2552992
7	7	amsua_n15	2342	28	0.250	-0.0830293	0.0453835	0.6141132	0.2522587	0.2481427
8	8	amsua_n15	2311	59	0.275	0.0201868	0.1758359	0.7782391	0.3173862	0.2642267
9	9	amsua_n15	2124	246	0.340	0.1262987	0.5008846	1.4907547	0.5544190	0.2376868
10	10	amsua_n15	82	2288	0.400	0.6700157	0.8263504	2.0387089	0.8304076	0.0819868
15	15	amsua_n15	862	300	3.000	0.8244923	-2.2960147	0.3769625	2.7445692	1.5036544

...

4. GSI Diagnostics and Tuning

Table 4.7: content of summarizing radiance observation process in fort.207

Name	Explanation
<i>sat</i>	satellite name
<i>type</i>	instrument type
<i>penalty</i>	contribution to cost function from this observation type
<i>nobs</i>	number of good observations used in the assimilation
<i>iland</i>	number of observations over land
<i>isnoice</i>	number of observations over sea ice and snow
<i>icoast</i>	number of observations over coast
<i>ireduce</i>	number of observations that reduce qc bounds in tropics
<i>ivarl</i>	number of observations tossed by gross check
<i>nlgross</i>	number of observation tossed by nonlinear qc
<i>qcpenalty</i>	nonlinear qc penalty from this data type
<i>qc1-7</i>	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
<i>rad total penalty_all</i>	summary of penalty for all radiance observation types
<i>rad total qcpenalty_all</i>	summary of qcpenalty for all radiance observation types
<i>rad total failed nonlingc</i>	summary of observation tossed by nonlinear qc for all radiance observation types

4. GSI Diagnostics and Tuning

```

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

```

The Table 4.8 lists the meaning of each column in above statistics:

Table 4.8: content of fit statistic for each channel in fort.207

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

- Final summary for each observation type

```

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

```

The table 4.9 lists the meaning of each column in the above statistics:

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. GSI Diagnostics and Tuning

Table 4.9: content of final summary section in fort.207

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

4.6 Convergence Information

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

1. Standard output file (*stdout*):

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

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

```

GLBSOI:  START pcgsoi  jiter=          1
pcgsoi:  gnorm(1:2),b=  9.869857497554413276E+05  9.869857497554413276E+05  0.000000000000000000E+00
Initial cost function =  3.915930707165839704E+04
Initial gradient norm =  9.934715646436194447E+02
cost,grad,step,b,step? =  1  0  3.915930707165839704E+04  9.934715646436194447E+02  4.821051367939106942E-03  0.000000000000000000E+00  good
pcgsoi:  gnorm(1:2),b=  4.101618851769856410E+05  4.101618851769853500E+05  4.155702200144395508E-01
cost,grad,step,b,step? =  1  1  3.440099807266351127E+04  6.404388223530688720E+02  5.837551195665082078E-03  4.155702200144395508E-01  good
pcgsoi:  gnorm(1:2),b=  4.679752995326447999E+05  4.679752995326457312E+05  1.140952673675940110E+00
cost,grad,step,b,step? =  1  2  3.200665706943234181E+04  6.840872017021256397E+02  3.727016244273905453E-03  1.140952673675940110E+00  good
pcgsoi:  gnorm(1:2),b=  2.579351145588153449E+05  2.579351145588165091E+05  5.511724973869557287E-01
...
cost,grad,step,b,step? =  1  9  2.515019683100273687E+04  2.511847728725914237E+02  5.552768045789822915E-03  9.872916643640710088E-01  good
pcgsoi:  gnorm(1:2),b=  5.868194961445817898E+04  5.868194961445837544E+04  9.300748853414525508E-01
cost,grad,step,b,step? =  1  10  2.479985164931967302E+04  2.422435749704379191E+02  5.481216572289848883E-03  9.300748853414525508E-01  good

```

The following are the iterations from the second outer loop:

```

Initial cost function =  2.792919782749931983E+04
Initial gradient norm =  4.241369976412337337E+02
cost,grad,step,b,step? =  2  0  2.792919782749931983E+04  4.241369976412337337E+02  4.301269527061492466E-03  0.000000000000000000E+00  good
pcgsoi:  gnorm(1:2),b=  1.641799966598818428E+05  1.641799966598813771E+05  9.126577097845959274E-01
cost,grad,step,b,step? =  2  1  2.715543302058953486E+04  4.051913087171068923E+02  3.965037770683364597E-03  9.126577097845959274E-01  good
pcgsoi:  gnorm(1:2),b=  7.369958699881075881E+04  7.369958699881142820E+04  4.488950450613589660E-01
cost,grad,step,b,step? =  2  2  2.650445313264244396E+04  2.714766785541821719E+02  6.065878225371862040E-03  4.488950450613589660E-01  good
pcgsoi:  gnorm(1:2),b=  5.010117245725526300E+04  5.010117245725520479E+04  6.798026216627733875E-01
...
cost,grad,step,b,step? =  2  9  2.488459155328830457E+04  1.491452772915179139E+02  6.349376870755444636E-03  9.522651754285673675E-01  good
pcgsoi:  gnorm(1:2),b=  2.350218829989638834E+04  2.350218829989681763E+04  1.056548139732611746E+00
cost,grad,step,b,step? =  2  10  2.474335402213209454E+04  1.533042344486817683E+02  5.304880218120413757E-03  1.056548139732611746E+00  good

```

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

- **cost**: the values of cost function, (**=J**)

4. GSI Diagnostics and Tuning

- grad: inner product of gradients (norm of the gradient (Y*X))
- step: stepsize
- b: parameter to estimate the new search direction

As a quick check, the cost function reduced from $3.915930707165839704E+04$ to $2.479985164931967302E+04$ in the 1st outerloop and reduced from $2.792919782749931983E+04$ to $2.474335402213209454E+04$ in the 2nd outer loop.

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

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

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

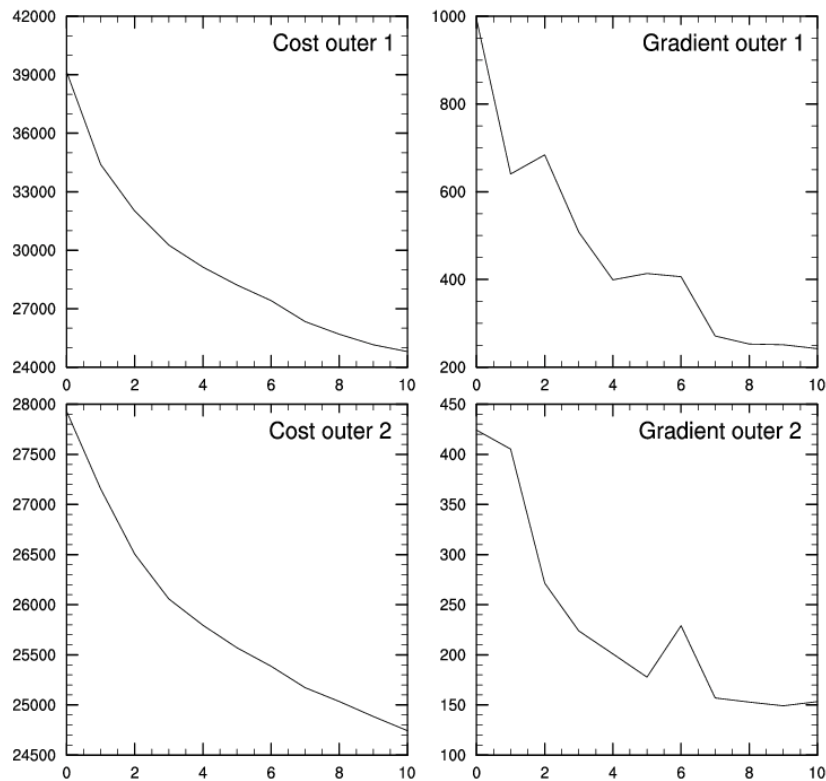


Figure 4.2: Evolution of cost function (left column) and the norm of gradient (right column) in the first outer loop (top row) and the second outer loop (bottom row). The Y-axis is the iteration number

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

4.7 Conventional Observation Errors

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

4.7.1 Getting Original Observation Errors

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

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

The external observation error table file, *errtable*, includes observation errors for all types of conventional observations. It is copied from the `/comGSIv3.5_EnKFv1.1/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 4.10:

```

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

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

```

Table 4.10: Description of each column in the observation error table file

Column #	1	2	3	4	5	6
<i>Content</i>	Pressure	T	RH	UV	Ps	Pw
<i>Unit</i>	hPa	degree C	percent/10	m/s	mb	kg/m2(or mm)

4. GSI Diagnostics and Tuning

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), relative humidity (RH), horizontal wind component (UV), surface pressure (Ps), and the total column precipitable water (Pw). 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 (Ps) observations, while number 200-299 are horizontal wind component (UV) observations. The detailed explanation of each data type number can be found from the following table in the EMC website:

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

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

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

4.7.2 Observation Error Gross Error Check within GSI

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

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

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

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

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

4.8 Background Error Covariance

The GSI package has several files in `/comGSIv3.5_EnKFv1.1/fix/` to hold the pre-computed background error statistics for different GSI applications with different grid configurations. Within the `./fix` directory subdirectories `./fix/Big_Endian` and `./fix/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.

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.

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", which is a copy of `nam_nmmstat_na.gcv` or `nam_glb_berror.f77.gcv`, 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 4.11.

4. GSI Diagnostics and Tuning

Table 4.11: recommended tuning values for the provided B matrix

	Global	Regional
<i>fixed B matrix</i>	nam_glb_berror.f77.gcv	nam_nmmstat_na.gcv
<i>vs</i>	0.7	1.0
<i>hzscl</i>	1.7, 0.8, 0.5	0.373,0.746,1.50
<i>hswgt</i>	0.45, 0.3,0.25	0.45, 0.3,0.25
<i>ss/tsfc_sdv</i>	<pre>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</pre>	<pre>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</pre>

4.9 Analysis Increments

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

4.10 Running Time and Memory Usage

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

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

```
* . . . . .
PROGRAM GSI_ANL HAS BEGUN. COMPILED 1999232.55      ORG: NP23
STARTING DATE-TIME  JUL 02,2016  20:36:21.760  184  SAT  2457572
```

...

4. GSI Diagnostics and Tuning

```
ENDING DATE-TIME    JUL 02,2016  20:43:40.422  184  SAT   2457572
PROGRAM GSI_ANL HAS ENDED.
* . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * . * .
```

This tells us this case started at 20:36:21.760 and ended at 20:43:40.422. Meaning GSI used 7 minutes and 19 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           = 438.663534
The total amount of time in user mode   = 427.578998
The total amount of time in sys mode    = 9.457562
The maximum resident set size (KB)     = 2020132
Number of page faults without I/O activity = 312762
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    = 7641
Number of InVoluntary Context Switches  = 851
*****END OF RESOURCE STATISTICS*****
```



5

GSI Applications for Regional 3DVar and 3D Hybrid EnVar

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 file was chosen to increase the amount of data used in the analysis (comparing to NAM PrepBUFR file)
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

5. GSI Applications for Regional 3DVar and 3D Hybrid EnVar

Note: GDAS data was chosen to get better coverage for radiance and GPS/RO refractivity

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 3.5) or create a new background and get the observation data from the above server. The background and observations used in this case study are as follows:

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

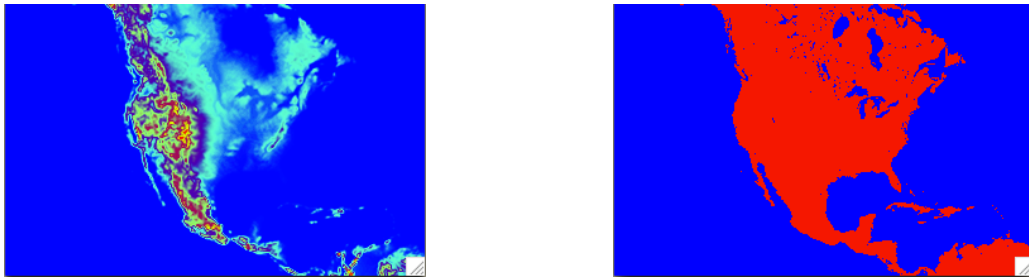


Figure 5.1: The terrain (left) and land mask (right) of the background used in this case study

2. Conventional data: NAM PrepBUFR data from 00UTC 17 June 2014
 - File: *nam.t00z.prepbufr.tm00.nr*
3. Radiance and GPS RO data: GDAS PREPBUFR data from 00 UTC 17 June 2014
 - Files: *gdas.t00z.1bamua.tm00.bufr_d*
gdas.t00z.1bhrs4.tm00.bufr_d
gdas.t00z.gpsro.tm00.bufr_d

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

Assume the background file is located at:

data/2014061700/arw

all the observations are located at:

data/2014061700/obs

and the GSI release version 3.5 is located at

code/comGSIv3.5_EnKFv1.1

5.1 Assimilating Conventional Observations with Regional GSI

5.1.1 Run Script

With GSI successfully compiled and background and observational data acquired, move to the `./run` directory under `./comGSIv3.5_EnKFv1.1` to run the GSI using the sample script `run_gsi_regional.ksh`. The `run_gsi_regional.ksh` script must be modified in several places before running:

- Set up batch queuing system.

To run GSI with multi-processors, a job queuing head has to be added at the beginning of the `run_gsi_regional.ksh` script. The set up of the job queue is dependent on the machine and the job control system. More examples of the setup are described in section 3.2.2. The following example is set up to run on a Linux cluster supercomputer with LSF. The job head is as follows:

```
#BSUB -P ??????????          # project code
#BSUB -W 00:20                # wall-clock time (hrs:mins)
#BSUB -n 4                    # number of tasks in job
#BSUB -R "span[ptile=16]"     # run 16 MPI tasks per node
#BSUB -J gsi                  # job name
#BSUB -o gsi.%J.out           #
#BSUB -e gsi.%J.err           #
#BSUB -q small                # queue
```

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

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

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

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

Set up analysis time:

```
ANAL_TIME=2014061700
```

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

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

Set path to the background directory and file:

```
BK_ROOT=data/20140617/${ANAL_TIME}/arw
BK_FILE=${BK_ROOT}/wrfinput_d01.${ANAL_TIME}
```

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

```
OBS_ROOT=data/20140617/${ANAL_TIME}/obs
PREPBUFR=${OBS_ROOT}/nam.t${HH}z.prepbufr.tm00.nr
```

5. GSI Applications for Regional 3DVar and 3D Hybrid EnVar

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 and the namelist file:

```
CRTM_ROOT=data/fix/CRTM_2.2.3
GSI_ROOT=/comGSIv3.5_EnKFv1.1
FIX_ROOT=${GSI_ROOT}/fix
GSI_EXE=${GSI_ROOT}/run/gsi.exe
GSI_NAMELIST=${GSI_ROOT}/run/comgsi_namelist.sh
```

- Set which background and background error file to use:

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

This example uses the ARW NetCDF background; therefore `bk_core` is set to 'ARW'. The regional background error covariance file is used in this case, as set by `bkcv_option=NAM`. Finally, the run scripts are set to clean the run directory to delete all temporary intermediate files.

5.1.2 Run GSI and Check the Run Status

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

```
$ bsub < run_gsi_regional.ksh
```

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

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

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

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

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

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

- `stdout`: standard out file

5. GSI Applications for Regional 3DVar and 3D Hybrid EnVar

- wrf_inout: background file
- gsiparm.anl: GSI namelist
- prepbuf: 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 stdout
[mhu@yslogin2 testarw_conv]$ tail stdout
pcgsoi: gnorm(1:2),b= 4.971613638691962543E-04 4.971613637988174387E-04 6.539937387039491679E-01
cost,grad,step,b,step? = 2 21 2.283071832605468444E+04 2.229711559527815246E-02 4.235685495878949780E+01 6.539937387039491679E-01 good
pcgsoi: gnorm(1:2),b= 2.4279984557554152E-04 2.427998455925739387E-04 4.883723137754825139E-01
cost,grad,step,b,step? = 2 22 2.283069726786290266E+04 1.558203598942562579E-02 5.370146211199837438E+01 4.883723137754825139E-01 good
pcgsoi: gnorm(1:2),b= 1.349871612172568163E-04 1.349871612633241600E-04 5.559606553423739328E-01
cost,grad,step,b,step? = 2 23 2.283068422915619885E+04 1.161839753224414365E-02 6.137769018004394894E+01 5.559606553423739328E-01 good
pcgsoi: gnorm(1:2),b= 9.116990149652581994E-05 9.116990145686963470E-05 6.753968350377786978E-01
cost,grad,step,b,step? = 2 24 2.283067594395604101E+04 9.548293119533240308E-03 5.219011413742769179E+01 6.753968350377786978E-01 good
pcgsoi: gnorm(1:2),b= 5.783146215365002791E-05 5.783146214669085264E-05 6.343262545796939378E-01
cost,grad,step,b,step? = 2 25 2.283067118578847294E+04 7.604700004184913528E-03 5.599317818817987558E+01 6.343262545796939378E-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          fort.202  fort.214  fort.228          satbias_ang.out
berror_stats     fort.203  fort.215  fort.229          satbias_in
convinfo         fort.204  fort.217  fort.230          satbias_out
diag_conv_anl.2014061700 fort.205  fort.218  gsi.exe          satbias_out.int
diag_conv_ges.2014061700 fort.206  fort.219  gsiparm.anl     satbias_pc
errtable         fort.207  fort.220  l2rwbufr        satbias_pc.out
fit_p1.2014061700 fort.208  fort.221  list_run_directory satinfo
fit_q1.2014061700 fort.209  fort.223  ozinfo          stdout
fit_rad1.2014061700 fort.210  fort.224  pcpbias_out     stdout.anl.2014061700
fit_t1.2014061700 fort.211  fort.225  pcpinfo         wrfanl.2014061700
fit_w1.2014061700 fort.212  fort.226  prepbuf         wrf_inout
fort.201         fort.213  fort.227  prepobs_prep.bufhtable
```

5.1.3 Check for Successful GSI Completion

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

1. Read in the anavinfo and namelist

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

5. GSI Applications for Regional 3DVar and 3D Hybrid EnVar

```

gsi_metguess_mod*init_: 2D-MET STATE VARIABLES:
ps
z
gsi_metguess_mod*init_: 3D-MET STATE VARIABLES:
u
v
... ..

control_vectors*init_anacv: ALL CONTROL VARIABLES
sf          vp
ps          t
q           oz
... ..

GSI_4DVAR: nobs_bins =          1
SETUP_4DVAR: l4dvar= F
SETUP_4DVAR: l4densvar= F
SETUP_4DVAR: winlen=  3.000000000000000
SETUP_4DVAR: winoff=  3.000000000000000
SETUP_4DVAR: hr_obsbin= 3.000000000000000
SETUP_4DVAR: nobs_bins=          1
... ..

&SETUP
GENCODE = 78.00000000000000 ,
FACTQMIN = 0.000000000000000E+000,
FACTQMAX = 0.000000000000000E+000,
CLIP_SUPERSATURATION = F,
FACTV = 1.000000000000000 ,
FACTL = 1.000000000000000 ,
FACTP = 1.000000000000000 ,
FACTG = 1.000000000000000 ,
... ..

```

2. Read in the background field

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

```

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

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

```

3. Read in observational data

5. GSI Applications for Regional 3DVar and 3D Hybrid EnVar

Skipping through a majority of the content towards the middle of the stdout file, the following lines will appear:

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

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

4. Inner iteration

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

```
GLBISOI:  START pcgsoi jiter=          1
pcgsoi:  gnorm(1:2),b=  1.131520548923311509E+02  1.131520548923311509E+02  0.000000000000000000E+00
Begin J table inner/outer loop          0          1
  J term          J
surface pressure  5.7012207042385944E+03
temperature      6.4242087278840627E+03
wind             1.6782607330525603E+04
moisture         3.5878183830232451E+03
-----
J Global          3.2495855145671507E+04
End Jo table inner/outer loop          0          1
Initial cost function =  3.249585514567150676E+04
Initial gradient norm =  1.063729546888358080E+01
cost,grad,step,b,step? =  1  0  3.249585514567150676E+04  1.063729546888358080E+01  2.548553547231620442E+01  0.000000000000000000E+00  good
pcgsoi:  gnorm(1:2),b=  9.711890653545569307E+01  9.711890653545563623E+01  8.583043995786756586E-01
cost,grad,step,b,step? =  1  1  2.961211443694752961E+04  9.854892517701838273E+00  2.514521805501437157E+01  8.583043995786756586E-01  good
pcgsoi:  gnorm(1:2),b=  8.597091659074136771E+01  8.597091659074150982E+01  8.852129791983981422E-01
cost,grad,step,b,step? =  1  2  2.717003835484893352E+04  9.272050290563644381E+00  1.036575164309021346E+01  8.852129791983981422E-01  good
pcgsoi:  gnorm(1:2),b=  5.683515872824821002E+01  5.683515872824790449E+01  6.610975081120487040E-01
...

```

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

```
...
cost,grad,step,b,step? =  2  43  2.283066393607995633E+04  1.877001128106250809E-04  2.191568845986752123E+01  1.034537274286373876E+00  good
pcgsoi:  gnorm(1:2),b=  1.819672329827284049E-08  1.819678976283112548E-08  5.164945106960998622E-01
cost,grad,step,b,step? =  2  44  2.283066393530783535E+04  1.348952308210814343E-04  4.180909725254741716E+01  5.164945106960998622E-01  good
pcgsoi:  gnorm(1:2),b=  1.127913513618353988E-08  1.127903191777782680E-08  6.198386233002942669E-01
cost,grad,step,b,step? =  2  45  2.283066393454704667E+04  1.062032727187987424E-04  2.399192596022169610E+01  6.198386233002942669E-01  good
PCGSOI:  WARNING **** Stopping inner iteration ***
gnorm 0.996812222890348903E-10 less than 0.100000000000000004E-09
update_guess: successfully complete

```

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 3.249585514567150676E+04 at the beginning and a value of 2.283066393454704667E+04 at the final iteration. This means the value has reduced by about one third, which is an expected reduction.

5. Write out analysis results

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

```
... ..
max,min psfc=  102799.9    66793.78
```

5. GSI Applications for Regional 3DVar and 3D Hybrid EnVar

```

max,min MU=    2799.898      -1195.195
rmse_var=MU
ordering=XY
WrfType,WRF_REAL=      104      104
ndim1=      2
staggering= N/A
start_index=      1      1      1      0
end_index1=    332      215      50      0
k,max,min,mid T=      1      321.6157      270.8151      309.3634
k,max,min,mid T=      2      321.7112      270.9660      309.4070
k,max,min,mid T=      3      321.4324      271.2166      309.3831
k,max,min,mid T=      4      321.2418      271.6100      309.3864
k,max,min,mid T=      5      321.6863      272.2831      309.3308
k,max,min,mid T=      6      320.9441      272.8387      309.1977
k,max,min,mid T=      7      320.8511      273.1703      309.0908

```

... ..

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

```

ENDING DATE-TIME    JUL 05,2016  11:30:38.156  187  TUE  2457575
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

a 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 or larger than 20. In the run scripts, several fort files have also been renamed as *fit_t1(q1, p1, rad1, w1).YYYYMMDDHH*. Please check Section 4.5.1 for a detailed explanation of the fit files. Here illustrates how to use these fit files.

- fit_t1.2014061700 (fort.203)*

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

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		107	350	357	866	1153	719	252	450	551	884	745	7188
o-g 01	t	120 0000	bias		0.80	0.32	-0.10	-0.12	-0.15	-0.20	-0.24	-0.60	-0.22	0.15	-0.10	-0.07
o-g 01	t	120 0000	rms		2.06	1.55	0.83	0.77	0.69	0.66	0.73	1.20	1.44	1.65	1.65	1.23
o-g 01	t	120 0000	cpen		0.81	0.49	0.23	0.33	0.33	0.30	0.36	0.79	0.91	0.98	0.79	0.58
o-g 01	t	120 0000	qcpen		0.81	0.49	0.23	0.33	0.33	0.30	0.36	0.79	0.91	0.98	0.79	0.58
o-g 01	t	180 0000	count		339	35	0	0	0	0	0	0	0	0	0	374
o-g 01	t	180 0000	bias		0.17	1.12	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.26
o-g 01	t	180 0000	rms		1.66	4.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	2.01

5. GSI Applications for Regional 3DVar and 3D Hybrid EnVar

o-g 01	t	180 0000	cpen	0.63	7.18	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
o-g 01	t	180 0000	qcpen	0.63	7.18	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
o-g 01	t	180 0001	count	1344	15	0	0	0	0	0	0	0	0	0	0	0	0	1359
o-g 01	t	180 0001	bias	0.82	4.17	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.86
o-g 01	t	180 0001	rms	2.07	5.44	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	2.13
o-g 01	t	180 0001	cpen	0.47	23.37	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.73
o-g 01	t	180 0001	qcpen	0.47	23.37	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.73
o-g 01	all		count	1792	405	358	871	1172	725	325	800	651	884	745	9482			
o-g 01	all		bias	0.69	0.53	-0.10	-0.12	-0.15	-0.19	-0.09	-0.50	-0.04	0.15	-0.10	0.08			
o-g 01	all		rms	1.99	2.14	0.83	0.77	0.69	0.67	0.84	1.32	1.58	1.65	1.65	1.45			
o-g 01	all		cpen	0.52	1.91	0.23	0.33	0.36	0.31	0.44	0.97	1.18	0.98	0.79	0.68			
o-g 01	all		qcpen	0.52	1.91	0.23	0.33	0.36	0.31	0.44	0.97	1.18	0.98	0.79	0.68			

o-g 03	t	120 0000	count	107	350	357	866	1153	719	252	450	551	884	745	7188			
o-g 03	t	120 0000	bias	0.58	0.29	-0.04	-0.02	-0.04	-0.02	0.01	-0.16	-0.04	0.06	0.04	0.01			
o-g 03	t	120 0000	rms	1.72	1.35	0.70	0.61	0.49	0.43	0.50	0.79	1.14	1.40	1.59	1.05			
o-g 03	t	120 0000	cpen	0.57	0.33	0.14	0.19	0.16	0.12	0.18	0.34	0.57	0.72	0.73	0.39			
o-g 03	t	120 0000	qcpen	0.57	0.33	0.14	0.19	0.16	0.12	0.18	0.34	0.57	0.72	0.73	0.39			
o-g 03	t	180 0000	count	339	35	0	0	0	0	0	0	0	0	0	374			
o-g 03	t	180 0000	bias	-0.24	0.21	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-0.19			
o-g 03	t	180 0000	rms	1.55	2.83	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.71			
o-g 03	t	180 0000	cpen	0.34	2.57	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.55			
o-g 03	t	180 0000	qcpen	0.34	2.57	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.55			
o-g 03	t	180 0001	count	1344	16	0	0	0	0	0	0	0	0	0	1360			
o-g 03	t	180 0001	bias	0.30	1.97	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.32			
o-g 03	t	180 0001	rms	1.75	2.88	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.77			
o-g 03	t	180 0001	cpen	0.27	6.05	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.34			
o-g 03	t	180 0001	qcpen	0.27	6.05	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.34			
o-g 03	all		count	1792	406	358	871	1172	725	325	800	651	884	745	9483			
o-g 03	all		bias	0.21	0.34	-0.04	-0.02	-0.04	-0.02	0.04	-0.13	0.06	0.06	0.04	0.05			
o-g 03	all		rms	1.71	1.61	0.69	0.61	0.49	0.43	0.61	0.94	1.26	1.40	1.59	1.22			
o-g 03	all		cpen	0.30	0.75	0.14	0.19	0.18	0.14	0.24	0.49	0.76	0.72	0.73	0.42			
o-g 03	all		qcpen	0.30	0.75	0.14	0.19	0.18	0.14	0.24	0.49	0.76	0.72	0.73	0.42			

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

Skipping ahead in the fort file, "o-g 03" columns (under "it") show the observation minus analysis (O-A) information. Under the summary ("all") lines, it can be seen that there were 9483 total observations, a bias of 0.05, and a rms of 1.22. This shows that from the background to the analysis, one more observation data is being used because of the recalculation of the innovation and the gross check after each outer loop, the bias reduced from 0.08 to 0.05, and the rms reduced from 1.45 to 1.22. This is about a 16% reduction, which is a reasonable value for large-scale analysis.

- *fit_w1.2014061700 (fort.202)*

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

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

O-B: 14513 observations in total, bias is 0.64 and rms is 3.59

O-A: 14520 observations in total, bias is 0.39 and rms is 2.90

The total bias was reduced from 0.64 to 0.39 and the rms reduced from 3.59 to 2.90 (20% reduction).

5. GSI Applications for Regional 3DVar and 3D Hybrid EnVar

- *fit_q1.2014061700 (fort.204)*

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

it	obs	type	styp	ptop pbot	1000.0 1200.0	950.0 1000.0	900.0 950.0	850.0 900.0	800.0 850.0	700.0 800.0	600.0 700.0	500.0 600.0	400.0 500.0	300.0 400.0	0.0 300.0	0.0 2000.0
o-g 01		all	count		543	186	182	211	146	457	406	520	621	623	0	3895
o-g 01		all	bias		1.17	-3.68	-2.47	-1.30	-3.55	0.19	0.64	-1.80	-4.28	-5.55	0.00	-2.05
o-g 01		all	rms		9.09	10.63	9.03	9.34	12.73	12.30	14.53	15.27	16.45	16.01	0.00	13.66
o-g 03		all	count		543	186	182	211	146	457	406	520	621	623	0	3895
o-g 03		all	bias		-0.39	-0.88	-0.68	0.45	-0.51	0.06	0.13	-0.10	-0.70	-1.90	0.00	-0.53
o-g 03		all	rms		5.48	5.19	4.37	5.73	8.13	9.31	12.19	13.82	13.01	12.36	0.00	10.64

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

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

The total bias and rms were reduced.

- *fit_p1.2014061700 (fort.201)*

This file demonstrates how the background and analysis fields fit to surface pressure observations. Because the surface pressure is a 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:

```
-----
pressure levels (hPa)= 0.0 2000.0
  it  obs  type  stype  count  bias  rms  cpen  qcpen
o-g 01          all      13890  0.1912  0.7931  0.4105  0.4105
-----
o-g 03          all      13916  0.0403  0.6764  0.2921  0.2921
```

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

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

Both the total bias and rms were reduced.

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

b 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
```

5. GSI Applications for Regional 3DVar and 3D Hybrid EnVar

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

```

1  0  3.249585514567150676E+04  1.063729546888358080E+01  2.548553547231620442E+01  0.000000000000000000E+00
1  1  2.961211443694752961E+04  9.854892517701838273E+00  2.514521805501437157E+01  8.583043995786756586E-01
1  2  2.717003835484893352E+04  9.272050290563644381E+00  1.036575164309021346E+01  8.852129791983981422E-01
1  3  2.627888518494048549E+04  7.538909651153024249E+00  1.948409284114243079E+01  6.610975081120487040E-01
1  4  2.517150367563822510E+04  5.715354258100989071E+00  2.423238066649372513E+01  5.747370998254690555E-01
... ..
2  41  2.283066394128102547E+04  2.700371175626514980E-04  4.661917669555924704E+01  3.046618826707093719E-01
2  42  2.283066393788155256E+04  1.845403996927800583E-04  5.290241773629001187E+01  4.670206697407201513E-01
2  43  2.283066393607995633E+04  1.877001128106250809E-04  2.191568845986752123E+01  1.034537274286373876E+00
2  44  2.283066393530783535E+04  1.348952308210814343E-04  4.180909725254741716E+01  5.164945106960998622E-01
2  45  2.283066393454704667E+04  1.062032727187987424E-04  2.399192596022169610E+01  6.198386233002942669E-01

```

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 fourth 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_Uutilities/plot_ncl/GSI_cost_gradient.ncl.
```

the plot is shown as Fig.5.2:

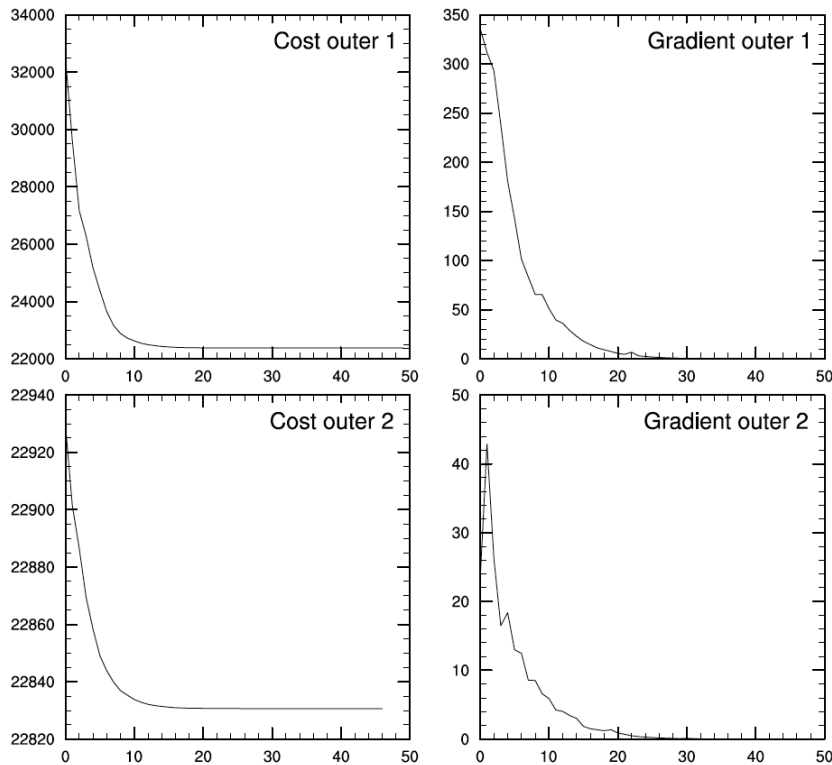


Figure 5.2: Cost function (y-axes) and norm of gradient(y-axes) change with iteration steps (x-axes).

5. GSI Applications for Regional 3DVar and 3D Hybrid EnVar

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.

c 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_Uutilities/plot_ncl/Analysis_increment.ncl.
```

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

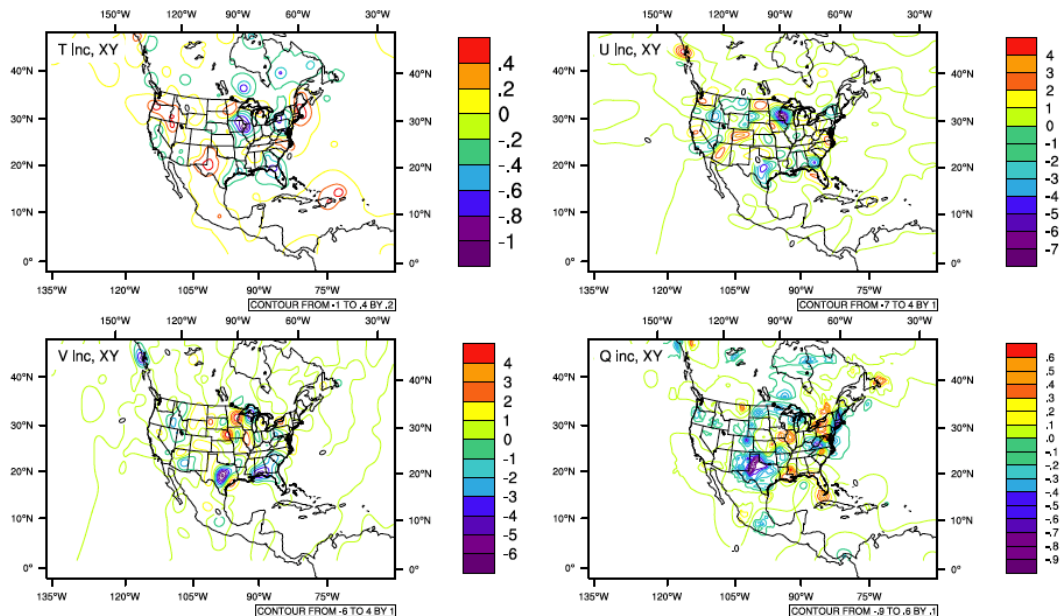


Figure 5.3: Analysis increment at the 15th level

The analysis increment shows the fact that the conventional observations are mostly located in the U.S. CONUS domain and the data availability over the ocean is very sparse.

5.2 Assimilating Radiance Data with Regional GSI

5.2.1 Run Script

Adding radiance data into the GSI analysis is very straightforward after a successful run of GSI with conventional data. The same run script from the above section can be used to run

5. GSI Applications for Regional 3DVar and 3D Hybrid EnVar

GSI with radiance with or without PrepBUFR data. The key step to adding the radiance data is linking the radiance BUFR data files to the GSI run directory with the names listed in the `&OBS_INPUT` section of the GSI namelist. The following example adds the two radiance BUFR files:

```
AMSU-A: gdas1.t00z.1bamua.tm00.bufr_d
HIRS4:  gdas1.t00z.1bhrs4.tm00.bufr_d
```

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

```
ln -s ${OBS_ROOT}/gdas1.t${HH}z.1bamua.tm00.bufr_d amsuabufr
ln -s ${OBS_ROOT}/gdas1.t${HH}z.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_regional.ksh*:

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

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

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

```
! dfile      dtype      dplat      dsis          dval      dthin dsfcalc
   amsuabufr  amsua      n15        amsua_n15    10.0      2      0
```

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

- Radiance data thinning

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

5. GSI Applications for Regional 3DVar and 3D Hybrid EnVar

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

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

- Radiance data bias correction

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

```
# for satellite bias correction
cp ${FIX_ROOT}/gdas1.t00z.abias.20150617 ./satbias_in
cp ${FIX_ROOT}/gdas1.t00z.abias_pc.20150617 ./satbias_pc
```

For this case, the GDAS bias correction files were downloaded and saved in the `fix` directory as examples. For different cases, the run script should have those two lines to link the bias correction coefficient files. The first line sets the path to the bias coefficient file, and the second copies the bias correction coefficients for passive (monitored) channels into the working directory. Those two coefficient files are usually calculated from within GSI in the previous cycle. 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. Please note the GSI release version before v3.5 has coefficients for mass bias correction and angle bias correction calculated separately.

Once these links are set, we are ready to run the case.

5.2.2 Run GSI and Check Run Status

The process for running GSI is the same as described in section 5.1.2. Once `run_gsi_regional.ksh` has been submitted, move into the run directory to check the GSI analysis results. For our current case, the run directory will look almost as it did for the conventional data case, the exception being the two links to the radiance BUFR files and new diag files for the radiance data types used. Following the same steps as in section 5.1.2, check the `stdout` file to see if GSI has run through each part of the analysis process successfully. In addition to the information outlined for the conventional run, the radiance BUFR files should have been read in and distributed to each sub domain:

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

5. GSI Applications for Regional 3DVar and 3D Hybrid EnVar

OBS_PARA: hirs4	metop-a	0	0	1146	1661
OBS_PARA: hirs4	n19	213	1020	0	0
OBS_PARA: hirs4	metop-b	0	0	85	555
OBS_PARA: amsua	n15	1458	2026	830	234
OBS_PARA: amsua	n18	2223	2318	108	0
OBS_PARA: amsua	n19	176	960	0	0
OBS_PARA: amsua	metop-a	0	0	1077	1559
OBS_PARA: amsua	metop-b	0	0	265	1829

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

5.2.3 Diagnose GSI Analysis Results

a 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	83190	58236	25226	10356.	10356.	0.41053	0.41053
o-g 01 rad	n18	amsua	83595	69147	27677	11067.	11067.	0.39988	0.39988

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

o-g 03 rad	n15	amsua	83190	58236	30136	4672.4	4672.4	0.15504	0.15504
o-g 03 rad	n18	amsua	83595	69147	32253	8546.8	8546.8	0.26499	0.26499

From the above information, it can be seen that AMSU-A data from NOAA-15 have 83190 observations within the analysis time window and domain. After thinning, 58236 of this data type remained, and only 25226 passed quality check and were used in the analysis. The penalty for this data decreased from 10356 to 4672.4 after 2 outer loops. It is also very interesting to see that the number of AMSU-A observations assimilated in the O-A calculation increased to 30136 from 25226 because more data passed quality check in 2nd outer loop.

The statistics for each channel can be viewed in the *fort.207* file as well. Below channels from AMSU-A NOAA-15 are listed as an example:

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

5. GSI Applications for Regional 3DVar and 3D Hybrid EnVar

1	1	amsua_n15	1903	24	3.000	1.6543287	-0.3164878	0.1411234	1.7151923	1.6857402
2	2	amsua_n15	1927	0	2.200	1.0105548	-0.2430249	0.0385590	1.0284683	0.9993427
3	3	amsua_n15	1927	0	2.000	1.7941589	-0.1480894	0.0575956	0.7909800	0.7769935
4	4	amsua_n15	1927	0	0.600	-0.1848763	-0.0460476	0.0856369	0.2497797	0.2454985
5	5	amsua_n15	1927	4	0.300	0.0314288	-0.0292998	0.3025052	0.2008865	0.1987383
6	6	amsua_n15	4126	10	-0.230	-1.8448526	-0.0778284	0.8969513	0.2463875	0.2337725
7	7	amsua_n15	4468	13	0.250	-0.1497841	-0.0810899	0.5245399	0.2042760	0.1874916
8	8	amsua_n15	4468	13	0.275	-0.0251081	-0.0869918	0.6195120	0.2420568	0.2258847
9	9	amsua_n15	4463	18	0.340	0.1824903	0.2108491	0.6987405	0.3453174	0.2734717
10	10	amsua_n15	294	4187	0.400	0.7946961	0.7908103	2.5281681	0.7982470	0.1087077
15	15	amsua_n15	1922	5	3.500	1.6785936	-0.1471413	0.0940713	1.6461928	1.6396037

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

1	1	amsua_n15	2050	10	3.000	2.0842622	0.1363398	0.0414276	1.0332965	1.0242622
2	2	amsua_n15	2060	0	2.200	1.0926873	-0.0731534	0.0177924	0.8048238	0.8014923
3	3	amsua_n15	2060	0	2.000	1.8733559	-0.0257656	0.0406385	0.6841159	0.6836305
4	4	amsua_n15	2060	0	0.600	-0.1278763	0.0136412	0.0547146	0.2103718	0.2099291
5	5	amsua_n15	2060	4	0.300	0.0730860	0.0118626	0.1759475	0.1624923	0.1620587
6	6	amsua_n15	4234	0	-0.230	-1.7911421	-0.0262042	0.5247873	0.1975867	0.1958414
7	7	amsua_n15	4470	11	0.250	-0.0766482	-0.0094307	0.1918646	0.1288789	0.1285334
8	8	amsua_n15	4475	6	0.275	0.0675371	-0.0047226	0.1715888	0.1341703	0.1340871
9	9	amsua_n15	4481	0	0.340	-0.0508290	-0.0396425	0.1722334	0.1711928	0.1665396
10	10	amsua_n15	4362	119	0.400	0.2373520	0.1943598	0.3425567	0.3140220	0.2466457
15	15	amsua_n15	2058	2	3.500	1.8261335	0.0617429	0.0487809	1.2317674	1.2302190

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

b 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).

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

1. The weighting functions of each channel and the data coverage at this analysis time. There are several sources on the Internet to show the weighting function for the AMSU-A channels. Channel 1 is the moisture channel, while the others are mainly temperature channels (Channels 2, 3 and 15 also have large moisture signals). Because a model top of 20 mb was specified for this case study, the actual impact should come from channels with the peak of the weighting below 20hPa.
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

5. GSI Applications for Regional 3DVar and 3D Hybrid EnVar

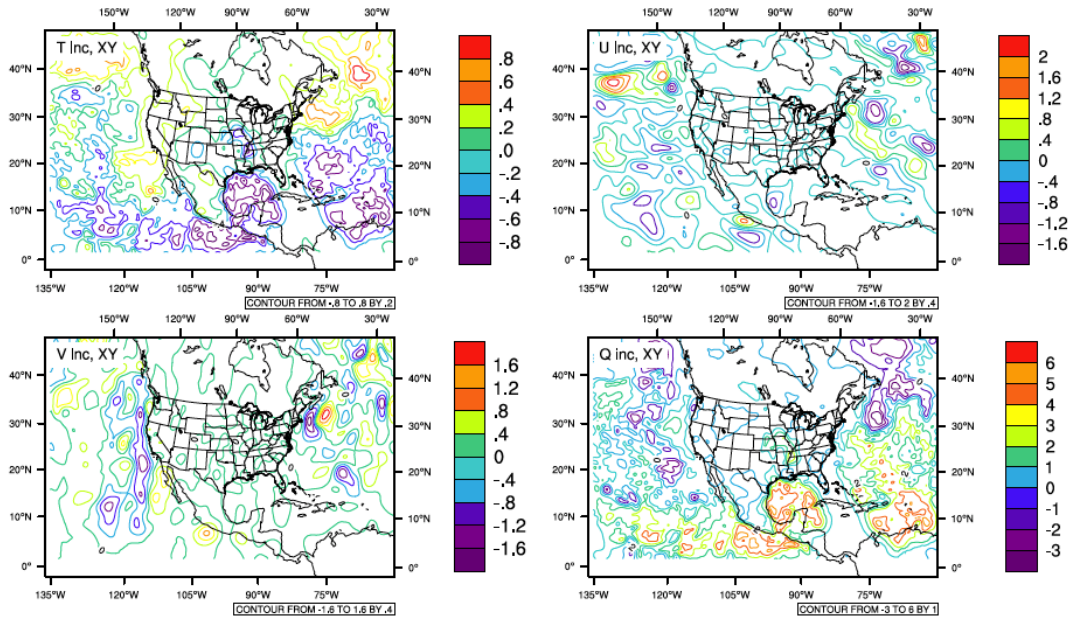


Figure 5.4: Analysis increment fields of PrepBUFR and Radiance data analysis comparing to the analysis with PREPBUFR only at level 6

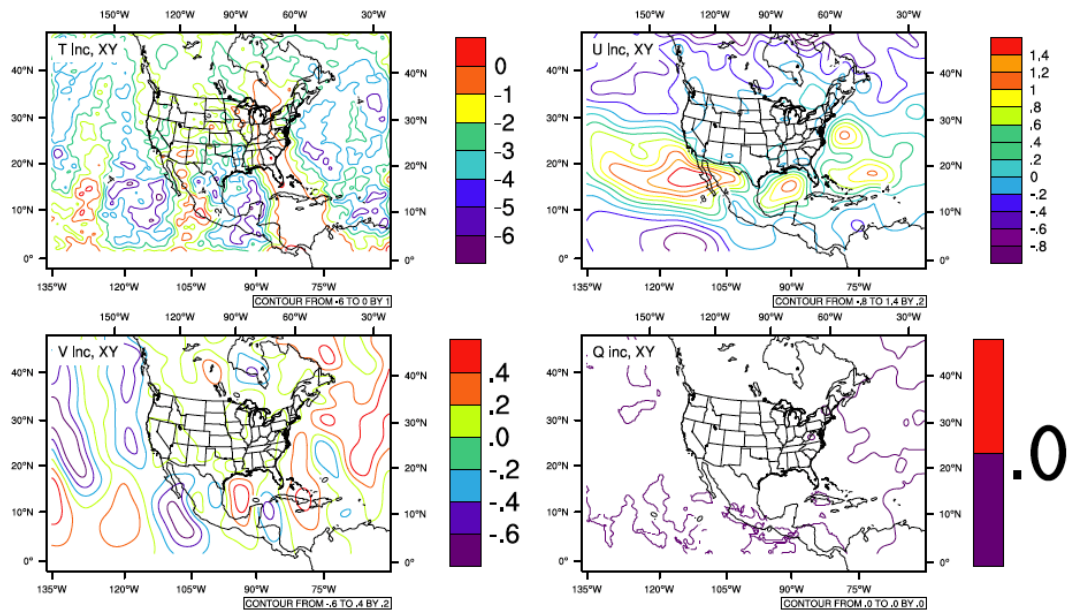


Figure 5.5: Analysis increment fields PrepBUFR and Radiance data analysis comparing to the analysis with PREPBUFR only at level 49

amsua_n15 and amsua_n18 data were used, they should be checked in detail. In this case, Channels 6, 11 and 14 from amsua_n15 and channels 9 and 14 from amsua_n18 were turned off.

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

5. GSI Applications for Regional 3DVar and 3D Hybrid EnVar

for a successful radiance data analysis. The run script can only link to the GDAS bias correction coefficients that are provided as an example in *./fix*:

```
cp ${FIX_ROOT}/gdas1.t00z.abias.20150617 ./satbias_in
cp ${FIX_ROOT}/gdas1.t00z.abias_pc.20150617 ./satbias_pc
```

Users can 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 ${OBS_ROOT}/gdas1.t${HH}z.gpsro.tm00.bufr_d gpsrobufr
```

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

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

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

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

5.3.2 Run GSI and Check the Run Status

The process of running GSI is the same as described in section 5.1.2. Once *run_gsi_regional.ksh* has been submitted, move into the working directory, *gsiprd_2014061700_gps_prepbufr*, to check the GSI analysis results. The run directory will look exactly the same as with the conventional data, with the exception of the link to

5. GSI Applications for Regional 3DVar and 3D Hybrid EnVar

the GPS RO BUFR files used in this case. Following the same steps as in section 5.1.3, check the stdout file to see if GSI has run through each part of the analysis process successfully. In addition to the information outlined for the conventional run, the GPS RO BUFR files should have been read in and distributed to each sub domain:

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

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

5.3.3 Diagnose GSI Analysis Results

a Check File *fort.212*

The file *fort.212* shows the fit of analysis/background to the GPS/RO 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)

```

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

```

Observation - Analysis (O-A)

```

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

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

b 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.

5. GSI Applications for Regional 3DVar and 3D Hybrid EnVar

The following figures give detailed information of how the new data impacts the analysis result. Using the NCL script used in section ??, 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.

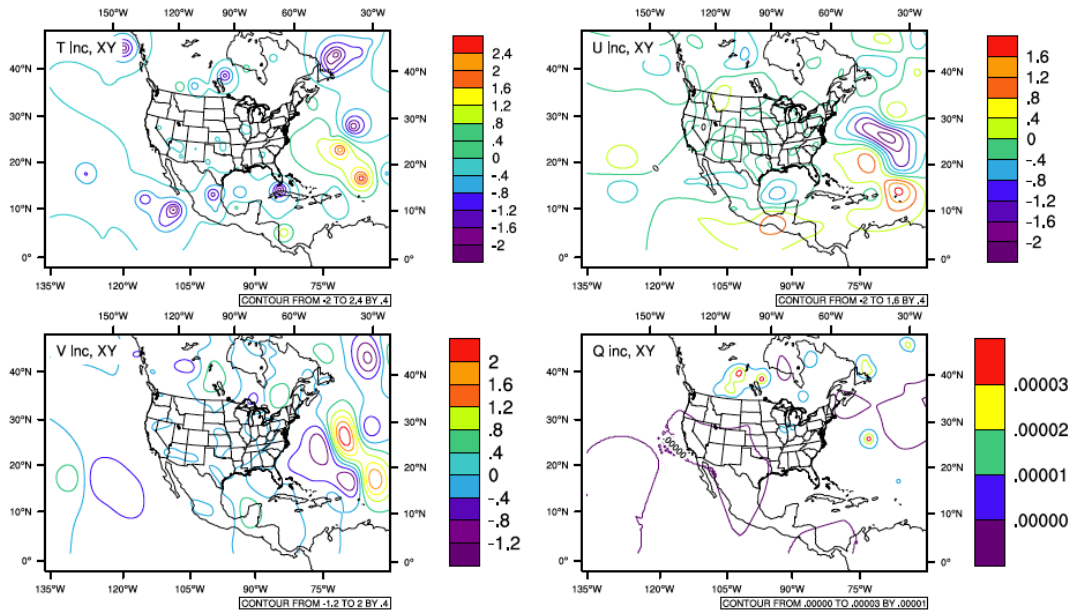


Figure 5.6: Analysis increment fields comparing the use of GPS RO and conventional observations to create the analysis to only PrepBUFR at level 48.

5.4 Introduction to GSI 3D Hybrid EnVar Analysis

The 3 dimensional hybrid ensemble-variational (3D hybrid EnVar) 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.

Users have to change the GSI run script to add the links to the ensemble forecasts if they want to use the GSI hybrid function. Below is an example of using ensembles of ARW netcdf format, assuming that all the ensemble members are located in a directory

5. GSI Applications for Regional 3DVar and 3D Hybrid EnVar

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	<i>filelist</i> : a text file including path and name of ensemble files
2	ensembles are WRF NMM (HWRF) format	get_wrf_nmm_enspers	<i>d01_en001</i> , <i>d01_en002</i> , ...
3	ensembles are ARW netcdf format	get_wrf_mass_enspers_netcdf	<i>wrf_en001</i> , <i>wrf_en002</i> , ...
4	ensembles are NEMS NMMB format	get_nmmb_enspers	<i>nmmb_ens_mem001</i> , <i>nmmb_ens_mem002</i> , ...

defined by the parameter *\$mempath* and the ensemble members have a name such as: *wrfout_d01_\$\$iimem*, where *\$\$iimem* is an integer indicating the ensemble member ID. The following lines should be added to the run script with loop *iiimem* from 1 to the total number of ensemble members:

```
if [ -r ${mempath}/wrfout_d01_$$iimem ]; then
  ln -sf ${mempath}/wrfout_d01_$$iimem ./wrf_en$$iimem
else
  echo "member ${mempath}/wrfout_d01_$$iimem does not exist"
fi
```

- *Change 2: Set up the namelist options in section HYBRID_ENSEMBLE*

Users need to set `l_hyb_ens=.true.` to turn on hybrid ensemble analysis. Commonly used namelist options for the hybrid analysis are listed in table 5.2:

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.

5.5 Summary

This chapter applied the knowledge from the previous chapters to demonstrate how to set up, run, and analyze GSI for various regional 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.

5. GSI Applications for Regional 3DVar and 3D Hybrid EnVar

Table 5.2: the list of namelist options for GSI hybrid

Options	explanation
l_hyb_ens	if true, turn on hybrid ensemble option;
uv_hyb_ens	if true, ensemble perturbation wind variables are u and v; otherwise, ensemble perturbation wind variables are stream function and velocity potential functions.
generate_ens	if true, generate internal ensemble based on existing background error; recommended to be false.
n_ens	number of ensemble members.
beta1_inv	(1/beta1), the weight given to the static background error covariance. $0 \leq \text{beta1_inv} \leq 1$, should be tuned for optimal performance; $\text{beta2_inv} = 1 - \text{beta1_inv}$ 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.

5. GSI Applications for Regional 3DVar and 3D Hybrid EnVar

It is important to realize that GSI applications are not limited to regional analysis with WRF. Other GSI applications, including the global analysis for GFS, the chemical analysis, and others will be introduced in the next chapter .



6

Introduction to more GSI Applications

6.1 Introduction to Global GSI analysis

The *Global Forecast System (GFS)* is a global numerical weather prediction system containing a global computer model and variational analysis run by the U.S. National Weather Service (NWS). As of February 2015, the mathematical model is run four times a day, and produces forecasts for up to 16 days in advance, with decreased spatial resolution after 10 days. The model is a spectral model with a resolution of T1534 from 0 to 240 hours (0-10 days) and T574 from 240 to 384 hours (10-16 days). In the vertical, the model is divided into 64 layers and temporally, it produces forecast output every hour for the first 12 hours, every 3 hours out to 10 days, and every 12 hours after that. Its data assimilation system runs 6-hourly continuous cycles using the GSI-hybrid.

GSI has many functions specially designed and tuned for GFS. Although the release version of the community GSI includes all the functions used by the operational systems, the DTC can only support the GSI regional applications because the DTC is not able to run GFS on community computers. Beginning with release version 3.2, the DTC began to introduce the use of GSI for global applications, assuming users can obtain the GFS background through the NCEP data hub or by running GFS themselves.

6.1.1 The Difference between Global and Regional GSI

As mentioned above, all of the NCEP operational systems use GSI as their analysis system. The majority of the GSI code is common to these operational systems. Very little of the source code is specific to a particular operational system. The main differences in GSI operational application come from the configuration the run scripts and namelist parameters.

The different GSI applications need different backgrounds, observations, and fixed files. For the GFS system, GSI needs:

- **GFS Backgrounds:** typically, GSI uses 6-h GFS forecasts as the background. GFS 3-h and 9-h forecasts are also needed for the FGAT function in the GSI analysis. Both surface and atmosphere forecasts are needed.
- **Observations:** NCEP has several sets of BUFR/PrepBUFR observations files with global coverage for global systems. The files that start with the prefix **GDAS** are for the 6-hourly global data assimilation system. These files have more data available for the analysis, but have a longer delay for use in real-time. The files that start with **gfs** are for the 4 time daily GFS forecast. The different operational systems need different observation data files because they require different kinds of observations with different coverage, cut-off times, and quality control processes. All these observation files are read in and processed in GSI by the same section of code. Therefore, there is no problem using GFS observation data files for regional GSI applications, as is described in the practice cases and the GSI User's Guide. Using regional BUFR files for global applications will cause scientific problems since the data only covers part of the analysis domain, but GSI can still read in the observations and perform the analysis.
- **Fixed files:** Section 3.1 of the GSI User's Guide introduced the notion that different operational systems have their own fixed files. These global fixed files can be downloaded as a separate tar ball from the GSI user's website (<http://www.dtcenter.org/com-GSI/users/downloads/index.php>). For the GFS GSI application, the big difference is the background error covariance (BE). Different resolutions of the GFS backgrounds use their matched BE files, which are different from the BE files used by the regional GSI applications. In release version 3.4, ten new BE files were provided for users (in addition to the two for release version 3.3):

1. global_berror.l64y386.f77
2. global_berror.l64y96.f77
3. global_berror.l64y1154.f77
4. global_berror.l64y290.f77
5. global_berror.l64y882.f77
6. global_berror.l64y130.f77
7. global_berror.l64y192.f77
8. global_berror.l64y578.f77
9. global_berror.l64y258.f77
10. global_berror.l64y674.f77

6.1.2 Global GFS Scripts

Starting with release version 3.3, support for running GSI global applications was added to the community release. Specifically a sample global analysis run script was added to the `./run` directory named `run_gsi_global.ksh` of the community release. This script `run_gsi_global.ksh` is based on the GSI GFS regression tests. This script retains the same basic structure as the regional run script `run_gsi_regional.ksh`, but includes additional details needed for the global analysis.

Table 6.1: The grid dimensions for GFS

SPECTRAL RESOLUTION	EULERIAN		SEMI-LAGRANGIAN	
	LONB	LATB	LONB	LATB
T62	192	94	128	64
T126	384	190	256	128
T170	512	256	352	176
T190	576	288	384	192
T254	768	384	512	256
T382	1152	576	768	384
T574	1760	880	1152	576
T878	2304	1152	1760	880
T1148			2304	1152
T1534			3072	1536
T2014			4032	2016
T2046			4096	2048
T3070			6144	3072

The first part of the global analysis run script, just as in the regional script, sets up the computer environment and case configuration. The primary differences between the global and regional are the specification of the GFS case and the global application namelist.

```
GFSCASE=T126
GSI_NAMELIST=${GSI_ROOT}/run/comgsi_namelist_gfs.sh
```

While the regional script simply specifies the background and BE files, the global script needs to know the background resolution setup by defining the following parameters:

```
# Set the JCAP resolution which you want.
# All resolutions use LEVS=64
if [[ "$GFSCASE" = "T62" ]]; then
  JCAP=62
  JCAP_B=62
elif [[ "$GFSCASE" = "T126" ]]; then
  JCAP=126
  JCAP_B=126
elif [[ "$GFSCASE" = "enkf_glb_t254" ]]; then
  JCAP=254
  JCAP_B=254
elif [[ "$GFSCASE" = "T254" ]]; then
  JCAP=254
```

6. Introduction to more GSI Applications

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

Just as with the regional analysis run script, the global script double checks the needed parameters, creates a run directory, copies the background, observations, and fixed files into the run directory. It generates the namelist, and places that in the run directory as well.

1. Specify the values of LATA, LONA, DELTIME, resol based on the choice of JCAP:

```
# Given the requested resolution, set dependent resolution parameters
if [[ "$JCAP" = "382" ]]; then
  LONA=768
  LATA=384
  DELTIM=180
  resol=1
elif [[ "$JCAP" = "574" ]]; then
  LONA=1152
  LATA=576
  DELTIM=1200
  resol=2
elif [[ "$JCAP" = "254" ]]; then
  LONA=512
  LATA=256
  DELTIM=1200
  resol=2
elif [[ "$JCAP" = "126" ]]; then
  LONA=256
  LATA=128
  DELTIM=1200
  resol=2
elif [[ "$JCAP" = "62" ]]; then
  LONA=192
  LATA=94
  DELTIM=1200
  resol=2
else
  echo "INVALID JCAP = $JCAP"
  exit
fi
NLAT=' expr $LATA + 2 '
```

2. Set up CO₂, CH₄, N₂O, CO file decisions:

```
IC02=${IC02:-0}
if [ $IC02 -gt 0 ] ; then
  # Copy co2 files to $workdir
  co2dir=${FIX_ROOT}
  yyyy='echo $ANAL_TIME | cut -c1-4'
  rm ./global_co2_data.txt
  co2=${co2dir}/global_co2.gcmssl_`$yyyy`.txt
  while [ ! -s $co2 ] ; do
    ((yyyy-=1))
    co2=${co2dir}/global_co2.gcmssl_`$yyyy`.txt
  done
  if [ -s $co2 ] ; then
    cp $co2 ./global_co2_data.txt
  fi
  if [ ! -s ./global_co2_data.txt ] ; then
    echo "\./global_co2_data.txt" not created
    exit 1
  fi
fi
```

6. Introduction to more GSI Applications

```
#CH4 file decision
...
```

3. Set up the namelist parameters and generate the namelist:

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

vs_op='0.7,'
hzscl_op='1.7,0.8,0.5,'

...

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

$comgsi_namelist

EOF
```

4. Multiple time level backgrounds are needed:

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

cp $BK_ROOT/sfcanl_2014040506_fhr03_ensmean ./sfcf03
cp $BK_ROOT/sfcanl_2014040506_fhr06_ensmean ./sfcf06
cp $BK_ROOT/sfcanl_2014040506_fhr09_ensmean ./sfcf09

cp $BK_ROOT/sfg_2014040506_fhr03_mem001    ./sigf03
cp $BK_ROOT/sfg_2014040506_fhr06_mem001    ./sigf06
cp $BK_ROOT/sfg_2014040506_fhr09_mem001    ./sigf09
else
cp $OBS_ROOT/satbias_in      ./satbias_in
cp $OBS_ROOT/satbias_angle   ./satbias_angle

cp $BK_ROOT/sfcf03 ./sfcf03
cp $BK_ROOT/sfcf06 ./sfcf06
cp $BK_ROOT/sfcf09 ./sfcf09

cp $BK_ROOT/sigf03 ./sigf03
cp $BK_ROOT/sigf06 ./sigf06
cp $BK_ROOT/sigf09 ./sigf09
fi
```

Both surface and atmosphere files at 03, 06, 09 hour forecasts are needed.

5. More observations files are available

In the sample run script, many more observations are listed for use:

```
# Link to the other observation data

if [[ "$GFSCASE" = "enkf_glb_t254" ]]; then
obsfile_amua=gdas1.t12z.1bamua.tm00.bufr_d
obsfile_amub=gdas1.t12z.1bamub.tm00.bufr_d
else
obsfile_amua=amsuabufr
obsfile_amub=amsubbufr
fi

if [ -r "${OBS_ROOT}/satwnd" ]; then
ln -s ${OBS_ROOT}/satwnd .
fi

if [ -r "${OBS_ROOT}/gpsrobufr" ]; then
ln -s ${OBS_ROOT}/gpsrobufr .
fi

.....
```

6. Introduction to more GSI Applications

```

if [ -r "${OBS_ROOT}/amsubbufrears" ]; then
  ln -s ${OBS_ROOT}/amsubbufrears .
fi
if [ -r "${OBS_ROOT}/tcvitl" ]; then
  ln -s ${OBS_ROOT}/tcvitl .
fi

```

Table 6.1 lists the grid dimensions for GFS of different resolutions from *Running Global Model Parallel Experiments Version 6.0* from NCEP/EMC, which may provide users more information on the above mentioned resolution parameters:

6.1.3 Sample Results

After a successful run of the GSI GFS analysis, the contents of the run directory, with the clean option turned on, will look something like this:

amsrebufr	diag_mhs_n18_anl.2011080100.gz	fort.213
amsuabufr	diag_mhs_n18_ges.2011080100.gz	fort.214
amsubbufr	diag_mhs_n19_anl.2011080100.gz	fort.215
anavinfo	diag_mhs_n19_ges.2011080100.gz	fort.217
atms_beamwidth.txt	diag_omi_aura_anl.2011080100.gz	fort.218
berror_stats	diag_omi_aura_ges.2011080100.gz	fort.219
bftab_sstphr	diag_pcp_tmi_trmm_anl.2011080100.gz	fort.220
convinfo	diag_pcp_tmi_trmm_ges.2011080100.gz	fort.221
diag_amsre_hig_aqua_anl.2011080100.gz	diag_sbu2_n16_anl.2011080100.gz	gomebufr
diag_amsre_hig_aqua_ges.2011080100.gz	diag_sbu2_n16_ges.2011080100.gz	gpsrobufr
diag_amsre_low_aqua_anl.2011080100.gz	diag_sbu2_n17_anl.2011080100.gz	gsi.exe
diag_amsre_low_aqua_ges.2011080100.gz	diag_sbu2_n17_ges.2011080100.gz	gsiparm.anl
diag_amsre_mid_aqua_anl.2011080100.gz	diag_sbu2_n18_anl.2011080100.gz	hirs3bufr
diag_amsre_mid_aqua_ges.2011080100.gz	diag_sbu2_n18_ges.2011080100.gz	hirs4bufr
diag_amsua_metop-a_anl.2011080100.gz	diag_sbu2_n19_anl.2011080100.gz	mhsbufr
diag_amsua_metop-a_ges.2011080100.gz	diag_sbu2_n19_ges.2011080100.gz	omibufr
diag_amsua_n15_anl.2011080100.gz	diag_seviri_m09_anl.2011080100.gz	ozinfo
diag_amsua_n15_ges.2011080100.gz	diag_seviri_m09_ges.2011080100.gz	pcpbias_out
diag_amsua_n18_anl.2011080100.gz	errtable	pcpinfo
diag_amsua_n18_ges.2011080100.gz	fit_p1.2011080100	prepbufr
diag_amsua_n19_anl.2011080100.gz	fit_q1.2011080100	prepobs_prep.bufrtable
diag_amsua_n19_ges.2011080100.gz	fit_rad1.2011080100	satbias_angle
diag_amsub_n17_anl.2011080100.gz	fit_t1.2011080100	satbias_in
diag_amsub_n17_ges.2011080100.gz	fit_w1.2011080100	satbias_out
diag_conv_anl.2011080100.gz	fort.201	satinfo
diag_conv_ges.2011080100.gz	fort.202	satwnd
diag_gome_metop-a_anl.2011080100.gz	fort.203	sbu2bufr
diag_gome_metop-a_ges.2011080100.gz	fort.204	scaninfo
diag_hirs3_n17_anl.2011080100.gz	fort.205	seviri2bufr
diag_hirs3_n17_ges.2011080100.gz	fort.206	sfc anl.gsi
diag_hirs4_metop-a_anl.2011080100.gz	fort.207	siganl
diag_hirs4_metop-a_ges.2011080100.gz	fort.208	stdout
diag_hirs4_n19_anl.2011080100.gz	fort.209	stdout.anl.2011080100
diag_hirs4_n19_ges.2011080100.gz	fort.210	tcvitl
diag_mhs_metop-a_anl.2011080100.gz	fort.211	tmirrbufr
diag_mhs_metop-a_ges.2011080100.gz	fort.212	

The majority of these files were seen after running the GSI regional analysis examples in section 3.2.3 of the Basic User's Guide, and they provide the same information about the GSI run. Of note, the GSI global analysis run includes more radiance observations, resulting in more radiance diag files in this list. Instead of the single background file `wrf_inout` as seen with the regional analysis, the global analysis background is split between the two files `siganl`, for atmosphere, and `sfc anl.gsi` for the surface. A quick check of the standard output file `stdout` shows information similar to that generated by the regional runs for the

6. Introduction to more GSI Applications

namelist, data ingest, and minimization, but is quite different with respect to information on the background IO.

Please visit our online tutorial for more details regarding how to conduct a global GSI run.

6.2 Introduction to Chemical Analysis

The GSI has been developed to analyze chemical observations, such as MODIS AOD or PM2.5, to improve the pollution forecast with chemical models. In this release, GSI can do the following chemical analyses:

Table 6.2: List of GSI chemical analyses

case	Chemical Model	background species	Observations
1	WRF-Chem	GOCART	MODIS AOD
2	WRF-Chem	GOCART	PM2.5
3	WRF-Chem	PM2.5	PM2.5
4	CMAQ	CMAQ	PM2.5

The GSI run scripts for chemical analysis (`./run/run_gsi_chem.ksh`) and namelist (`./run/comgsi_namelist_chem.sh`) are provided with this release. Sample background and observation files are provided through the on-line tutorial.

6.2.1 Setup GSI Run Scripts for Chemical Analysis

The script `run_gsi_chem.ksh` was built based on regional GSI run scripts and has a similar structure to the regional run script `run_gsi_regional.ksh`, but include a couple of different details.

The first part of the run script sets up the computer environment and case configuration. This is similar to the regional analysis run scripts, except for the specification of the chemical cases (`bk_core` and `obs_type`), and the namelist for the chemical application:

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

#-----
# bk_core= set background (WRFCHEM_GOCART WRFCHEM_PM25 or CMAQ)
# obs_type= set observation type (MODISAOD or PM25)
bk_core=CMAQ
obs_type=PM25
```

The choice of the chemical cases (`bk_core` and `obs_type`) need to match with the options `PREPBUFR` and `BK_FILE`, which set background and observation files. Table 6.3 shows how to setup these two options for each case:

Similar to the regional run script, this chemical run script will also double check the needed parameters. Then it creates a run directory and generates the namelist in the directory and

6. Introduction to more GSI Applications

Table 6.3: List of GSI chemical analyses

case	background (<i>BK_FILE</i> ; <i>bk_core</i>)	Observation (<i>PREPBUFR</i> ; <i>obs_type</i>)
1	wrfinput_enkf_d01_2012-06-03_18:00:00; WRFCHEM_GOCART	Aqua_Terra_AOD_BUFR:2012-06-03_00:00:00; MODISAOD
2	wrfinput_enkf_d01_2012-06-03_18:00:00; WRFCHEM_GOCART	anow.2012060318.bufr; PM25
3	wrfinput_enkf_d01_2012-06-03_18:00:00; WRFCHEM_PM25	anow.2012060318.bufr; PM25
4	cmaq2gsi_4.7_20130621_120000.bin; CMAQ	anow.2013062112.bufr; PM25

copies the background, observations, and fixed files into the run directory. Users who run the cases listed in table 6.2 do not need to change the rest of the run scripts. But users who need to build new cases may need to know the differences between chemical and regional applications, which is shown below.

1. Specify the name of the background and observations:

```
# Bring over background field (it's modified by GSI so we can't link to it)

if [ ${bk_core} = WRFCHEM_GOCART ] ; then
  cp ${BK_FILE} ./wrf_inout
fi
if [ ${bk_core} = WRFCHEM_PM25 ] ; then
  cp ${BK_FILE} ./wrf_inout
fi
if [ ${bk_core} = CMAQ ] ; then
  cp ${BK_FILE} ./cmaq_in.bin
fi

# Link to the observation data
if [ ${obs_type} = MODISAOD ] ; then
  ln -s ${PREPBUFR} ./modisbufr
fi
if [ ${obs_type} = PM25 ] ; then
  ln -s ${PREPBUFR} ./pm25bufr
fi
```

2. Specify the background error file and anavinfo:

```
if [ ${bk_core} = WRFCHEM_GOCART ] ; then
  BERROR=${FIX_ROOT}/wrf_chem_berror_little_endian
  BERROR_CHEM=${FIX_ROOT}/wrf_chem_berror_little_endian
  ANAVINFO=${FIX_ROOT}/anavinfo_wrfchem_gocart
fi
if [ ${bk_core} = WRFCHEM_PM25 ] ; then
  BERROR=${FIX_ROOT}/wrf_chem_berror_little_endian
  BERROR_CHEM=${FIX_ROOT}/wrf_chem_berror_little_endian
  ANAVINFO=${FIX_ROOT}/anavinfo_wrfchem_pm25
fi
if [ ${bk_core} = CMAQ ] ; then
  BERROR=${FIX_ROOT}/cmaq_berror_little_endian
  BERROR_CHEM=${FIX_ROOT}/cmaq_berror_little_endian
  ANAVINFO=${FIX_ROOT}/anavinfo_cmaq_pm25
fi
```

3. Specify the options for building the namelist:

```
if [ ${bk_core} = WRFCHEM_GOCART ] ; then
  bk_core_arw='.true.'
  bk_if_netcdf='.true.'
  bk_core_cmaq='.false.'
  bk_wrf_pm2_5='.false.'
  bk_laeroana_gocart='.true.'
fi
if [ ${bk_core} = WRFCHEM_PM25 ] ; then
```

6. Introduction to more GSI Applications

```

bk_core_arw='.true.'
bk_if_netcdf='.true.'
bk_core_cmaq='.false.'
bk_wrf_pm2_5='.true.'
bk_laeroana_gocart='.false.'
fi
if [ ${bk_core} = CMAQ ] ; then
bk_core_arw='.false.'
bk_if_netcdf='.false.'
bk_core_cmaq='.true.'
bk_wrf_pm2_5='.false.'
bk_laeroana_gocart='.false.'
fi

```

6.2.2 Sample Results

In this section, the case 1 in Table 6.2 will be used as example. After a successful run of the GSI Chem analysis, the contents of the run directory, with the clean option turned on, will look something like this:

```

aeroinfo          fit_w1.2012060318  fort.212  fort.226          pcpinfo
anavinfo          fort.201          fort.213  fort.227          prepobs_prep.bufhtable
berror_stats     fort.202          fort.214  fort.228          satbias_angle
berror_stats_chem fort.203          fort.215  fort.229          satbias_in
convinfo         fort.204          fort.217  fort.230          satbias_out
diag_conv_anl.2012060318 fort.205          fort.218  gsi.exe          satinfo
diag_conv_ges.2012060318 fort.206          fort.219  gsiparm.anl      stdout
errtable        fort.207          fort.220  l2rwbuf          stdout.anl.2012060318
fit_p1.2012060318 fort.208          fort.221  list_run_directory wrfanl.2012060318
fit_q1.2012060318 fort.209          fort.223  modisbuf          wrf_inout
fit_rad1.2012060318 fort.210          fort.224  ozinfo
fit_t1.2012060318 fort.211          fort.225  pcpbias_out

```

Following the instruction in Chapter 5, the following steps are conducted to check the results of this GSI chemical analysis:

1. Check stdout file:

- Read in chemical background fields:

```

rmse_var=sulf
ordering=XYZ
WrfType,WRF_REAL=      104      104
ndim1=      3
staggering= N/A
start_index=      1      1      1      0
end_index=      112      122      40      0
k,max,min,mid var=sulf
3.3167184E-13  0.7885093
k,max,min,mid var=sulf
3.3167214E-13  0.7910572
... ..
rmse_var=BC1
...
rmse_var=BC2
...
rmse_var=OC1
...
rmse_var=OC2
...

```

- Read in modis AOD observations:

6. Introduction to more GSI Applications

```

OBS_PARA: modis_aod terra          7      64      50      11      34      25      7      81
          40      31      50      27      38      46      56      25      7      0
          26      53      62      60      19      0
OBS_PARA: modis_aod aqua          29      34      44      23      50      18      55      76
          33      30      49      22      20      67      76      30      2      0
          36      76      38      50      14      0

```

- **Minimizations:**

```

Begin J table inner/outer loop          0          1
      J term                             J
aerosol aod                             7.8897778549100276E+03
-----
J Global                                7.8897778549100276E+03
End Jo table inner/outer loop          0          1
Initial cost function = 7.88977E+03
Initial gradient norm = 1.531557E+02
cost,grad,step,b,step? = 1 0 7.8897778E+03 1.5315573E+02 1.3701618E-01 0.0000000E+00 good
pcgsoi: gnorm(1:2),b= 5.2207374E+03 5.220737604157E+03 2.225693512490E-01
cost,grad,step,b,step? = 1 1 4.67583332E+03 7.2254671E+01 1.4308398E-01 2.2256935E-01 good
...

pcgsoi: gnorm(1:2),b= 1.254432996913E-06 1.254434051E-06 2.97203315E-01
cost,grad,step,b,step? = 2 16 3.15885215E+03 1.12001473E-03 2.3966297E-01 2.9720331E-01 good
PCGSOI: WARNING **** Stopping inner iteration ***
gnorm 0.534787149781733860E-10 less than 0.100000004E-09
update_guess: successfully complete
\end{scriptsize}
\end{tiny}

\item Update chemical background fields:
\begin{scriptsize}
\begin{verbatim}
...
k,max,min,mid var=sulf                38 9.1036431E-02
4.5203370E-07 1.8112745E-02
k,max,min,mid var=sulf                39 2.5694052E-02
6.2565640E-08 5.7557132E-03
k,max,min,mid var=sulf                40 1.1622031E-02
1.4633585E-09 8.4061045E-03
rmse_var=sulf
...

```

2. Analysis increments: After successfully run through, the analysis increment should be check the see if the data impact are reasonable.

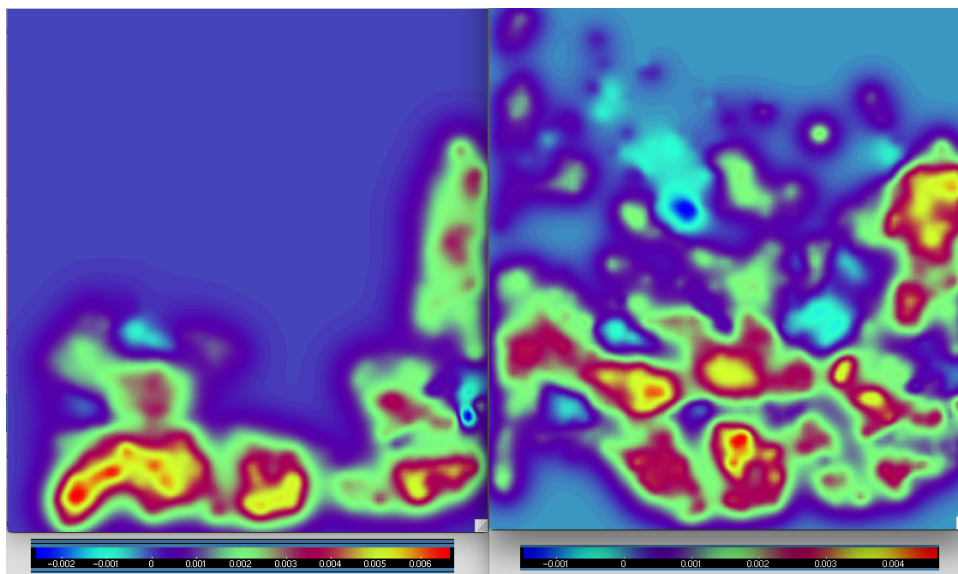


Figure 6.1: The analysis increment in the lowest level for SEAS_1 (left) and BC1 (right)



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 *gdasl.t12z.prepbufr.nr* in tutorial case) are in Big Endian binary format. For release version older than 3.2, the BUFR files have to be converted from Big Endian BUFR file to Little Endian file when used by the GSI on Linux platform. Please refer to older version User's guide on how to convert.

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,

A. GSI Community Tools

namelist variable *write_diag* in namelist section *&SETUP* needs to be true. The *write_diag* variable has been introduced in Part 4 of Section 3.4. The following is an example of using the *write_diag* variable to control diagnostic files. When we set the number of outer loops to 2, and set the *write_diag* namelist variable to the following:

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

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

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

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

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

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

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

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

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

To compile the programs, use the makefile provided:

```
./make
```

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

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

A. GSI Community Tools

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

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

```
./read_diag_conv.exe
```

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

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

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

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

For the conventional observations, the data is stored in two arrays: *rdiagbuf* and *cdiagbuf*. Their contents are listed as follows, for temperature (check *src/main/setupt.f90*) as an example:

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

For wind observations, the content after index 16 (check *src/main/setupw.f90*) is:

```
rdiagbuf(17) = earth relative u wind component observation (m/s)
```

A. GSI Community Tools

```

rdiagbuf(18) = earth relative u obs-ges used in analysis (m/s)
rdiagbuf(19) = earth relative u obs-ges w/o bias correction (m/s)
rdiagbuf(20) = earth relative v wind component observation (m/s)
rdiagbuf(21) = earth relative v obs-ges used in analysis (m/s)
rdiagbuf(22) = earth relative v 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. For example:

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

For wind, the last 4 columns are the wind components in the order of: U observation, O-B for U, V observation, O-B for V.

For radiance observations, the data is stored in two arrays: *diagbuf* and *diagbufchan*. Their contents are listed as follows (please refer to *src/main/setuprad.f90* for more details):

```

diagbuf(1) = observation latitude (degrees)
diagbuf(2) = observation longitude (degrees)
diagbuf(3) = model (guess) elevation at observation location
diagbuf(4) = observation time (hours relative to analysis time)

diagbuf(5) = sensor scan position
diagbuf(6) = satellite zenith angle (degrees)
diagbuf(7) = satellite azimuth angle (degrees)
diagbuf(8) = solar zenith angle (degrees)
diagbuf(9) = solar azimuth angle (degrees)
diagbuf(10) = sun glint angle (degrees) (sgagl)

diagbuf(11) = surface fractional coverage by water
diagbuf(12) = surface fractional coverage by land
diagbuf(13) = surface fractional coverage by ice
diagbuf(14) = surface fractional coverage by snow
if(.not. retrieval)then
diagbuf(15) = surface temperature over water (K)
diagbuf(16) = surface temperature over land (K)
diagbuf(17) = surface temperature over ice (K)
diagbuf(18) = surface temperature over snow (K)
diagbuf(19) = soil temperature (K)
diagbuf(20) = soil moisture
diagbuf(21) = surface land type
else
diagbuf(15) = SST first guess used for SST retrieval
diagbuf(16) = NCEP SST analysis at t
diagbuf(17) = Physical SST retrieval
diagbuf(18) = Navy SST retrieval
diagbuf(19) = d(ta) corresponding to sstph
diagbuf(20) = d(qa) corresponding to sstph
diagbuf(21) = data type
endif
diagbuf(22) = vegetation fraction
diagbuf(23) = snow depth
diagbuf(24) = surface wind speed (m/s)
! Note: The following quantities are not computed for all sensors
if (.not.microwave) then
diagbuf(25) = cloud fraction (%)
diagbuf(26) = cloud top pressure (hPa)
else
diagbuf(25) = cloud liquid water (kg/m**2)
diagbuf(26) = total column precip. water (km/m**2)
endif
endif

```

A. GSI Community Tools

```
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_Uutilities/plot_cost_grad* directory:

The ksh script, *filter_fort220.ksh*, only has one line:

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

To run *filter_fort220.ksh*, the *fort.220* needs to be in the same directory. The script will filter out the values of the cost function and the norm of gradient at each iteration from *fort.220* into a text file called *cost_gradient.txt*.

Once the file *cost_gradient.txt* is ready, run ncl script *GSI_cost_gradient.ncl* to generate the plot:

```
ncl GSI_cost_gradient.ncl
```

The pdf file *GSI_cost_gradient.pdf* is created. The pdf file contains plots of the convergence of the GSI analysis like those in section 4.6.

A.4 Plot Single Observation Test Result and Analysis Increment

In Section 4.2, we introduced how to do a single observation test for GSI. Here we provide users with the ncl scripts to plot the single observation test results.

There are 5 ncl scripts in the *util/Analysis_Uutilities/plots_ncl* directory:

Table A.1: the list of ncl plotting tools

GSI_singleobs_arw.ncl	Plot single observation test with ARW NetCDF background
GSI_singleobs_nmm.ncl	Plot single observation test with NMM NetCDF background
Analysis_increment.ncl	Plot analysis increment from the case with ARW NetCDF background
Analysis_increment_nmm.ncl	Plot analysis increment from the case with NMM NetCDF background
fill_nmm_grid2.ncl	E grid to A grid convertor

The main difference between the ARW and NMM core used in GSI is that ARW is on a C grid, while NMM is on an E grid. *GSI_singleobs_nmm.ncl* calls *fill_nmm_grid2.ncl* to convert the E grid to an A grid for plotting, while *GSI_singleobs_arw.ncl* itself includes a C grid to A grid convertor.

Before running ncl scripts, users need to set up two links:

- *cdf_analysis* - Link to analysis result in NetCDF format (*wrf_inout*)
- *cdf_bk* - Link to background file in netCDF format

These scripts read in the analysis and background fields of temperature (T), U component of wind (U), V component of wind (V), and moisture (Q) and calculate the difference of the analysis field minus the background field. Then XY sections (left column) and XZ sections (right column) are plotted for T, U, V, and Q through the point that has maximum analysis increment of single observation. Here the default single observation test is T. If the user conducts other single observation tests, the corresponding changes should be made based on the current scripts.

The scripts *Analysis_increment.ncl* and *Analysis_increment_nmm.ncl* are very similar the one for the single observation but only the XY section for a certain analysis level is plotted.

For more information on how to use ncl, please check the NCL website at:

<http://www.ncl.ucar.edu/>



Contents of Namelist Section

OBS_INPUT

```
&OBS_INPUT
  dmesh(1)=120.0,dmesh(2)=60.0,dmesh(3)=30,time_window_max=1.5,ext_sonde=.true.,
/
OBS_INPUT::
!  dfile      dtype      dplat      dsis          dval      dthin dsfcalc
  prepbufr    ps          null       ps            1.0       0       0
  prepbufr    t          null       t             1.0       0       0
  prepbufr    q          null       q             1.0       0       0
  prepbufr    pw         null       pw            1.0       0       0
  satwndbufr  uv         null       uv            1.0       0       0
  prepbufr    uv         null       uv            1.0       0       0
  prepbufr    spd        null       spd           1.0       0       0
  prepbufr    dw         null       dw            1.0       0       0
  radarbufr   rw         null       rw            1.0       0       0
  prepbufr    sst        null       sst           1.0       0       0
  gpsrobufr   gps_ref     null       gps           1.0       0       0
  ssmirrbufr  pcp_ssmi   dmstp     pcp_ssmi     1.0      -1       0
  tmirrbufr   pcp_tmi    trmm      pcp_tmi     1.0      -1       0
  sbuvbufr    sbuv2      n16       sbuv8_n16    1.0       0       0
  sbuvbufr    sbuv2      n17       sbuv8_n17    1.0       0       0
  sbuvbufr    sbuv2      n18       sbuv8_n18    1.0       0       0
  hirs3bufr   hirs3      n16       hirs3_n16    0.0       1       0
  hirs3bufr   hirs3      n17       hirs3_n17    6.0       1       0
  hirs4bufr   hirs4      metop-a   hirs4_metop-a 6.0       2       0
  hirs4bufr   hirs4      n18       hirs4_n18    0.0       1       0
  hirs4bufr   hirs4      n19       hirs4_n19    1.0       2       0
  hirs4bufr   hirs4      metop-b   hirs4_metop-b 1.0       1       0
  gimgrbufr   goes_img    g11       imgr_g11     0.0       1       0
  gimgrbufr   goes_img    g12       imgr_g12     0.0       1       0
  airsbufr    airs        aqua      airs281SUBSET_aqua 20.0      2       0
  amsuabufr   amsua      n15       amsua_n15    10.0      2       0
  amsuabufr   amsua      n18       amsua_n18    10.0      2       0
  amsuabufr   amsua      n19       amsua_n19    10.0      2       0
  amsuabufr   amsua      metop-a   amsua_metop-a 10.0      2       0
```

B. Contents of Namelist Section OBS_INPUT

amsuabufr	amsua	metop-b	amsua_metop-b	10.0	2	0
airsbufr	amsua	aqua	amsua_aqua	5.0	2	0
amsubbufr	amsub	n17	amsub_n17	1.0	1	0
mhsbufr	mhs	n18	mhs_n18	3.0	2	0
mhsbufr	mhs	n19	mhs_n19	3.0	2	0
mhsbufr	mhs	metop-a	mhs_metop-a	3.0	2	0
mhsbufr	mhs	metop-b	mhs_metop-b	3.0	2	0
ssmitbufr	ssmi	f13	ssmi_f13	0.0	2	0
ssmitbufr	ssmi	f14	ssmi_f14	0.0	2	0
ssmitbufr	ssmi	f15	ssmi_f15	0.0	2	0
amsrebufr	amsre_low	aqua	amsre_aqua	0.0	2	0
amsrebufr	amsre_mid	aqua	amsre_aqua	0.0	2	0
amsrebufr	amsre_hig	aqua	amsre_aqua	0.0	2	0
ssmisbufr	ssmis_las	f16	ssmis_f16	0.0	2	0
ssmisbufr	ssmis_uas	f16	ssmis_f16	0.0	2	0
ssmisbufr	ssmis_img	f16	ssmis_f16	0.0	2	0
ssmisbufr	ssmis_env	f16	ssmis_f16	0.0	2	0
gsnd1bufr	sndrd1	g12	sndrD1_g12	1.5	1	0
gsnd1bufr	sndrd2	g12	sndrD2_g12	1.5	1	0
gsnd1bufr	sndrd3	g12	sndrD3_g12	1.5	1	0
gsnd1bufr	sndrd4	g12	sndrD4_g12	1.5	1	0
gsnd1bufr	sndrd1	g11	sndrD1_g11	1.5	1	0
gsnd1bufr	sndrd2	g11	sndrD2_g11	1.5	1	0
gsnd1bufr	sndrd3	g11	sndrD3_g11	1.5	1	0
gsnd1bufr	sndrd4	g11	sndrD4_g11	1.5	1	0
gsnd1bufr	sndrd1	g13	sndrD1_g13	1.5	1	0
gsnd1bufr	sndrd2	g13	sndrD2_g13	1.5	1	0
gsnd1bufr	sndrd3	g13	sndrD3_g13	1.5	1	0
gsnd1bufr	sndrd4	g13	sndrD4_g13	1.5	1	0
gsnd1bufr	sndrd1	g15	sndrD1_g15	1.5	2	0
gsnd1bufr	sndrd2	g15	sndrD2_g15	1.5	2	0
gsnd1bufr	sndrd3	g15	sndrD3_g15	1.5	2	0
gsnd1bufr	sndrd4	g15	sndrD4_g15	1.5	2	0
iasibufr	iasi	metop-a	iasi616_metop-a	20.0	1	0
gomebufr	gome	metop-a	gome_metop-a	1.0	2	0
omibufr	omi	aura	omi_aura	1.0	2	0
sbuvbufr	sbuv2	n19	sbuv8_n19	1.0	0	0
tcvltl	tcp	null	tcp	1.0	0	0
seviribufr	seviri	m08	seviri_m08	1.0	1	0
seviribufr	seviri	m09	seviri_m09	1.0	1	0
seviribufr	seviri	m10	seviri_m10	1.0	1	0
iasibufr	iasi	metop-b	iasi616_metop-b	0.0	1	0
gomebufr	gome	metop-b	gome_metop-b	0.0	2	0
atmsbufr	atms	npp	atms_npp	0.0	1	0
crisbufr	cris	npp	cris_npp	0.0	1	0
mlsbufr	mls30	aura	mls30_aura	0.0	0	0
oscatbufr	uv	null	uv	0.0	0	0
prepbufr	mta_cld	null	mta_cld	1.0	0	0
prepbufr	gos_ctp	null	gos_ctp	1.0	0	0
refInGSI	rad_ref	null	rad_ref	1.0	0	0
lghtInGSI	lghtn	null	lghtn	1.0	0	0
larcInGSI	larccld	null	larccld	1.0	0	0
::						



GSI Namelist: Name, Default Value, Explanation

The following are lists and explanations of the GSI namelist variables. You can also find them in the source code **gsimod.F90**.

Variable name	Default value	Description
SETUP		General control namelist
gencode	80	source generation code
factqmin	1	weighting factor for negative moisture constraint
factqmax	1	weighting factor for supersaturated moisture constraint
clip_supersaturation	.false.	flag to remove supersaturation during each outer loop
factv	1	weighting factor for negative visibility constraint
factl		
factp		
factg		
factw10m		
facthowv		
deltim	1200	model timestep
dtphys	3600	physics timestep
biascor	-1	background error bias correction coefficient
bcoption	1	0=ibc (no bias correction to bkg); 1= sbc(original implementation)
diurnalbc	0	1= diurnal bias; 0= persistent bias

C. GSI Namelist: Name, Default Value, Explanation

Variable name	Default value	Description
SETUP		General control namelist
niter(0:50)	0,...	Maximum number of inner loop iterations for each outer loop
niter_no_qc(0:50)	1000000	Inner loop iteration at which to turn on variational quality control
miter	1	number of outer loops
qoption	1	option for moisture analysis variable; 1:q/qsatg 2:normalized RH
cwoption		
pseudo_q2	.false.	breed between q1/q2 options, that is, (q1/sig(q))
nhr_assimilation	6	assimilation time interval (currently 6 hours for global, 3 hours for regional)
min_offset	3	time of analysis in assimilation window
iout_iter	220	output file number for iteration information
npredp	6	number of predictors for precipitation bias correction
retrieval	.false.	logical to turn off or on the SST physical retrieval
nst_gsi	0	indicator to control the Tr Analysis mode: 0 = no nst info ingi at all; 1 = input nst info, but used for monitoring only 2 = input nst info, and used in CRTM simulation, but no Tr analysis 3 = input nst info, and used in CRTM simulation and Tr analysis is on
nst_tzr	0	indicator to control the Tzr_QC mode: 0 = no Tz retrieval; 1 = Do Tz retrieval and applied to QC
nstinfo	0	number of nst variables
fac_dtl	0	index to apply diurnal thermocline layer or not: 0 = no; 1 = yes
fac_tsl	0	index to apply thermal skin layer or not: 0 = no; 1 = yes.
nst_tzr		
tzr_bufsave	.false.	logical to turn off or on the bufr Tz retrieval file true=on
diag_rad	.true.	logical to turn off or on the diagnostic radiance file (true=on)
diag_pcp	.true.	logical to turn off or on the diagnostic precipitation file (true=on)
diag_conv	.true.	logical to turn off or on the diagnostic conventional file (true=on)
diag_ozone	.true.	logical to turn off or on the diagnostic ozone file (true=on)
diag_aero	.false.	logical to turn off or on the diagnostic aerosol file (true=on)
diag_co	.false.	logical to turn off or on the diagnostic carbon monoxide file (true=on)
iguess	1	flag for guess solution (currently not working) -1 do not use guess file 0 write only guess file 1 read and write guess file 2 read only guess file
write_diag	.false., ...	logical to write out diagnostic files for outer iteration
reduce_diag	.false.	namelist logical to produce reduced radiance diagnostic files

C. GSI Namelist: Name, Default Value, Explanation

Variable name	Default value	Description
SETUP		General control namelist
oneobtest	.false.	one observation test flag true=on
sfcmodel	.false.	if true, then use boundary layer forward model for surface temperature data.
dtbduv_on	.true.	logical for switching on (.true.) sensitivity of uv winds to microwave brightness temperatures. if true, use d(microwave brightness temperature)/d(uv wind) in inner loop
ifact10	0	flag for recomputing 10m wind factor = 1 compute using GFS surface physics = 2 compute using MM5 surface physics = 0 or any other value - DO NOT recompute - use value from guess file
l_foto	.false.	option for First-Order Time extrapolation to observation
offtime_data	.false.	if true, then allow use of obs files with ref time different from analysis time. default value = .false., in which case analysis fails if observation file reference time is different from analysis time.
npred_conv_max	0	maximum number of conventional observation bias correction coefficients
id_bias_ps	0	prepbufr id to have conv_bias added for testing
id_bias_t	0	prepbufr id to have conv_bias added for testing
id_bias_spd	120	prepbufr id to have conv_bias added for testing
conv_bias_ps	0	magnitude of ps bias(mb)
conv_bias_t	0	magnitude of t bias(deg K)
conv_bias_spd	0	magnitude of spd bias(m/sec)
id_bias_pm2_5		
conv_bias_pm2_5		
id_bias_pml0		
conv_bias_pml0		
stndev_conv_ps	1.0	
stndev_conv_t	1.0	
stndev_conv_spd	1.0	
use_pbl	.false.	Logical flag to include PBL effects in tendency model.
use_compress	.false.	option to turn on the use of compressibility factors in geopotential heights
nsig_ext	13	number of layers above the model top which are necessary to compute the bending angle for gpsro
gpstop	30.0	maximum height for gpsro data assimilation. Reject anything above this height. (km)
perturb_obs	.false.	logical flag to perturb observation (true=on)
perturb_fact	1	magnitude factor for observation perturbation
oberror_tune	.false.	logical to tune (=true) oberror
preserve_restart_date	.false.	if true, then do not update regional restart file date.
crtm_coeffs_path	./	path of directory w/ CRTM coeffs files
berror_stats	berror_stats	filename if other than "berror_stats"
newpc4pred	.false.	option for additional preconditioning for pred coeff
adp_anglebc	.false.	option to perform variational angle bias correction
angord	0	order of polynomial for variational angle bias correction

C. GSI Namelist: Name, Default Value, Explanation

Variable name	Default value	Description
SETUP		General control namelist
passive_bc	.false.	option to turn on bias correction for passive (monitored) channels
use_edges	.true.	option to exclude radiance data on scan edges
biaspredvar	0.1	set background error variance for radiance bias coeffs
lobsdiagsave	.false.	write out additional observation diagnostics
l4dvar	.false.	turn 4D-Var on/off (default=off=3D-Var)
lbicg	.false.	use B-precond w/ bi-conjugate gradient for minimization
lsqrtb	.false.	Use sqrt(B) preconditioning
lcongrad	.false.	Use conjugate gradient/Lanczos minimizer
lbfgsmin	.false.	Use L-BFGS minimizer
ltilint	.false.	Use TL inner loop (ie TL intall)
nhr_obsbin	-1	length of observation bins
nhr_subwin	-1	length of weak constraint 4d-Var sub-window intervals
nwrvecs	-1	Number of precond vectors (Lanczos) or pairs of vectors (QN) being saved
iorthomax	0	max number of vectors used for orthogonalization of various CG options
ladtest	.false.	Run adjoint test
ladtest_obs	.false.	if true, doing the adjoint check for the observation operators
lgrtest	.false.	Run gradient test
lobskeep	.false.	keep obs from first outer loop for subsequent OL
lsensrecompute	.false.	does adjoint by recomputing forward solution
jsiga	-1	calculate approximate analysis errors from lanczos for jiter=jsiga
lrcost	.false.	calculate true cost when using Lanczos (this is very expensive)
lobsensfc	.false.	compute forecast sensitivity to observations
lobsensjb	.false.	compute Jb sensitivity to observations
lobsensincr	.false.	compute increment sensitivity to observations
lobsensadj	.false.	use adjoint of approx. Hessian to compute obs sensitivity
lobsensmin	.false.	use minimisation to compute obs sensitivity
iobsconv	0	compute convergence test in observation space =1 at final point, =2 at every iteration
idmodel	.false.	uses identity model when running 4D-Var (test purposes)
iwrinc	.false.	when .t., writes out increments instead of analysis
jiterstart	1	first outloop iteration number
jiterend	1	last outloop iteration number
lobserver	.false.	when .t., calculate departure vectors only
lanczosave	.false.	save lanczos vectors for forecast sensitivity computation
llancdone	.false.	use to tell adjoint that Lanczos vecs have been pre-computed
lferrscale	.false.	Something related to forecast error
print_diag_pcg	.false.	logical turn on of printing of GMAO diagnostics in pcgsoi.f90
tsensible	.false.	option to use sensible temperature as the analysis variable. Works only for twodvar_regional=.true.
lgschmidt	.false.	option for re-biorthogonalization of the gradx and grady set from pcgsoi when twodvar_regional=.true.

C. GSI Namelist: Name, Default Value, Explanation

Variable name	Default value	Description
SETUP		General control namelist
lread_obs_save	.false.	option to write out collective obs selection info
lread_obs_skip	.false.	option to read in collective obs selection info
use_gfs_ozone	.false.	option to read in gfs ozone and interpolate to regional model domain
check_gfs_ozone_date	.false.	option to date check gfs ozone before interpolating to regional model domain
regional_ozone	.false.	option to turn on ozone in regional analysis
lwrite_predterms	.false.	option to write out actual predictor terms instead of predicted bias to the radiance diagnostic files
lwrite_peakwt	.false.	option to writ out the approximate pressure of the peak of the weighting function for satellite data to the radiance diagnostic files
use_gfs_nemsio	.false.	option to use nemsio to read global model NEMS/GFS first guess
liauon	.false.	treat 4dvar CV as tendency perturbation (default=false)
use_prepb_satwnd	.false.	allow using satwnd's from prepbufr (historical) file
l4densvar	.false.	logical to turn on ensemble 4dvar
ens4d_nstarthr	3	start hour for ensemble perturbations (generally should match min_offset)
use_gfs_stratosphere		When true, a guess gfs valid at the same time as the nems-nmmb guess is used to replace the upper levels with gfs values. The purpose of this is to allow direct use of gdas derived sat radiance bias correction coefs.
pblend0	152	The nems-nmmb vertical coordinate is smoothly merged with gfs above this level. Below this level, is original nems-nmmb.
pblend1	79.0	The nems-nmmb vertical coordinate is smoothly merged with gfs below this level. Above this level,is gfs.
step_start	1.e-4	initial stepsize in minimization
diag_precon	.false.	if true do preconditioning
lrun_subdirs	.false.	logical to toggle use of subdirectires at runtime for pe specific files
emiss_bc	.false.	option to turn on emissivity bias predictor
upd_pred	1	bias update indicator for radiance bias correction; 1.0=bias correction coefficients evolve
use_reflectivity	.false.	option of using reflectivity
lnested_loops	.false.	allow for nested resolution outer/inner loops
lwrite4danl	.false.	logical to write out 4d analysis states if 4dvar or 4denvar mode
lsingleradob	.false.	logical for single radiance observation assimilation. Uses existing bufr file and rejects all radiances that don't fall within a tight threshold around oblat/oblon (SINGLEOB_TEST)
ssmis_method	1	choose method for SSMIS noise reduction 0=no smoothing 1=default
ssmis_precond	0.01	weighting factor for SSMIS preconditioning (if not using newpc4pred)
R_option	.false.	Option to use variable correlation length for lcbas based on data density - follows Hayden and Purser (1995) (twodvar_regional only)
thin4d		

C. GSI Namelist: Name, Default Value, Explanation

Variable name	Default value	Description
GRIDOPTS		Grid setup variables, including regional specific variables
jcap	62	spectral resolution of the analysis
jcap_b	62	spectral resolution of background (model guess field)
nsig	42	number of sigma levels
nlat	96	number of latitudes
nlon	384	number of longitudes
hybrid	logical	hybrid data file flag true=hybrid
nlat_regional	0	Number of y grid point in whole regional domain
nlon_regional	0	Number of x grid point in whole regional domain
diagnostic_reg	.false.	logical for regional debugging
update_regsfc	.false.	logical to write out updated surface fields to the regional analysis file (default = false)
netcdf	.false.	if true, then wrf files are in netcdf format, otherwise wrf files are in binary format.
regional	.false.	logical for regional GSI run
wrf_nmm_regional	.false.	logical for input from WRF NMM
nems_nmmb_regional	.false.	logical for input from NEMS NMMB
wrf_mass_regional	.false.	logical for input from WRF MASS-CORE (ARW)
twodvar_regional	.false.	logical for regional 2d-var analysis
filled_grid	.false.	logical to fill in points on WRF-NMM E-grid
half_grid	.false.	logical to use every other row of WRF-NMM E-Grid
nvege_type	24	number of types of vegetation; old=24, IGBP=20
nlayers(100)	1	number of sub-layers to break indicated model layer into prior to calling radiative transfer model
cmaq_regional	.false.	Background input is from CMAQ model
nmmb_reference_grid	H	= 'H', then analysis grid covers H grid domain = 'V', then analysis grid covers V grid domain
grid_ratio_nmmb	sqrt(2)	ratio of analysis grid to nmmb model grid in nmmb model grid units.
grid_ratio_wrfmass	1.0	ratio of analysis grid to wrf mass grid in wrf grid units
jcap_gfs		spectral truncation used to transform high wavenumber spectral coefficients to a coarser resolution grid, when use_gfs_ozone = .true. or use_gfs_stratosphere = .true.
jcap_cut		
use_sp_eqspac	.false.	if .true., then ensemble grid is equal spaced, staggered 1/2 grid unit off poles. if .false., then gaussian grid assumed for ensemble (global only)

C. GSI Namelist: Name, Default Value, Explanation

Variable name	Default value	Description
BKGERR		Background error related variables
vs	1/1.5	scale factor for vertical correlation lengths for background error
nhscrf	3	number of horizontal scales for recursive filter
hzscl(3)	1, 1, 1	scale factor for horizontal smoothing, n=1,number of scales (3 for now) specifies factor by which to reduce horizontal scales (i.e. 2 would then apply 1/2 of the horizontal scale)
hswgt(3)	1/3, 1/3, 1/3	empirical weights to apply to each horizontal scale
norh	2	order of interpolation in smoothing
ndeg	4	degree of smoothing in recursive filters
noq	3	1/4 of accuracy in compact finite differencing
bw	0	factor in background error calculation
norsp	0	order of interpolation for smooth polar cascade routine default is norsp=0, in which case norh is used with original polar cascade interpolation (global only).
fstat	.false.	logical to separate f from balance projection
pert_berr	.false.	logical to turn on random inflation/deflation of background error tuning parameters
pert_berr_fct	0	factor for increasing/decreasing berror parameters, this is multiplied by random number
bkgv_flowdep	.false.	flag to turn on flow dependence to background error variances
bkgv_rewgtfct	0	factor used to perform flow dependent reweighting of error variances
bkgv_write	.false.	flag to turn on=.true. /off=.false. generation of binary file with reweighted variances
fpsproj	.true.	controls full nsig projection to surface pressure
fut2ps		controls the projection from unbalance T to surface pressure
adjustozvar		adjusts ozone variances in the stratosphere based on guess field
cwcoveqqcov		sets cw Bcov to be the same as B-cov(q) (presently glb default)

C. GSI Namelist: Name, Default Value, Explanation

Variable name	Default value	Description
ANBKGERR		Anisotropic background error related variables
anisotropic	.false.	if true, then use anisotropic background error covariance
ancovmdl	0	covariance model settings - 0: pt-based, 1: ensemble based
triad4	.true.	for 2d variables, if true, use blended triad algorithm
ifilt_ord	4	filter order for anisotropic filters
npass	1	2 \sqrt{N} npass = number of factors in background error
normal	200	number of random vectors to use for filter normalization (if < 0 then slightly slower, but results independent of number of processors)
binom	.true.	if true, weight correlation lengths of factors using binomial distribution, with shortest scales on outside, longest scales on inside. This can help to produce smoother correlations in the presence of strong anisotropy
ngauss	3	number of Gaussians to add together in each factor
rgauss	0	multipliers on reference aspect tensor for each Gaussian factor
anhswgt	1.0	empirical weights to apply to each gaussian
an_vs	1	scale factor for background error vertical scales (temporary carry over from isotropic inhomogeneous option)
grid_ratio	2.0	ratio of coarse to fine grid in fine grid units
grid_ratio_p	0	ratio of coarse to fine grid in fine grid units for polar patches
nord_f2a	4	order of interpolation for transfer operators between filter grid and analysis grid
an_flen_u	1	coupling parameter for connecting horizontal wind to background error
an_flen_t	1	coupling parameter for connecting grad(potential temperature) to background error
an_flen_z	1	coupling parameter for connecting grad(terrain) to background error
rtma_subdomain_option	.false.	if true, then call alternative code which calls recursive filter directly from subdomain mode, bypassing transition to/from horizontal slabs. This is mainly to improve efficiency for 2d rtma analysis. at the moment, this only works for twodvar_regional=.true. rtma_subdomain_option will be forced to false when twodvar_regional=.false.
lreadnorm	.false.	if true, then read normalization from fixed files
nsmooth	0	number of 1-2-1 smoothing passes before and after background error application
nsmooth_shapiro	0	number of 2nd moment preserving (shapiro) smoothing passes before and after background error application. NOTE: default for nsmooth and nsmooth_shapiro is 0. if both are > 0, then nsmooth will be forced to zero.
afact0	0.0	anistropy effect parameter, the range must be in 0.0-1.0.
covmap	.false.	if true, covariance map would be drawn

C. GSI Namelist: Name, Default Value, Explanation

Variable name	Default value	Description
JCOPTS		Constraint term in cost function (Jc)
ljcdfi	.false.	if .false., uses original formulation based on wind, temp, and ps tends when .t. uses digital filter initialization of increments (4dvar)
alphajc	10.0	parameter for digital filter
switch_on_derivatives	.false., $\hat{\Delta}\epsilon$	if true, then compute horizontal derivatives of all state variables (to be used eventually for time derivatives, dynamic constraints and observation forward models that need horizontal derivatives)
tendsflag	.false.	if true, compute time tendencies
ljcpdry	.false.	when .t. uses dry pressure constraint on increment
bamp_jcpdry	0.0	parameter for pdry_jc
eps_eer	-1.0	Errico-Ehrendofer parameter for q-term in energy norm
ljc4tlevs	.false.	when true and in 4D mode, apply any weak constraints over all time levels instead of just at a single time

Variable name	Default value	Description
STRONGOPTS		Strong dynamic constraint
reg_tlnmc_type	1	=1 for 1st version of regional strong constraint =2 for 2nd version of regional strong constraint
tlnmc_option	0	integer flag for strong constraint (various capabilities for hybrid): =0: no TLNMC =1: TLNMC for 3DVAR mode =2: TLNMC on total increment for single time level only (for 3D EnVar) or if 4D EnVar mode, TLNMC applied to increment in center of window =3: TLNMC on total increment over all time levels (if in 4D EnVar mode) =4: TLNMC on static contribution to increment ONLY for any EnVar mode
nstrong	0	if > 0, then number of iterations of implicit normal mode initialization to apply for each inner loop iteration
period_max	1000000.0	cutoff period for gravity waves included in implicit normal mode initialization (units = hours)
period_width	1.0	defines width of transition zone from included to excluded gravity waves
nmodes_keep	0	number of vertical modes to use in implicit normal mode initialization
baldiag_full	.false.	flag to toggle balance diagnostics for the full fields
baldiag_inc	.false.	flag to toggle balance diagnostics for the analysis increment

C. GSI Namelist: Name, Default Value, Explanation

Variable name	Default value	Description
OBSQC		Observation quality control variables Parameters used for gross error checks are set in file con- vinfo (ermin, ermax, ratio) Parameters below used for non- linear (variational) quality control
dfact	0	factor for duplicate observation at same location for conven- tional data
dfact1	3.0	time factor for duplicate observation at same location for con- ventional data
erradar_inflate	1	radar error inflation factor
tdrerr_inflate	.false.	logical for tdr obs error inflation
tdrgross_fact	1	factor applied to tdr gross error
oberrflg	.false.	logical for reading in new observation error table (if set to true)
vadfile	'none'	character(10) variable holding name of VAD wind bufr file
noiqc	.false.	logical flag to bypass OI QC (if set to true)
c_varqc	1	constant number to control variance qc turning on speed
blacklst	.false.	logical for reading in raob blacklist (if set to true)
use_poq7	.false.	Logical to toggle accept (.true.) or reject (.false.) SBUV/2 ozone observations flagged with profile ozone quality mark
hilbert_curve	.false.	option for hilbert-curve based cross-validation. works only with twodvar_regional=.true.
tcp_refps	1000.0	reference pressure for tcps oberr calculation (mb)
tcp_width	50.0	parameter for tcps oberr inflation (width, mb)
tcp_ermin	0.75	parameter for tcps oberr inflation (minimum oberr, mb)
tcp_erman	5.0	parameter for tcps oberr inflation (maximum oberr, mb)
qc_noirjaco3	.false.	controls whether to use O3 Jac from IR instruments
qc_noirjaco3_pole	.false.	controls wheter to use O3 Jac from IR instruments near poles
qc_satwnds	.true.	allow bypass sat-winds qc normally removing lots of mid-tropo obs
njqc		
vqc		
aircraft_t_bc_pof	.false.	logical for aircraft temperature bias correction, pof is used for predictor
aircraft_t_bc	.false.	logical for aircraft temperature bias correction
aircraft_t_bc_ext	.false.	logical for reading aircraft temperature bias correction from external file
buddycheck_t	.false.	When true, run buddy check algorithm on temperature obser- vations
buddydiag_save	.false.	When true, output files containing buddy check QC info for all obs run through the buddy check
biaspredt	1	berror var for temperature bias correction coefficients
upd_aircraft	.true.	indicator if update bias at 06Z & 18Z
cleanup_tail	.false.	logical to remove tail number no longer used

C. GSI Namelist: Name, Default Value, Explanation

Variable name	Default value	Description
OBS_INPUT		Controls input data
dfile	' '	input observation file name
dtype	' '	observation type
dplat	' '	satellite (platform) id (for satellite data)
dsis	' '	sensor/instrument/satellite flag from satinfo files
dthin	' '	satellite group
dval	' '	relative value of each profile within group relative weight for observation = dval/sum(dval) within grid box
dmesh(max(dthin))		thinning mesh for each group mesh size (km) for radiance thinning grid (used in satthin)
dsfcalc	' '	specifies method to determine surface fields within a FOV. when equal to one, integrate model fields over FOV. when not one, bilinearly interpolate model fields to FOV center.
time_window_max	3	upper limit on time window for all input data
ext_sonde	.false.	logical for extended forward model on sonde data
l_foreaft_thin	.false.	separate TDR fore/aft scan for thinning

Variable name	Default value	Description
SINGLEOB_TEST		Single observation test case setup
maginnov	1	magnitude of innovation for one observation
magoberr	1	magnitude of observational error
oneob_type	' '	observation type (t, u, v, etc.)
oblat	0	observation latitude
oblon	0	observation longitude
obpres	1000.0	observation pressure (hPa)
obdattim	2000010100	observation date (YYYYMMDDHH)
obhourset	0	observation delta time from analysis time
pctswitch	.false.	if .true. innovation & oberr are relative (%) of background value (level ozone only)
obchan	0	if > 0, selects the channel number. If <= zero, it will use all channels that pass qc in setuprad.

Variable name	Default value	Description
SUPEROB_RADAR		Level 2 bufr file to radar wind superobs
del_azimuth	5.0	azimuth range for superob box (default 5 degrees)
del_elev	0.25	elevation angle range for superob box (default .05 degrees)
del_range	5000.0	radial range for superob box (default 5 km)
del_time	0.5	1/2 time range for superob box (default .5 hours)
elev_angle_max	5.0	max elevation angle (default of 5 deg) minnum 50 minimum number of samples needed to make a superob
range_max	100000.0	max radial range in meters to use in constructing superobs (default 100km)
l2superob_only	.false.	if true, then process level 2 data creating superobs, then quit. (added for easier retrospective testing, since level 2 bufr files are very large and hard to work with)

Variable name	Default value	Description
LAG_DATA 		Lagrangian data assimilation related variables
lag_accur	1.0e-6	Accuracy used to decide whether or not a balloon is on the grid
infile_lag	inistate_lag.dat	File containing the initial position of the balloon
lag_stepduration	900.0	Duration of one time step for the propagation model
lag_nmax_bal	1000	Maximum number of balloons at starting time
lag_vorcore_stderr_a	2.0e3	Observation error for vorcore balloon
lag_vorcore_stderr_b	0.0	error = b + a*timestep(in hours)

C. GSI Namelist: Name, Default Value, Explanation

Variable name	Default value	Description
HYBRID_ENSEMBLE		Parameters for use with hybrid ensemble option
l_hyb_ens	.false.	if true, then turn on hybrid ensemble option
uv_hyb_ens	.false.	if true, then ensemble perturbation wind variables are u,v, otherwise, ensemble perturbation wind variables are stream, pot. Functions.
q_hyb_ens	.false.	if true, then use specific humidity ensemble perturbations, otherwise, use relative humidity
aniso_a_en	.false.	if true, then use anisotropic localization of hybrid ensemble control variable a_en.
generate_ens	.true.	if true, then generate internal ensemble based on existing background error
n_ens	0	number of ensemble members.
nlon_ens	0	number of longitudes on ensemble grid (may be different from analysis grid nlon)
nlat_ens	0	number of latitudes on ensemble grid (may be different from analysis grid nlat)
jcap_ens	0	for global spectral model, spectral truncation
pseudo_hyb_ens	.false.	if true, turn on pseudo ensemble hybrid for HWRF
merge_two_grid_enspers	.false.	if true, merge ensemble perturbations from two forecast domains to analysis domain (one way to deal with hybrid DA for HWRF moving nest)
regional_ensemble_option	0	integer, used to select type of ensemble to read in for regional application. Currently takes values from 1 to 4 =1: use GEFS internally interpolated to ensemble grid. =2: ensembles are WRF NMM format =3: ensembles are ARW netcdf format. =4: ensembles are NEMS NMMB format.
full_ensemble	.false.	if true, first ensemble perturbation on first guess instead of on ens mean
betaflg	.false.	if true, use vertical weighting on beta1_inv and beta2_inv, for regional
coef_bw	0.9	fraction of weight given to the vertical boundaries when betaflg is true
pwgtflg	.false.	if true, use vertical integration function on ensemble contribution of Psfc
jcap_ens_test	0	for global spectral model, test spectral truncation (to test dual resolution)
beta1_inv	1	1/beta1, the default weight given to static background error covariance if (.not. readin_beta) 0 <= beta1_inv <= 1, tuned for optimal performance =1, then ensemble information turned off =0, then static background turned off the weights are applied per vertical level such that : betas_inv(:) = beta1_inv , vertically varying weights given to static B ; betae_inv(:) = 1 - beta1_inv , vertically varying weights given ensemble derived covariance. If (readin_beta) then betas_inv and betae_inv are read from a file and beta1_inv is not used.

C. GSI Namelist: Name, Default Value, Explanation

Variable name	Default value	Description
HYBRID_ENSEMBLE		Parameters for use with hybrid ensemble option
s_ens_h	2828	homogeneous isotropic horizontal ensemble localization scale (km)
s_ens_v	30	vertical localization scale (grid units for now) s_ens_h, s_ens_v, and betal_inv are tunable parameters.
use_gfs_ens	.true.	controls use of global ensemble: .t. use GFS (default); .f. uses user-defined ens
readin_localization	.false.	flag to read (.true.)external localization information file
readin_beta	.false.	flag to read (.true.) the vertically varying beta parameters betas_inv and betae_inv from a file.
eqspace_ensgrid	.false.	if .true., then ensemble grid is equal spaced, staggered 1/2 grid unit off ploe. if .false., then gaussian grid assumed for ensemble (global only)
use_localization_grid	.false.	if true, then use extra lower res gaussian grid for horizontal localization (global runs only--allows possiblity for non-gaussian ensemble grid)
grid_ratio_ens	1	for regional runs, ratio of ensemble grid resolution to analysis grid resolution default value = 1 (dual resolution off)
oz_univ_static	.false.	if true, decouple ozone from other variables and defaults to static B (ozone only)
write_ens_sprd	.false.	writing global ensemble spread in byte addressable format for plotting with grads
enspreproc	.false.	flag to read(.true.) pre-processed ensemble data already
i_en_perts_io	0	flag to read in ensemble perturbations in ensemble grid. This is to speed up RAP/HRRR hybrid runs because the same ensemble perturbations are used in 6 cycles =0: No ensemble perturbations IO (default) =2: skip get_gefs_for_regional and read in ensemble perturbations from saved files.
l_ens_in_diff_time	.false.	if use ensembles that are available at different time from analysis time. =false: only ensembles available at analysis time can be used for hybrid. (default) =true: ensembles available time can be different from analysis time in hybrid analysis
ensemble_path		

C. GSI Namelist: Name, Default Value, Explanation

Variable name	Default value	Description
rapidrefresh_cldsrf		Options for cloud analysis and surface enhancement for RR application
dfi_radar_latent_heat_time_period	30.0	DFI forward integration window in minutes
metar_impact_radius	10.0	metar cloud observation impact radius in grid number
metar_impact_radius_lowCloud	4.0	impact radius for METAR cloud observation that indicate low cloud base
l_gsd_terrain_match_surfTobs	.false.	if .true., GSD terrain match for surface temperature observation
l_sfcobserror_ramp_t	.false.	namelist logical for adjusting surface temperature observation error
l_sfcobserror_ramp_q	.false.	namelist logical for adjusting surface moisture observation error
l_PBL_pseudo_SurfobsT	.false.	if .true. produce pseudo-obs in PBL layer based on surface obs T
l_PBL_pseudo_SurfobsQ	.false.	if .true. produce pseudo-obs in PBL layer based on surface obs Q
l_PBL_pseudo_SurfobsUV	.false.	if .true. produce pseudo-obs in PBL layer based on surface obs UV
pblH_ration	0.75	percent of the PBL height within which to add pseudo-obs
pps_press_incr	30hPa	pressure increase for each additional pseudo-obs on top of previous level
l_gsd_limit_ocean_q	.false.	if .true. do GSD limitation of Q over ocean
l_pw_hgt_adjust	.false.	if .true. do GSD PW adjustment for model vs. obs station height
l_limit_pw_innov	.false.	if .true. do GSD limitation of PW obs
max_innov_pct	0.1	sets limit of PW ob to a percent of the background value (0-1)
l_cleanSnow_WarmTs	.false.	if .true. do GSD limitation of using retrieved snow over warn area ($T_s > r_cleanSnow_WarmTs_threshold$)
l_conserve_thetaV	.false.	if .true. conserve thetaV during moisture adjustment in cloud analysis
r_cleanSnow_WarmTs_threshold	8.0	threshold for using retrieved snow over warn area
i_conserve_thetaV_iternum	3	iteration number for conserving thetaV during moisture adjustment
l_gsd_soilTQ_nudge	.false.	if .true. do GSD GOES cloud building
l_cld_bld	.false.	if .true. do GSD soil T and Q nudging based on the lowest t analysis increment
cld_bld_hgt	1200m	sets limit below which GOES cloud building occurs
build_cloud_frac_p	0.95	sets the threshold for building clouds from satellite
clear_cloud_frac_p	0.1	sets the threshold for clearing clouds from satellite
nesdis_npts_rad	1	NESDIS cloud product impact radiu (grid points)
iclean_hydro_withRef	1	if =1, then clean hydrometeors if the grid point has no echo and maxref=0
iclean_hydro_withRef_allcol	0	if =1, then clean whole column hydrometeors if the observed max ref =0 and satellite cloud shows clean

C. GSI Namelist: Name, Default Value, Explanation

Variable name	Default value	Description
rapidrefresh_cldsrf		Options for cloud analysis and surface enhancement for RR application
l_use_2mq4b	0	background used for calculate surface moisture observation innovation =0 Use Q from the 1st model level. (default) =1 use 2m Q as part of background
i_use_2mt4b	0	background used for calculate surface temperature observation innovation =0 Use T from the 1st model level. (default) =1 use 2m T as part of background
i_gsdclदानal_type	0	options for how GSD cloud analysis should be conducted =0. no cloud analysis (default) =1. cloud analysis after var analysis =5. skip cloud analysis and NETCDF file update
i_gsdsc_uselist	0	options for how to use surface observation use or rejection list =0 . EMC method (default) =1 . GSD method
i_lightpcp	0	options for how to deal with light precipitation =0 . don't add light precipitation (default) =1 . add light precipitation in warm section
i_sfct_gross	0	if use extended threshold for surface T gross check =0 use threshold from convinfo (default) =1 for cold surface, threshold for gross check is enlarged to bring more large negative innovation into analysis.

C. GSI Namelist: Name, Default Value, Explanation

Variable name	Default value	Description
CHEM		Chemistry data assimilation
berror_chem	.false.	if berror file is supplied for chemistry
oneobtest_chem	.false.	single observation test for chemistry
maginnov_chem	30.0	if oneobtest_chem=T magnitude of innovation for chemistry
magoberr_chem	2.0	if oneobtest_chem=T magnitude of observation error for chemistry
oneob_type_chem	pm2_5	if oneobtest_chem=T type of chemical observation
oblat_chem	45.0	if oneobtest_chem=T latitude of the observation
oblon_chem	270.0	if oneobtest_chem=T longitude of the observation
obpres_chem	1000.0	if oneobtest_chem=T pressure of the observation
diag_incr	.false.	if user wishes to output to a binary file increment
elev_tolerance	500.0	for surface chemical observation sometimes elevation (elev_obs) of the measurement is available (sometimes not).
tunable_error	0.5	tuning parameter to specify representativeness error for in-situ observations
in_fname	cmaq_input.bin	name of background file for cmaq
out_fname	cmaq_output.bin	name analysis file for cmaq
incr_fname	chem_increment.bin	if diag_incr=T name of the binary dump for pm2_5
laeroana_gocart	.false.	when true, do chem analysis with wrfchem and modis
l_aoderr_table		
aod_qa_limit		
luse_deepblue		
aero_ratios		
wrf_pm2_5		

Bibliography

- [1] J. Purser, W.-S. Wu, D. F. Parrish, and N. M. Roberts. Numerical aspects of the application of recursive filters to variational statistical analysis. part i: Spatially homogeneous and isotropic gaussian covariances. *Mon. Wea. Rev.*, 131:1524–1535, 2003.
- [2] J. Purser, W.-S. Wu, D. F. Parrish, and N. M. Roberts. Numerical aspects of the application of recursive filters to variational statistical analysis. part ii: Spatially inhomogeneous and anisotropic general covariances. *Mon. Wea. Rev.*, 131:1536–1548, 2003.
- [3] H. Shao, J. Derber, X.-Y. Huang, M. Hu, K. Newman, D. Stark, M. Lueken, C. Zhou, L. Nance, Y.-H. Kuo, and B. Brown. Bridging research to operations transitions: Status and plans of community gsi. *Bulletin of the American Meteorological Society*, 2016.
- [4] W.-S. Wu, J. Purser, and D. F. Parrish. Three-dimensional variational analysis with spatially inhomogeneous covariances. *Mon. Wea. Rev.*, 130:2905–2916, 2002.