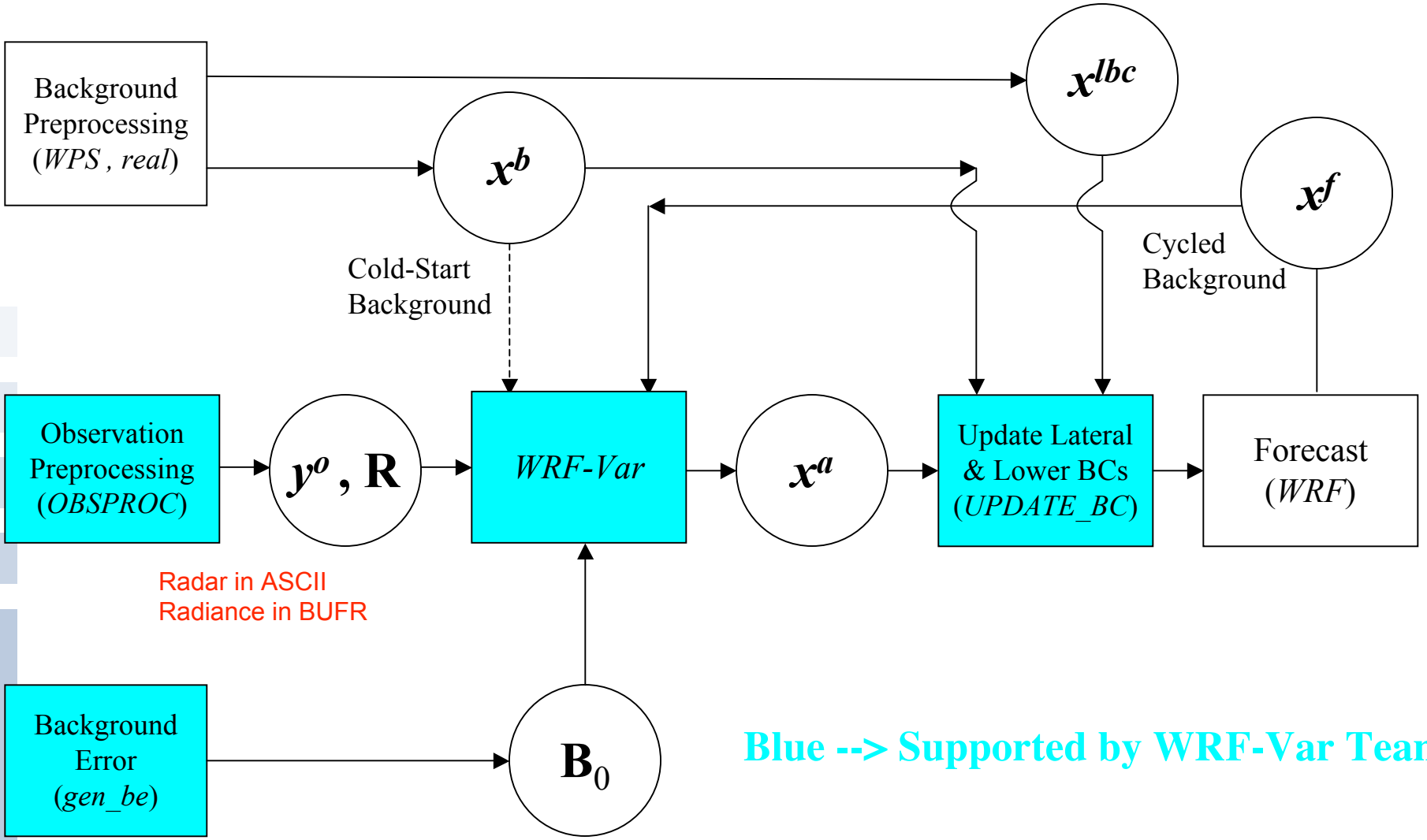
WRF-Var Setup, Run and Diagnostics

Hui Shao (huishao@ucar.edu)

Hui-Chuan Lin, Meral Demirtas, Yongrun Guo, Syed Rizvi,
Xin Zhang and Hans Huang

WRF-Var Tutorial, Feb 2-4, 2009

• WRF-Var in the WRF Modeling System



Outline

- Installing WRF-Var
- Running WRF-Var code
- Running UPDATE_BC
- WRF-Var diagnostics
  - Also check "**WRF-Var Tools and Verification Package**" talk.
- Basic runtime options (namelist)
  - Complementary to "**WRF-Var Namelists**" talk.

Installing WRF-Var

- WRFVAR-Var source code can be downloaded from http://www.mmm.ucar.edu/wrf/users/download/get_source.html

MPI Compiler

NCAR

- Please make sure your MPI compilers are MPICH2 compatible (WRF only needs MPICH1).
- If not, please download MPICH2 from

http://www.mcs.anl.gov/research/projects/mpich2/

Set environment variables:

```
>setenv MPIHOME    $your_installation_dir/mpich2
```

Libraries Required by WRF-Var

NCAR

- NetCDF:
  - http://www.unidata.ucar.edu/software/netcdf/
- BLAS (Basic Linear Algebra Subprograms)
  - http://netlib.org/blas/
- LAPACK: Linear Algebra PACKage
  - http://netlib.org/lapack/
- BUFR
  - http://www.nco.ncep.noaa.gov/sib/decoders/BUFRLIB/

Set environment variables:

```
>setenv BLAS         $your_installation_dir/blas
>setenv LAPACK   $your_installation_dir/lapack
>setenv BUFR       $your_installation_dir/bufr
>Setenv NETCDF  $your_installation_dir/netcdf
```

✓Make sure the required libraries are all compiled using the same compiler that will be used to build WRF-Var, since the libraries produced by one compiler may not be compatible with code compiled with another.

Configure and Compile WRF-Var

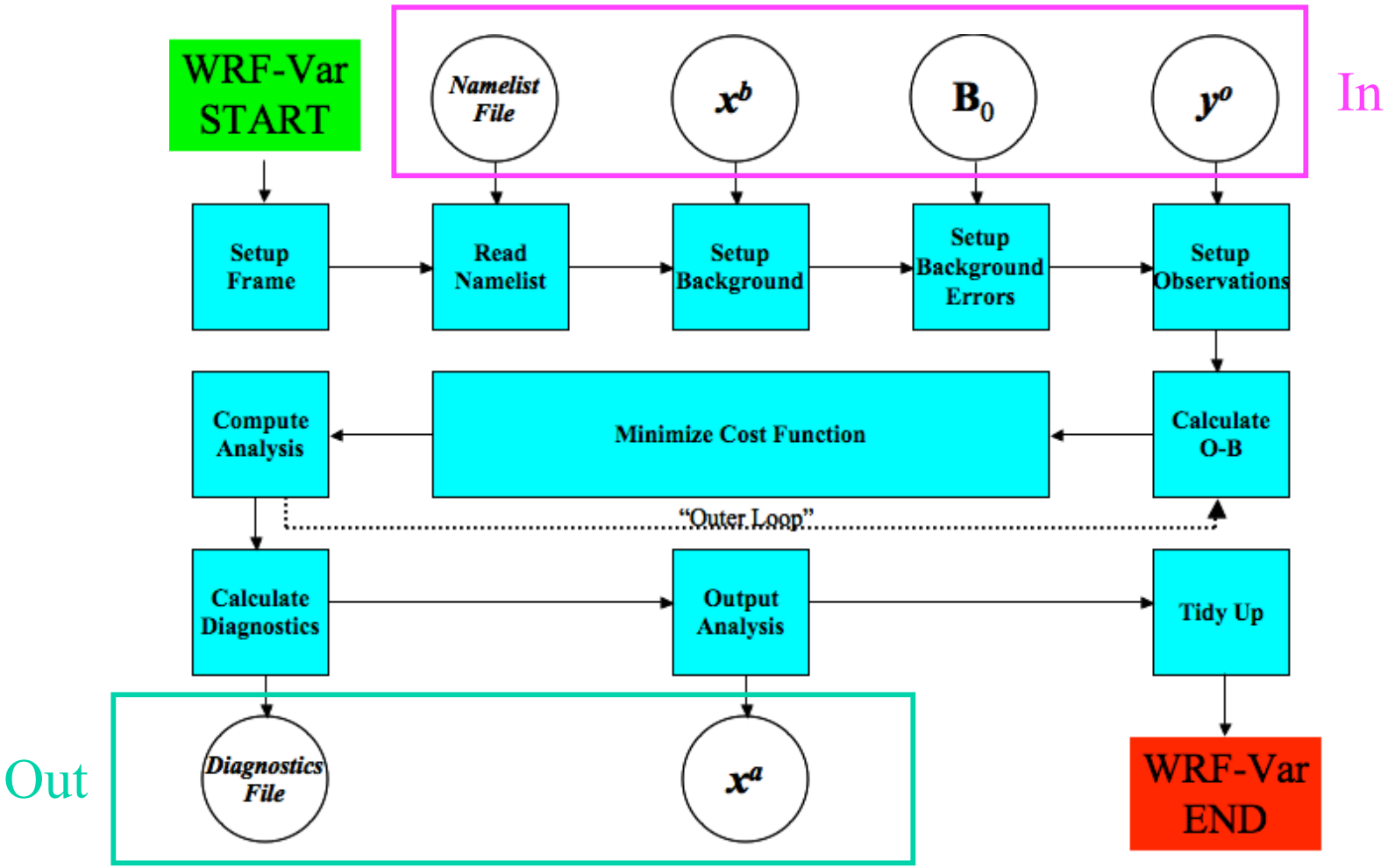
- cd $your_sourcecode_dir/WRFDA
- ./configure wrfda
  - configure.wrf is created.
- ./compile all_wrfvar
  - executables in the var/da

✓**Note: WRF compiles with -r4 option while WRFDA compiles with -r8.** For this reason, WRF and WRFDA cannot reside and be compiled under the same directory.

Running WRF-Var

NCAR

- **WRF-Var Code Flow**



Before You Run …

- Check WRF-Var executable has been created appropriately:
  - WRFDA/var/da/da_wrfvar.exe

- Get input files:
  - The following test data WRFV3-Var-testdata.tar.gz can be downloaded from http://www.mmm.ucar.edu/wrf/users/download/get_sources.htm.

  - Extract the test data into your local data directory, e.g., "*your_choice_of_dat_dir*".

  - Set up your environment variable $DAT_DIR:
    > Setenv DAT_DIR *your_choice_of_dat_dir*

Before You Run …

- Check input files:
  - Background (x^b): $DAT_DIR/rc/2007010200/wrfinput_d01
    - NETCDF format.
    - For cold-start mode, x^b is generated by WRF *real*.
    - For cycling mode, x^b is generated by WRF from previous cycle (e.g., 6hr forecast).
  - Background Error Statistics: $DAT_DIR/be/be.dat
    - Generated by *gen_be*.
    - Please refer to "**WRF-Var Background Error Estimations**" talk.
  - Observations (y^o) : $DAT_DIR/ob/2007010200/obs_gts_2007-01-02+00:00:00.3DVAR (GTS data only)
    - ASCII format.
    - Generated by OBSPROC from ob.little_r included in the tar file of the test data.
    - Please refer to "**Radar Data**" and "**Satellite Data**" talks for assimilations of radar and radiance data.
- Prepare namelists for runtime options:
  - WRFDA/var/test/namelist.input (example)

Working Directory - Input

- Create a working directory, for example, "*your_choice_of_ working_dir*".

- Go into the working directory:

  > cd *your_choice_of_working_dir*

- Prepare the input files for running WRF-Var:

  > ln -sf WRFDA/var/da/da_wrfvar.exe **./da_wrfvar.exe**

  > ln -sf WRFDA/run/LANDUSE.TBL **./LANDUSE.TBL**

  > ln -sf $DAT_DIR/rc/2007010200/wrfinput_d01 **./fg**

  > ln -sf $DAT_DIR/be/be.dat **./be.dat**

  > ln -sf $DAT_DIR/ob/obs_gts_2007-01-02_00:00:00.3DVAR **./ob.ascii**

  > cp WRFDA/var/test/namelist.input **./namelist.input**

Running WRF-Var

> **da_wrfvar.exe** >&! wrfda.log (or your own log file name)

If running in distributed-memory mode, you need to set up the computer resources (e.g., processor numbers, memory, wallclock…) based on the platform you are using. The log file names would be rsl.out.0000, rsl.out.0001,…

Working Directory - Output

In *your_choice_of_working_dir*, you should at least have the following files after running WRF-Var successfully:

-rw-r--r-- 1 username 1985 Jun 18 13:17 cost_fn (Cost function)

-rw-r--r-- 1 username 1745 Jun 18 13:17 grad_fn (Gradient of cost function)

-rw-r--r-- 1 username 9048641 Jun 18 13:17 gts_omb_oma (O, O-A information, etc)

-rw-r--r-- 1 username 276658 Jun 18 13:17 namelist.output (Complete namelist)

-rw-r--r-- 1 username 22730 Jun 18 13:17 statistics (Averaged O-B & O-A information)

-rw-r--r-- 1 username 3651560 Jun 18 13:17 **wrfvar_output (Analysis x^a)**

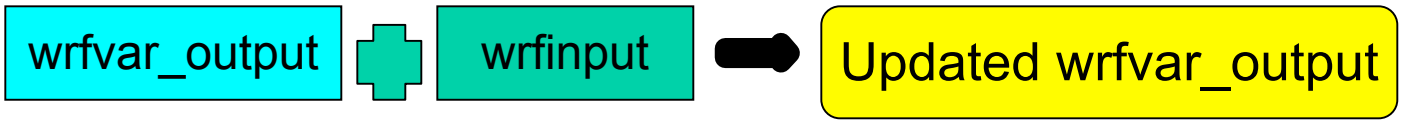
O: Observation
A: Analysis
B: Background (first-guess)

“update_bc” basic

Input and output for update_bc

- Update the ***lateral boundary*** condition:

- Update ***the low boundary*** condition:



- Input to update_bc:

  wrfvar_output from **WRF-Var** assimilation

  wrfbdy, wrfinput from **WPS** at analysis time.

- Output from update_bc:

  updated wrfbdy and updated wrfvar_output

  are used to run **WRF** model.

Applications of update_bc

NCAR

- Cold-start run

  only lateral boundary update needed

- Warm-start (cycling) run

  both lateral and low boundaries update needed

- Coarse and fine domains in nested model run

➢ for coarse domain (domain_id = 1),

  both lateral and low boundaries updated

➢ for fine mesh domains (domain_id > 1),

  low boundary updated only

Running UPDATE_BC

Steps to Run UPDATE_BC

- Check UPDATE_BC executable has been created appropriately:
  - WRFDA/var/da/da_update_bc.exe

- Go into the working directory:
  > cd *your_choice_of_working_dir*

- Prepare the namelist for UPDATE_BC: param.in

```
&control_param
 wrfvar_output_file = 'wrfvar_output'     - Analysis generated from WRF-Var
 wrf_bdy_file       = 'wrfbdy_d01'        - BC generated from WPS and WRF real
 wrf_input          = 'real_output'       - IC generated from WPS and WRF real

 cycling = .true.
 debug   = .true.
 low_bdy_only = .false.
 update_lsm = .false.
/
```

Steps to Run UPDATE_BC

- Check UPDATE_BC executable has been created appropriately:
  - WRFDA/var/da/da_update_bc.exe
- Go into the working directory:

  > cd *your_choice_of_working_dir*
- Prepare the input files for running WRF-Var:

  > ln -sf WRFDA/var/da/da_update_bc.exe **./da_update_bc.exe**

  > cp -p $DAT_DIR/rc/2007010200/wrfbdy_d01 **./wrfbdy_d01**

  > cp -p $DAT_DIR/rc/2007010200/wrfinput_d01 **./real_output**

  > cp WRFDA/var/test/param.in **./param.in** (or define your own file)

  And **wrfvar_output**
- Run UPDATE_BC:

  > da_update_bc.exe > &! da_update_bc.log
- Check output: **wrfvar_output**, **wrfbdy_d01** (overwrite the original ones!)

Steps to Run UPDATE_BC

- Check UPDATE_BC executable has been created appropriately:
  - WRFDA/var/da/da_update_bc.exe

- Go into the working directory:
  > cd *your_choice_of_working_dir*

- Prepare the namelist for UPDATE_BC: param.in

```
&control_param
 wrfvar_output_file = 'wrfvar_output'      - Analysis generated from WRF-Var
 wrf_bdy_file       = 'wrfbdy_d01'         - BC generated from WPS and WRF real
 wrf_input          = 'real_output'        - IC generated from WPS and WRF real

 cycling = .true.
 debug   = .true.
 low_bdy_only = .false.
 update_lsm = .false.
/
```

Steps to Run UPDATE_BC

- Check UPDATE_BC executable has been created appropriately:
  - WRFDA/var/da/da_update_bc.exe
- Go into the working directory:

  > cd *your_choice_of_working_dir*
- Prepare the input files for running WRF-Var:

  > ln -sf WRFDA/var/da/da_update_bc.exe **./da_update_bc.exe**

  > cp -p $DAT_DIR/rc/2007010200/wrfbdy_d01 **./wrfbdy_d01**

  > cp -p $DAT_DIR/rc/2007010200/wrfinput_d01 **./real_output**

  > cp WRFDA/var/test/param.in **./param.in** (or define your own file)

  And **wrfvar_output**
- Run UPDATE_BC:

  > da_update_bc.exe > &! da_update_bc.log
- Check output: **wrfvar_output**, **wrfbdy_d01** (overwrite the original ones!)

WRF-Var Diagnostics

ASCII output files in the WRF-Var working directory:

- wrfda.log or rsl.out.0000
- filtered_obs (analysis_type="QC-OBS")
- namelist.output
- check_max_iv
- cost_fn
- grad_fn
- gts_omb_oma
- statistics
- jo
- unpert_obs
- pert_obs (omb_add_noise=true)

wrfda.log

• Very important information about your WRF-Var run, including observation summary, values of cost function and its gradient, etc. If a WRF-Var run succeeded, the message

SUCCESS COMPLETE WRFVAR
**** WRF-Var completed successfully ****

would be written to the end of the log file.

filtered_obs

• Similar to ob.ascii (observation input file of WRF-Var) but with the observations filtered by WRF-Var using the following WRF-Var namelist option:

analysis_type = QC-OBS

namelist.output

- When WRF-Var is run, a namelist.output file will be produced with all values of namelist variables (default or/and from namelist.input).

namelist.input

```
&wrfvar1
write_increments=true,
var4d=false,
multi_inc=0,
global=false,
/
&wrfvar2
/
&wrfvar3
ob_format=2,
num_fgat_time=1,
/
&wrfvar4
/
```

namelist.output

```
&WRFVAR1
WRITE_INCREMENTS=T, WRFVAR_MEM_MODEL=0, VAR4D=F, MULTI_INC=0,
VAR4D_COUPLING=2, GLOBAL=F, PRINT_DETAI
L_AIREP=F, PRINT_DETAIL_RADAR=F, PRINT_DETAIL_RAD=F,
PRINT_DETAIL_XA=F, PRINT_DETAIL_XB=F, PRINT_DETAI
L_OBS=F, PRINT_DETAIL_F_OBS=F, PRINT_DETAIL_MAP=F,
PRINT_DETAIL_GRAD=F, PRINT_DETAIL_REGRESSION=F, PRI
NT_DETAIL_SPECTRAL=F, PRINT_DETAIL_TESTING=F,
PRINT_DETAIL_PARALLEL=F, PRINT_DETAIL_BE=F,
PRINT_DETAIL_TIMING=F, CHECK_MAX_IV_PRINT=T
 /
 &WRFVAR2
 ANALYSIS_ACCU=900, CALC_W_INCREMENT=F, DT_CLOUD_MODEL=F,
WRITE_QCW=F, WRITE_QRN=F, WRITE_QCI=F, WRITE_QSN=F, WRITE_QGR=F,
WRITE_FILTERED_OBS=F
 /
 &WRFVAR3
 FG_FORMAT=1, OB_FORMAT=2, NUM_FGAT_TIME=1
 /
 &WRFVAR4
 USE_SYNOPOBS=T, USE_SHIPSOBS=T, USE_METAROBS=T,
USE_SOUNDOBS=T, USE_MTGIRSOBS=T, USE_PILOTOBS=T,
```

check_max_iv

• Contains information about observations that fail check_max_iv check.

```
For outer iteration     1, Total Rejections for Synop follows:

Number of failed u-wind observations:        25 on    230
Number of failed v-wind observations:        30 on    230
Number of failed pressure observations:       5 on    230
Number of failed temperature observations:    0 on    230
Number of failed mixing ratio observations:   1 on    230
Finally Total Synop rejections     61 on   1150

Err_max failed:ID=47843FM-12 SYNOP Ix=  204 Ixf= 30 Err_max ratio = 1.3 for V     inv, error: 0.708141E+01 0.110000E+01
Err_max < 0 ==>       0.0 -4444440.0       5.0  for V     OBS ID: 98752FM-12 SYNOP  LA/LON/ELV:     9.93    125.51     46.00
```

• Generated with the following namelist options:

&wrfvar1
  CHECK_MAX_IV_PRINT=true (default)
&wrfvar5
  CHECK_MAX_IV=true (default)

• Observations are rejected if the innovation (O-B) values are larger than certain maximum errors (max_error). Currently the max_error values are hard-wired in var/da/da_control/da_control.f90.

```
real, parameter :: max_error_t           = 5.0, &
                   max_error_uv          = 5.0, &
                   max_error_pw          = 5.0, &
                   max_error_ref         = 5.0, &
                   max_error_rh          = 5.0, &
                   max_error_q           = 5.0, &
                   max_error_p           = 5.0, &
                   max_error_tb          = 5.0, &
                   max_error_thickness   = 5.0, &
                   max_error_rv          = 5.0, &
                   max_error_rf          = 5.0, &
                   max_error_buv         = 500.0, &
                   max_error_bt          = 500.0, &
                   max_error_bq          = 500.0, &
                   max_error_slp         = 500.0
```

cost_fn and grad_fn

• Contain values of cost function and its gradient at each iteration.
E.g., calculate_cg_cost_fn=.false. (default):

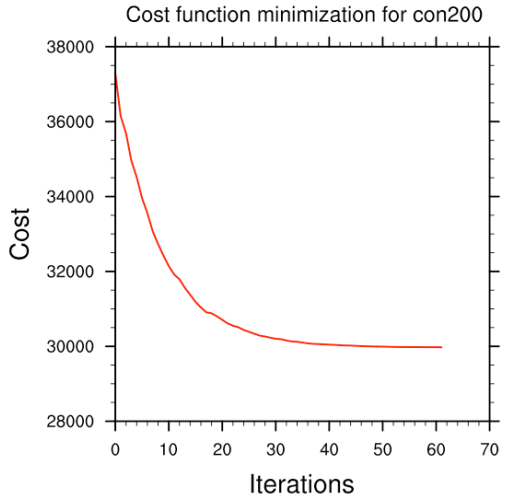
Outer Iter	EPS	Inner Iter	J	Jb	Jo	Jc	Je	Jp
1	0.100E-01	0	37293.267	0.000	37293.267	0.000	0.000	0.000
1	0.100E-01	61	29974.427	1093.616	28880.811	0.000	0.000	0.000

Outer Iter	EPS	Inner Iter	G	Gb	Go	Ge	Gp
1	0.100E-01	0	1619.534	0.000	1619.534	0.000	0.000
1	0.100E-01	61	12.931	46.768	48.511	0.000	0.000

b: background term
o: observation term
c: JcDFI term
e: alpha term
p: radiance variational bias correction term

• With calculate_cg_cost_fn=.true., the cost function and its gradient at each iteration can be written into **cost_fn** and **grad_fn**.

• NCL script example to plot these two files is var/graphics/ncl/plot_cost_grad_fn.ncl.



gts_omb_oma

• Contain complete point-by-point, detailed observation information.

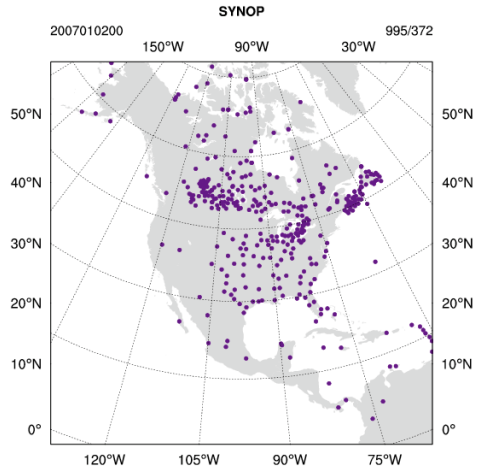
```
synop     995
     1
     1      176556    21.51  -104.90    89973.8836463       3.3147587       1.2193668       2       1.1000000       0.1849281      -1.5412909
  -1.4225501       2       1.1000000      -1.6862257     295.5511624       2.5999150       2       2.0000000       1.3689324    89973.8836463
-273.5464584       2     100.0000000    -236.6028635       0.0134689       0.0048657       0       0.0036749       0.0050584
```

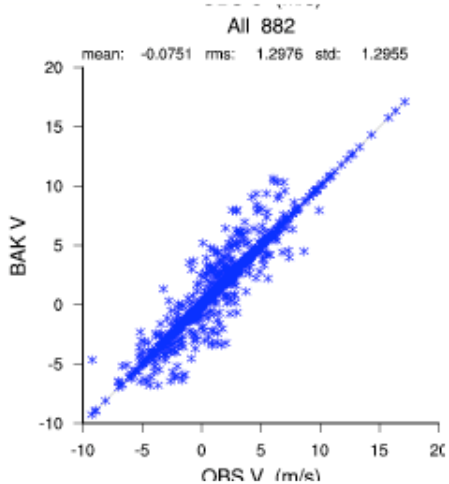
```
obs_type  number_of_obs
number_of_levels
level_index, station_id, lat, lon, pressure, (O, O-B, qcflag, Obs_err, O-A) for measured quantities for the obs_type
```

• Measured quantities for each observation type vary:

Synop: u, v, t, p, q	Polaramv: u, v	Airep: u, v, t	Profiler: u, v
Metar: u, v, t, p, q	Gpspw: tpw	Pilot: u, v	Buoy: u, v, t, p, q
Ship: u, v, t, p, q	Sound: u, v, t, q	Satem: thickness	Airsr: t, q
Geoamv: u, v	Sonde_sfc: u, v, t, p, q	Qscat: u, v	Gpsref: ref

• A NCL script example is var/graphics/ncl/plotobs.ncl.





statistics

- Contains domain-wise O-B and O-A information:

Diagnostics of OI for synop

var	u (m/s)	n	k	v (m/s)	n	k	t (K)	n	k	p (Pa)	n	k	q (kg/kg)	n	k
Number:	331			332			355			330			361		
Minimum(n,k):	-5.4017	363	0	-5.4086	878	0	-9.7206	592	0	-390.7893	931	0	-0.4461E-02	719	0
Maximum(n,k):	5.0466	886	0	5.2878	630	0	7.7302	421	0	471.9343	944	0	0.5408E-02	787	0
Average :	-0.8471			-0.1995			-1.1171			20.4177			-0.2525E-03		
RMSE :	2.3023			2.1150			3.1978			116.1518			0.8045E-03		

Diagnostics of AO for synop

var	u (m/s)	n	k	v (m/s)	n	k	t (K)	n	k	p (Pa)	n	k	q (kg/kg)	n	k
Number:	331			332			355			330			361		
Minimum(n,k):	-4.2496	172	0	-5.0463	683	0	-8.9005	583	0	-472.9290	931	0	-0.4152E-02	719	0
Maximum(n,k):	5.5540	886	0	5.7990	630	0	8.8192	421	0	392.4096	944	0	0.5058E-02	1	0
Average :	-0.0847			-0.0376			-0.4283			1.1709			0.1625E-04		
RMSE :	1.8650			1.8093			2.1990			101.3816			0.5958E-03		

Minimum of gridded analysis increments

Lvl	u	i	j	v	i	j	t	i	j	p	i	j	q	i	j
1	-1.8915	17	32	-1.9965	36	24	-5.2526	20	35	-314.7470	44	1	-0.1451E-02	18	32
2	-1.9476	16	32	-2.0070	36	24	-3.0142	21	36	-311.2885	44	1	-0.1438E-02	18	33

Maximum of gridded analysis increments

Lvl	u	i	j	v	i	j	t	i	j	p	i	j	q	i	j
1	1.3750	41	8	1.5739	28	12	3.2994	24	20	197.8351	28	2	0.1401E-02	39	8
2	1.4844	40	8	1.6180	28	13	1.7471	7	20	195.5165	28	2	0.1591E-02	39	8

Mean of gridded analysis increments

Lvl	u	v	t	p	q
1	-0.0327	0.0632	-0.1477	17.4414	-0.1047E-03
2	-0.0031	0.0736	0.0116	17.2543	-0.8066E-04

RMSE of gridded analysis increments

Lvl	u	v	t	p	q
1	0.7546	0.6040	1.3120	72.0441	0.4258E-03
2	0.7995	0.6483	0.9169	71.2614	0.4476E-03

jo

NCAR

• Contains cost function for each observation type:

synop	obs,	Jo(actual)	=	1007	1709	475.29555	1.00000	448.89633	1.00000	214.58090	1.00000	169.59091	1.00000	39.54654	1.00000
metar	obs,	Jo(actual)	=	2551	4996	1142.22791	1.00000	1139.04835	1.00000	450.85222	1.00000	141.48881	1.00000	127.23786	1.00000
ships	obs,	Jo(actual)	=	270	739	295.61942	1.00000	328.81980	1.00000	38.63147	1.00000	76.05158	1.00000	10.88285	1.00000
geoamv	ob,	Jo(actual)	=	18216	35619	4375.80943	1.00000	4291.11244	1.00000	0.00000	1.00000	0.00000	1.00000	0.00000	1.00000
gpspw	obs,	Jo(actual)	=	113	94	42.19891	1.00000	0.00000	1.00000	0.00000	1.00000	0.00000	1.00000	0.00000	1.00000
sound	obs,	Jo(actual)	=	122	12507	1501.01081	1.00000	1417.89485	1.00000	2934.71994	1.00000	1412.34202	1.00000	0.00000	1.00000
sonde	obs,	Jo(actual)	=	122	12507	77.96908	1.00000	70.37029	1.00000	43.28542	1.00000	45.34806	1.00000	4.58217	1.00000
airep	obs,	Jo(actual)	=	1527	4506	699.19993	1.00000	655.45784	1.00000	776.57509	1.00000	0.00000	1.00000	0.00000	1.00000
pilot	obs,	Jo(actual)	=	112	5895	2582.21854	1.00000	2434.46137	1.00000	0.00000	1.00000	0.00000	1.00000	0.00000	1.00000
satem	obs,	Jo(actual)	=	204	2079	108.15758	1.00000	0.00000	1.00000	0.00000	1.00000	0.00000	1.00000	0.00000	1.00000
buoy	obs,	Jo(actual)	=	241	400	133.21166	1.00000	104.72975	1.00000	31.86149	1.00000	38.47701	1.00000	1.04651	1.00000

• Sum of individual Jo (numbers in red boxes) equals the printout value in WRF-Var log file, e.g., rsl.out.0000:

Final value of Jo = 28880.81069

• Numbers in blue boxes are observation error factors used in WRF-Var:

Tuned obs_error = obs_error * factor

Where obs_error values are assigned by OBSPROC and factor=1 by default (use_obs_errfac=false).

WRF-Var Running Options - Namelist

What is a Namelist?

- The Fortran namelist (namelist.input) file helps the user to configure a WRF-Var run without recompiling the code.
  - Specific Fortran 90 namelist format

```
&namelistname        - start
…
/                    - end
```

- Description of WRF-Var namelist variables are given in WRF User's Guide (Chapter 6).

WRF-Var Namelist

- Default values of the namelist variables are defined by WRF-Var Registry (registry.wrfvar).
- Define namelist.input with non-default and desired variable values before running WRF-Var.
- A WRF-Var namelist file includes two parts:

```
&wrfvar1
/
&wrfvar2
/
…
&wrfvar23
/
&time_control
/
&dfi_control
/
…
&namelist_quilt
/
```

WRF-Var namelist:
Running options for WRF-Var code.

WRF namelist:
WRF-Var needs certain information from this file including domain and time setting. Please make sure this part of the namelist file is consistent with the namelist used in your WRF *real* and WRF runs.

Namelist - WRFVAR3

- Ob_format: The format of the conventional and satellite retrieval observation data going into WRF-Var.
  - 1 = BUFR (Please use this option with caution).
  - 2 = ASCII (ob.ascii): Default.
  - ✓ Both formats are supported by OBSPROC.

Namelist - WRFVAR4

- Use_obstype: Set to true to use particular observation types.
  - E.g, use_gpsrefobs=.true.: Assimilate GPS refractivity observations if any available in the data file.

Namelist - WRFVAR5

- Check_max_iv: Option for WRF-Var internal QC procedure, which is basically a maximum observation error check based on the innovations (Obs-Background).
  - .true. : default
  - .false: Use this option only if the observation data have been cleaned before going into WRF-Var.

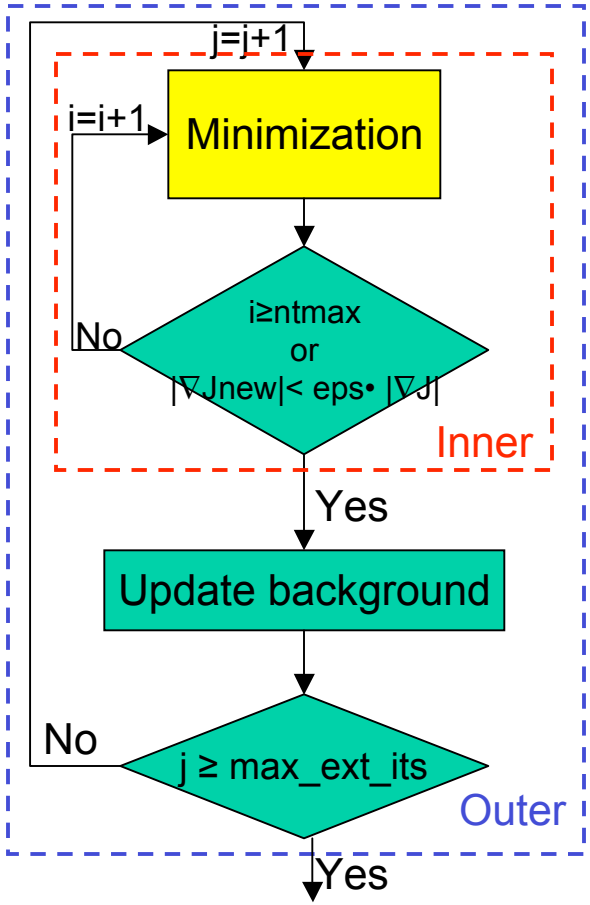
Namelist - WRFVAR6

The following namelist variables are for minimization options:

- Max_ext_its: Number of outer loops.
  - 1: Default. Only one outer loop.
  - Currently, maximum outer loop number is 10.

- Ntmax: Maximum number of iterations in an inner loop for the minimization in WRF-Var.
  - 200: Default. The minimization in the inner loop can not exceed 200.

- Eps: Value for minimization convergence criterion. It is an array with the dimension=max_ext_its.
  - 0.01(max_ext_its): The minimization is considered to converge when the norm of the cost function gradient is reduced at least 2 orders.



Namelist - WRFVAR9

The following namelist variables are for tracing:

Tracing gives additional diagnostics about program runs. It does not change results, but does slow the program down, so should be disabled in production environments.

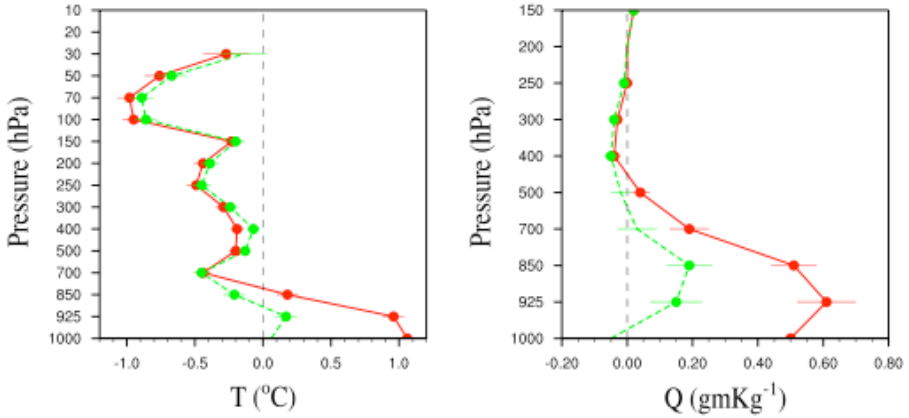
- Trace_use: .true. (default). Use tracing function in WRF-Var if true.

In your test run, please make sure to set trace_use=.false.

Namelist - WRFVAR11

- Sfc_assi_options:
  - 1 (default): The surface observations will be assimilated based on the lowest model level first guess.
  - 2: The surface observations will be assimilated based on surface similarity theory in PBL. Innovations are computed based on 10-m wind and 2-m temperature & moisture.

✓Please use this option with caution, since the results could be very sensitive.



Namelist - WRFVAR11

- Calculate_cg_cost_fn:
  - .false. : Only the initial and final cost functions are computed and output.
  - .true. : The cost functions are computed and output (cost_fn) at every iteration for diagnostic purpose.

✓ The conjugate gradient algorithm for the minimization does not require the computation of cost function at every iteration.

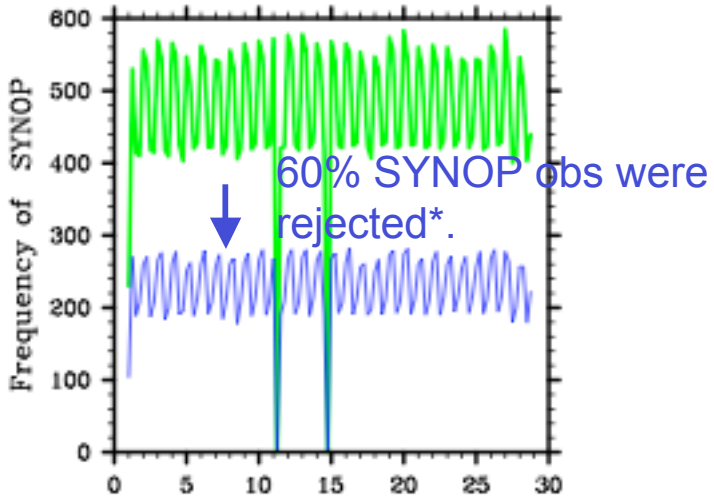
✓ The cost function gradient values will be output (grad_fn) at every iteration as well if this option is set true.

Outer Iter	EPS	Inner Iter	J	Jb	Jo	Jc	Je	Jp
1	0.100E-01	0	11251.182	0.000	11251.182	0.000	0.000	0.000
1	0.100E-01	19	8634.570	885.427	7749.143	0.000	0.000	0.000

Outer Iter	EPS	Inner Iter	J	Jb	Jo	Jc	Je	Jp
1	0.100E-01	0	11251.182	0.000	11251.182	0.000	0.000	0.000
1	0.100E-01	1	10384.156	41.768	10342.388	0.000	0.000	0.000
1	0.100E-01	2	9633.557	184.109	9449.448	0.000	0.000	0.000
1	0.100E-01	3	9245.700	327.121	8918.579	0.000	0.000	0.000
1	0.100E-01	4	9014.861	453.787	8561.075	0.000	0.000	0.000
1	0.100E-01	5	8872.989	559.714	8313.275	0.000	0.000	0.000
1	0.100E-01	6	8777.974	652.105	8125.869	0.000	0.000	0.000
1	0.100E-01	7	8720.998	721.735	7999.263	0.000	0.000	0.000
1	0.100E-01	8	8689.342	768.464	7920.878	0.000	0.000	0.000
1	0.100E-01	9	8665.605	810.136	7855.469	0.000	0.000	0.000
1	0.100E-01	10	8654.051	833.590	7820.461	0.000	0.000	0.000
1	0.100E-01	11	8646.376	851.091	7795.285	0.000	0.000	0.000
1	0.100E-01	12	8641.869	862.515	7779.355	0.000	0.000	0.000
1	0.100E-01	13	8638.219	872.853	7765.365	0.000	0.000	0.000
1	0.100E-01	14	8636.669	877.707	7758.962	0.000	0.000	0.000
1	0.100E-01	15	8635.794	880.667	7755.127	0.000	0.000	0.000
1	0.100E-01	16	8635.176	882.929	7752.247	0.000	0.000	0.000
1	0.100E-01	17	8634.861	884.169	7750.693	0.000	0.000	0.000
1	0.100E-01	18	8634.686	884.909	7749.777	0.000	0.000	0.000
1	0.100E-01	19	8634.570	885.427	7749.143	0.000	0.000	0.000

Namelist - WRFVAR17

- Analysis_type: Indicate job type of WRF-Var.
  - 3D-VAR (default): Run 3D-Var data assimilation.
  - VERIFY: Run WRF-Var verification mode (then Check_max_iv=.false. and ntmax=0 by default).
    - ✓ Please refer to "**WRF-Var Tools and Verification package**" talk.
  - QC-OBS: Run 3D-Var data assimilation and produce filtered_obs.
    - ✓ By combining with Check_max_iv=.true. and ntmax=0, you can produce a WRF-Var filtered (QCed) observation data set (filtered_obs) without running the data assimilation.
      - 1st screen/QC procedure performed by observation preprocessor (OBSPROC).
      - 2nd screen/QC procedure performed in WRF-Var.
      - Main impact of 2nd screen/QC is on surface observations*.
      - Rejection rates will reduce with higher resolution, higher-order interpolation.



* Surface observation rejection here is mostly due to surface elevation check with sfc_assim_options=1. Such a rejection may be bypassed by using sfc_assim_options=2.

Namelist - WRFVAR17

- Analysis_date: Specify the analysis time. It should be consistent with the first guess time.

NCAR

Thank you!

• UPDATE_BC Code Flow