

GSI Observation Operator

Temperature from prepbufr (setup)

setup includes forward operators for virtual, sensible temperature, T2m and PBL pseudo surface observations

General workflow:

1. surface temperature correction based on terrain (gsd_terrain_match_surfTobs.f90)
2. buddy check (for regional 2DVar): compares innovations of all obs within some radius with the mean innovation (buddycheck_mod.f90)
3. handle multiple reported data at station (choose one of the reports)
4. throw out the observations that are outside of the time window
5. bias correction for aircraft data
6. compute innovations
7. gross checks (compare innovation/obs error with some threshold)
8. (opt) generate PBL pseudo surface observations.
9. save data

QC methods from setup generally can be divided into methods that use only observation data (+ some configs):

- handle multiple reports
- observations out of time window
- also thinning in readprepbufr.f90

and methods that require state information:

- gross check
- buddy check

Note: gross checks potentially can use ensemble spread information.

```
type(t_obs_space)      :: obs
type(t_obs_operator)    :: obs_op
type(t_linear_obs_operator) :: lin_obs_op
type(t_obs_error)       :: obserr
type(obs_data)          :: hx
type(fields_at_locations) :: int_state

call obs%create(????)                                     ! reading observations

call obserr%create(t_obs_err_config, obs)                 ! initializing observation error (TODO: needs yobs on
input, ???)
call obserr%quality_control()                           ! QC that uses only observation data:
handle multiple reports; observations out of time window (3, 4)

call int_state%create(obs, geom, vars)                  ! interpolate fields to observation locations? (TODO: need
fields on input!)
call obs_op%create(t_hx_config)                         ! initializing observation operator
call lin_obs_op%create(obs_op, int_state, bias_pred, t_linhx_config) ! initializing linear observation operator
hx = obs_op%hoper(int_state, bias_pred)                ! calculate innovations; do surface temp correction and
bias correction for aircraft data (1, 5, 6)

call obserr%quality_control(hx)                         ! QC that uses state info: buddy check, gross check
(2, 7, 8?)
```

AOD from bufr (setupaod.f90)

AOD bufr created from HDF files, different HDF formats for MODIS, VIIRS. Also ABI GOES-R possible to implement

General workflow

- assign which channels to use based on a list in aeroinfo.txt,
- initialize CRTM for a given satellite and type of observation (e.g. obstype='modis_aod', call init_crtm(...)),
- read AODs at different wavelengths (channels) for a chosen satellite (MODIS Aqua+ Terra, VIIRS, namelist, only high quality data in bufr files),
- check if within window,
- calculate error based on surface type/retrieval algorithm,
- calculate layer model AODs and Jacobians using CRTM (call call_crtm(...), horizontal interpolation in call_crtm),

- reject if crtm returns error(?), select channels for processing (only 550nm now),
- accept/reject AODs depending on quality data from Deep Blue algorithm (MODIS),
- calculate innovations,
- save innovations, Jacobians, errors etc. in my_head list,
- save accepted and rejected in my_diag list,
- write diagnostics to a file (e.g. for EnKF),
- quality control largely done before creating bufr files.

Radiance (setuprad.f90)

Interface:

```
radiance obs operator interface

subroutine setuprad(lunin,mype,aivals,stats,nchanl,nreal,nobs,&          obstype,isis,is,rad_diagsave,init_pass,
last_pass)
!$$$  subprogram documentation block
!
! subprogram:      setuprad      compute rhs of oi equation for radiances
!   prgmmr: derber           org: np23           date: 1995-07-06
!
! abstract: read in data, first guess, and obtain rhs of oi equation
!           for radiances.
!
! program history log:
!
! input argument list:
!   lunin    - unit from which to read radiance (brightness temperature, tb) obs
!   mype     - mpi task id
!   nchanl   - number of channels per obs
!   nreal    - number of pieces of non-tb information per obs
!   nobs     - number of tb observations to process
!   obstype  - type of tb observation
!   isis     - sensor/instrument/satellite id ex.amsua_n15
!   is       - integer counter for number of observation types to process
!   rad_diagsave - logical to switch on diagnostic output (.false.=no output)
!   channelinfo - structure containing satellite sensor information
!
! output argument list:
!   aivals - array holding sums for various statistics as a function of obs type
!   stats  - array holding sums for various statistics as a function of channel
!
! attributes:
!   language: f90
!   machine: ibm RS/6000 SP
!
!$$$

use mpeu_util, only: die,perr,getindex
use kinds, only: r_kind,r_single,i_kind
use crtm_spccoeff, only: sc
use radinfo, only: nuchan,tlapmean,predx,cbias,ermax_rad,tzr_qc,&
npred,jpch_rad,varch,varch_cld,iuse_rad,icld_det,nusis,fbias,retrieval,b_rad,pg_rad,&
air_rad,ang_rad,adp_anglebc,angord,ssmis_precond,emiss_bc,upd_pred, &
passive_bc,ostats,rstats,newpc4pred,radjacnames,radjacindxs,nsigradjac
use gsi_nstcouplermod, only: nstinfo
use read_diag, only: get_radiag,ireal_radiag,ipchan_radiag
use guess_grids, only: sfcmod_gfs,sfcmod_mm5,comp_fact10
use m_prad, only: radheadm
use m_obsdiags, only: radhead
use obsmod, only: ianldate,ndat,mype_diaghdr,nchan_total, &
dplat,dtbdbuv_on,&
i_rad_ob_type,obsdiags,obsptr,lobsdiagsave,nobskeep,lobsdiag_allocated,&
```

```

dirname,time_offset,lwrite_predterms,lwrite_peakwt,reduce_diag
use m_obsNode, only: obsNode
use m_radNode, only: radNode, radNode_typecast
use m_obsLList, only: obsLList_appendNode
use m_obsLList, only: obsLList_tailNode
use obsmod, only: obs_diag,luse_obsdiag,dval_use
use gsi_4dvar, only: nobs_bins,hr_obsbin,14dvar
use gridmod, only: nsig,regional,get_ij
use satthin, only: super_vall
use constants, only: quarter,half,tiny_r_kind,zero,one,deg2rad,rad2deg,one_tenth, &
two,three,cg_term,wgtlim,r100,r10,r0_01,r_missing
use jfunc, only: jiter,miter,jiterstart
use sst_retrieval, only: setup_sst_retrieval,avhrr_sst_retrieval,&
finish_sst_retrieval,spline_cub
use m_dtime, only: dtime_setup, dtime_check, dtime_show
use crtmt_interface, only: init_crtm,call_crtm,destroy_crtm,sensorindex,surface, &
itime,ilon,ilat,ilzen_ang,ilazi_ang,iscan_ang,iscan_pos,iszen_ang,isazi_ang, &
ifrac_sea,ifrac_lnd,ifrac_ice,ifrac_sno,itsavg, &
izz,idomsfc,isfcr,iff10,ilone,ilate, &
isst_hires,isst_navy,idata_type,iclr_sky,itref,idtw,idtc,itz_tr
use clw_mod, only: calc_clw, ret_amsua
use qcmod, only: qc_ssmi,qc_seviri,qc_ssuv,qc_avhrr,qc_goesimg,qc_msu,qc_irsnd,qc_amsua,qc_mhs,qc_atms
use qcmod, only: igood_qc,ifail_gross_qc,ifail_interchan_qc,ifail_crtm_qc,ifail_satinfo_qc,qc_noirjaco3,
ifail_cloud_qc
use qcmod, only: qc_gmi,qc_saphir,qc_amsr2
use qcmod, only: setup_tzr_qc,ifail_scannedge_qc,ifail_outside_range
use gsi_metguess_mod, only: gsi_metguess_get
use oneobmod, only: lsingleradob,obchan,oblat,oblon,oneob_type
use radinfo, only: radinfo_adjust_jacobian,radinfo_get_rsqrtinv

implicit none

! Declare passed variables
logical ,intent(in ) :: rad_diagsave
character(10) ,intent(in ) :: obstype
character(20) ,intent(in ) :: isis
integer(i_kind) ,intent(in ) :: lunin,mype,nchanl,nreal,nobs,is
real(r_kind),dimension(40,ndat) ,intent(inout) :: aivals
real(r_kind),dimension(7,jpch_rad),intent(inout) :: stats
logical ,intent(in ) :: init_pass,last_pass ! state of "setup" processing

```

General flow:

- **Initialization:**
 - Initialize variables and constants.
 - Initialize logical flags for satellite platform
 - Determine whether or not cloud radiance: lcw4crtm
 - Parameters for the observation error model when lcw4crtm=true
 - Initialize channel related information: predchan; iuse_rad;passive_bc;...
 - Set error instrument channels
 - Logic to turn off print of reading coefficients if not first iteration or not mype_diaghdr or not init_pass
 - Initialize radiative transfer and pointers to values in data_s

call init_crtm(init_pass,iwrmype,mype,nchanl,isis,obstype)
- Get indexes of variables in jacobian to handle exceptions down below
 - Initialize ozone jacobian flags to .false. (retain ozone jacobian)
 - If SSM/I, check for non-use of 85GHz channel, for QC workaround set no85GHz true if any 85GHz is not used, and other freq channel is used no85GHz = .false.
- **Setup for diagnostic**
 - Find number of channels written to diag file
 - Set number of extra pieces of information to write to diagnostic file. For most satellite sensors there is no extra information. However, for GOES Imager data we write additional information.
 - Allocate array to hold channel information for diagnostic file and/or lobsdiagsave option
 - If diagnostic file requested, open unit to file and write header.
 - Initialize/write parameters for satellite diagnostic file on
- **Load data array for current satellite**

```
read(lunin) data_s,luse,oid
```

radiance observation content

```
! These variables are initialized in init_crtm
! isatid    = 1      ! index of satellite id
! itime     = 2      ! index of analysis relative obs time
! ilon      = 3      ! index of grid relative obs location (x)
! ilat      = 4      ! index of grid relative obs location (y)
! ilzen_ang = 5      ! index of local (satellite) zenith angle (radians)
! ilazi_ang = 6      ! index of local (satellite) azimuth angle (radians)
! iscan_ang = 7      ! index of scan (look) angle (radians)
! iscan_pos = 8      ! index of integer scan position
! iszen_ang = 9      ! index of solar zenith angle (degrees)
! isazi_ang = 10     ! index of solar azimuth angle (degrees)
! ifrac_sea = 11     ! index of ocean percentage
! ifrac_lnd = 12     ! index of land percentage
! ifrac_ice = 13     ! index of ice percentage
! ifrac_sno = 14     ! index of snow percentage
! its_sea   = 15     ! index of ocean temperature
! its_lnd   = 16     ! index of land temperature
! its_ice   = 17     ! index of ice temperature
! its_sno   = 18     ! index of snow temperature
! itsavg   = 19     ! index of average temperature
! ivty     = 20     ! index of vegetation type
! ivfr     = 21     ! index of vegetation fraction
! isty     = 22     ! index of soil type
! istp     = 23     ! index of soil temperature
! ism      = 24     ! index of soil moisture
! isn      = 25     ! index of snow depth
! izz      = 26     ! index of surface height
! idomsfc  = 27     ! index of dominate surface type
! isfcr    = 28     ! index of surface roughness
! iff10    = 29     ! index of ten meter wind factor
! ilone    = 30     ! index of earth relative longitude (degrees)
! ilate    = 31     ! index of earth relative latitude (degrees)
! itref    = 34/36   ! index of foundation temperature: Tr
! idtw     = 35/37   ! index of diurnal warming: d(Tw) at depth zob
! idtc     = 36/38   ! index of sub-layer cooling: d(Tc) at depth zob
! itz_tr   = 37/39   ! index of d(Tz)/d(Tr)

! Initialize sensor specific array pointers
! if (goes_img) then
!   iclr_sky    = 7 ! index of clear sky amount
! elseif (avhrr_navy) then
!   issst_navy  = 7 ! index of navy sst (K) retrieval
!   idata_type  = 30 ! index of data type (151=day, 152=night)
!   issst_hires = 31 ! index of interpolated hires sst (K)
! elseif (avhrr) then
!   iclavr     = 32 ! index CLAVR cloud flag with AVHRR data
!   issst_hires = 33 ! index of interpolated hires sst (K)
! elseif (seviri) then
!   iclr_sky    = 7 ! index of clear sky amount
! endif
```

- **PROCESSING OF SATELLITE DATA: Loop over data in this block**

do n = 1,nobs

- Extract analysis relative observation time.
- If desired recompute 10meter wind factor
- Set land/sea, snow, ice percentages and flags (no time interpolation)
- Count data of different surface types
- Set relative weight value

- Interpolate model fields to observation location, call crt_m and create jacobians. Output both tsim and tsim_clr for allsky radiance operator

```

1      if (lcw4crtm) then
2          call call_crtm(obstype,dtime,data_s(:,n),nchanl,nreal,ich, &
3                          tvp,qvp,clw_guess,prsltmp,prsitmap, &
4                          trop5,tzbgr,dtsavg,sfc_speed, &
5                          tsim,emissivity,ptau5,ts,emissivity_k, &
6                          temp,wmix,jacobian,error_status,tsim_clr=tsim_clr)
7      else
8          call call_crtm(obstype,dtime,data_s(:,n),nchanl,nreal,ich, &
9                          tvp,qvp,clw_guess,prsltmp,prsitmap, &
10                         trop5,tzbgr,dtsavg,sfc_speed, &
11                         tsim,emissivity,ptau5,ts,emissivity_k, &
12                         temp,wmix,jacobian,error_status)
13      endif

```

- If the CRTM returns an error flag, do not assimilate any channels for this ob and set the QC flag to ifail_crtm_qc.
- For SST retrieval, use interpolated NCEP SST analysis
- If using adaptive angle dependent bias correction, update the predictors for this part of bias correction. The AMSUA cloud liquid water algorithm uses total angle dependent bias correction for channels 1 and 2
- Compute microwave cloud liquid water or graupel water path for bias correction and QC.
- **COMPUTE AND APPLY BIAS CORRECTION TO SIMULATED VALUES**
 - Construct predictors for 1B radiance bias correction.
 - Apply bias correction
 - emissivity sensitivity bias predictor
 - Apply SST dependent bias correction with cubic spline
 - Compute retrieved microwave cloud liquid water and assign cld_rbc_idx for bias correction in allsky conditions
- Calculate cloud effect for QC
- **QC OBSERVATIONS BASED ON VARIOUS CRITERIA:** Separate blocks for various instruments.

Radiance obs QC

```

! ----- IR -----
! QC HIRS/2, GOES, HIRS/3 and AIRS sounder data
!
ObsQCs: if (hirs .or. goessndr .or. airs .or. iasi .or. cris) then
    frac_sea=data_s(ifrac_sea,n)

! NOTE: The qc in qc_irsnd uses the inverse squared obs error.
! The loop below loads array varinv_use accounting for whether the
! cloud detection flag is set. Array
! varinv_use is then used in the qc calculations.
! For the case when all channels of a sensor are passive, all
! channels with iuse_rad=-1 or 0 are used in cloud detection.

    do i=1,nchanl
        m=ich(i)
        if (varinv(i) < tiny_r_kind) then
            varinv_use(i) = zero
        else
            if ((icld_det(m)>0)) then
                varinv_use(i) = varinv(i)
            else
                varinv_use(i) = zero
            end if
        end if
    end do
    call qc_irsnd(nchanl,is,ndat,nsig,ich,sea,land,ice,snow,luse(n),goessndr, &
                  cris,zsges,cenlat,frac_sea,pangs,trop5,zasat,tzbgr,tsavg5,tbc,tb_obs,tnoise, &
                  wavenumber,ptau5,prsltmp,tvp,temp,wmix,emissivity_k,ts, &
                  id_qc,aivals,errf,varinv,varinv_use,cld,cldp,kmax,zero_irjaco3_pole(n))

```

```

! ----- MSU -----
!   QC MSU data
!   else if (msu) then

    call qc_msu(nchanl,is,ndat,nsig,sea,land,ice,snow,luse(n), &
               zsges,cenlat,tbc,ptau5,emissivity_k,ts,id_qc,aivals,errf,varinv)

! ----- AMSU-A -----
!   QC AMSU-A data
!   else if (amsua) then

    if (adp_anglebc) then
      tb_obsbcl=tb_obs(1)-cbias(nadir,ich(1))-predx(1,ich(1))
    else
      tb_obsbcl=tb_obs(1)-cbias(nadir,ich(1))
    end if
    cldeff_obs5=cldeff_obs(5) ! observed cloud effect for channel 5
    call qc_amsua(nchanl,is,ndat,nsig,npred,sea,land,ice,snow,mixed,luse(n), &
                  zsges,cenlat,tb_obsbcl,cosza,clw,tbc,ptau5,emissivity_k,ts, &
                  pred,predchan,id_qc,aivals,errf,errf0,clwp_amsua,varinv,cldeff_obs5,factch6, &
                  cld_rbc_idx,sfc_speed,error0,clw_guess_retrieval,scatp)

! If cloud impacted channels not used turn off predictor

    do i=1,nchanl
      if ( (i <= 5 .or. i == 15) .and. (varinv(i)<1.e-9_r_kind) ) then
        pred(3,i) = zero
      end if
    end do

! ----- AMSU-B -----
!   QC AMSU-B and MHS data

!   else if (amsub .or. hsb .or. mhs) then

    call qc_mhs(nchanl,ndat,nsig,is,sea,land,ice,snow,mhs,luse(n), &
                zsges,tbc,tb_obs,ptau5,emissivity_k,ts, &
                id_qc,aivals,errf,varinv,clw,tpwc)

! ----- ATMS -----
!   QC ATMS data

!   else if (atms) then

    if (adp_anglebc) then
      tb_obsbcl=tb_obs(1)-cbias(nadir,ich(1))-predx(1,ich(1))
      cldeff_obs5=cldeff_obs(6) ! observed cloud effect for ATMS channel 6
    else
      tb_obsbcl=tb_obs(1)-cbias(nadir,ich(1))
    end if
    call qc_atms(nchanl,is,ndat,nsig,npred,sea,land,ice,snow,mixed,luse(n), &
                 zsges,cenlat,tb_obsbcl,cosza,clw,tbc,ptau5,emissivity_k,ts, &
                 pred,predchan,id_qc,aivals,errf,errf0,clwp_amsua,varinv,cldeff_obs5,factch6, &
                 cld_rbc_idx,sfc_speed,error0,clw_guess_retrieval,scatp)

! ----- GOES imager -----
!   GOES imager Q C
!

!   else if(goes_img)then

    cld = data_s(iclr_sky,n)
    do i = 1,nchanl
      tb_obs_sdv(i) = data_s(i+29,n)
    end do
    call qc_goesimg(nchanl,is,ndat,nsig,ich,dplat(is),sea,land,ice,snow,luse(n), &
                    zsges,cld,tzbgr,tb_obs,tb_obs_sdv,tbc,tnoise,temp,wmix,emissivity_k,ts,id_qc,aivals,errf,
                    varinv)

```

```

! ----- SEVIRI -----
! SEVIRI Q C

    else if (seviri) then

        cld = 100-data_s(iclr_sky,n)

        call qc_seviri(nchanl,is,ndat,nsig,ich,sea,land,ice,snow,luse(n), &
                       zsges,tzbgr,tbc,tnoise,temp,wmix,emissivity_k,ts,id_qc,aivals,errf,varinv)
!

! ----- AVRHRR -----
! NAVY AVRHRR Q C

    else if (avhrr_navy .or. avhrr) then

        frac_sea=data_s(ifrac_sea,n)

! NOTE: The qc in qc_avhrr uses the inverse squared obs error.
!       The loop below loads array varinv_use accounting for whether the
!       cloud detection flag is set. Array
!       varinv_use is then used in the qc calculations.
!       For the case when all channels of a sensor are passive, all
!       channels with iuse_rad=-1 or 0 are used in cloud detection.
        do i=1,nchanl
            m=ich(i)
            if (varinv(i) < tiny_r_kind) then
                varinv_use(i) = zero
            else
                if ((icld_det(m)>0)) then
                    varinv_use(i) = varinv(i)
                else
                    varinv_use(i) = zero
                end if
            end if
        end do

        call qc_avhrr(nchanl,is,ndat,nsig,ich,sea,land,ice,snow,luse(n), &
                      zsges,cenlat,frac_sea,pangs,trop5,tzbgr,tsavg5,tbc,tb_obs,tnoise,      &
                      wavenumber,ptau5,prsltmp,tvp,temp,wmix,emissivity_k,ts, &
                      id_qc,aivals,errf,varinv,varinv_use,cld,cldp)

!

! ----- SSM/I , SSMIS, AMSRE -----
! SSM/I, SSMIS, & AMSRE Q C

    else if( ssmi .or. amsre .or. ssmis )then

        frac_sea=data_s(ifrac_sea,n)
        if(amsre)then
            bearaz= (270._r_kind-data_s(ilazi_ang,n))*deg2rad
            sun_zenith=data_s(iszen_ang,n)*deg2rad
            sun_azimuth=(r90-data_s(isazi_ang,n))*deg2rad
            sgagl = acos(coscon * cos( bearaz ) * cos( sun_zenith ) * cos( sun_azimuth ) + &
                         coscon * sin( bearaz ) * cos( sun_zenith ) * sin( sun_azimuth ) + &
                         sincon * sin( sun_zenith )) * rad2deg
        end if
        call qc_ssmi(nchanl,nsig,ich, &
                     zsges,luse(n),sea,mixed, &
                     temp,wmix,ts,emissivity_k,ierrret,kraintype,tpwc,clw,sgagl,tzbgr, &
                     tbc,tbcnob,tsim,tnoise,ssmi,amsre_low,amsre_mid,amsre_hig,ssmis, &
                     varinv,errf,aivals(1,is),id_qc)

!

! ----- AMSR2 -----
! AMSR2 Q C

    else if (amsr2) then

        sun_azimuth=data_s(isazi_ang,n)
        sun_zenith=data_s(iszen_ang,n)

```

```

    call qc_amr2(nchanl,zsges,luse(n),sea, &
                 kraintype,clw_obs,tsavg5,tb_obs,sun_azimuth,sun_zenith,amsr2,varinv,aivals(1,is),id_qc)

! ----- GMI -----
! GMI Q C

    else if (gmi) then

        call qc_gmi(nchanl,zsges,luse(n),sea,cenlat, &
                     kraintype,clw_obs,tsavg5,tb_obs,gmi,varinv,aivals(1,is),id_qc)

! ----- SAPHIR -----
! SAPHIR Q C

    else if (saphir) then

        call qc_saphir(nchanl,zsges,luse(n),sea, &
                      kraintype,varinv,aivals(1,is),id_qc)

! ----- SSU -----
! SSU Q C

    elseif (ssu) then

        call qc_ssu(nchanl,is,ndat,nsig,sea,land,ice,snow,luse(n), &
                    zsges,cenlat,tb_obs,ptau5,emissivity_k,ts,id_qc,aivals,errf,varinv)

    end if ObsQCs

! Done with sensor qc blocks. Now make final qc decisions.

! Apply gross check to observations. Toss obs failing test.
do i = 1,nchanl
    if (varinv(i) > tiny_r_kind ) then
        m=ich(i)
        if(lcw4crtm .and. sea) then
            if (i <= 3 .or. i==15) then
                errf(i) = 3.00_r_kind*errf(i)
            else if (i == 4) then
                errf(i) = 3.00_r_kind*errf(i)
            else if (i == 5) then
                errf(i) = 3.00_r_kind*errf(i)
            else
                errf(i) = min(three*errf(i),ermax_rad(m))
            endif
        else if (ssmis) then
            errf(i) = min(1.5_r_kind*errf(i),ermax_rad(m)) ! tighten up gross check for SSMIS
        else if (gmi .or. saphir .or. amsr2) then
            errf(i) = ermax_rad(m) ! use ermax for GMI, SAPHIR, and AMSR2 gross check
        else
            errf(i) = min(three*errf(i),ermax_rad(m))
        endif
    if (abs(tbc(i)) > errf(i)) then
        If mean obs-ges difference around observations
        location is too large and difference at the
        observation location is similarly large, then
        toss the observation.
        if(id_qc(i) == igood_qc)id_qc(i)=ifail_gross_qc
        varinv(i) = zero
        if(luse(n))stats(2,m) = stats(2,m) + one
        if(luse(n))aivals(7,is) = aivals(7,is) + one
    end if
    end if
end do

if(amsua .or. amsub .or. mhs .or. msu .or. hsb)then
    if(amsua)nlev=6
    if(amsub .or. mhs)nlev=5
    if(hsb)nlev=4
    if(msu)nlev=4
    kval=0

```

```

        do i=2,nlev
        do i=1,nlev
            channel_passive=iuse_rad(ich(i))==-1 .or. iuse_rad(ich(i))==0
            if (varinv(i)<tiny_r_kind .and. ((iuse_rad(ich(i))>=1) .or. &
                (passive_bc .and. channel_passive))) then
                kval=max(i-1,kval)
                if(amsub .or. hsb .or. mhs)kval=nlev
                if(amsua .and. i <= 3)kval = zero
            end if
        end do
        if(kval > 0)then
            do i=1,kval
                varinv(i)=zero
                if(id_qc(i) == igood_qc)id_qc(i)=ifail_interchan_qc
            end do
            if(amsua)then
                varinv(15)=zero
                if(id_qc(15) == igood_qc)id_qc(15)=ifail_interchan_qc
            end if
        end if
    end if

```

- Only process observations to be assimilated (fill in radhead, radtail)
- At the end of analysis, prepare for bias correction for monitored channels.
- Only "good monitoring" obs are included in J_passive calculation.
- summation of observation number
- Write diagnostics to output file: diagbuf; diagbufex; diagbufchan

diagnosis information

```

! Write diagnostics to output file.
if (rad_diagsave .and. luse(n) .and. nchanl_diag > 0) then
    diagbuf(1) = cenlat                                ! observation latitude (degrees)
    diagbuf(2) = cenlon                                ! observation longitude (degrees)
    diagbuf(3) = zsges                                  ! model (guess) elevation at observation location

    diagbuf(4) = dtime-time_offset                      ! observation time (hours relative to analysis time)

    diagbuf(5) = data_s(iscan_pos,n)                   ! sensor scan position
    diagbuf(6) = zasat*rad2deg                          ! satellite zenith angle (degrees)
    diagbuf(7) = data_s(ilazi_ang,n)                   ! satellite azimuth angle (degrees)
    diagbuf(8) = pangs                                 ! solar zenith angle (degrees)
    diagbuf(9) = data_s(isazi_ang,n)                   ! solar azimuth angle (degrees)
    diagbuf(10) = sgagl                                ! sun glint angle (degrees) (sgagl)

    diagbuf(11) = surface(1)%water_coverage           ! fractional coverage by water
    diagbuf(12) = surface(1)%land_coverage            ! fractional coverage by land
    diagbuf(13) = surface(1)%ice_coverage             ! fractional coverage by ice
    diagbuf(14) = surface(1)%snow_coverage            ! fractional coverage by snow
    if(.not. retrieval)then
        diagbuf(15) = surface(1)%water_temperature     ! surface temperature over water (K)
        diagbuf(16) = surface(1)%land_temperature       ! surface temperature over land (K)
        diagbuf(17) = surface(1)%ice_temperature         ! surface temperature over ice (K)
        diagbuf(18) = surface(1)%snow_temperature        ! surface temperature over snow (K)
        diagbuf(19) = surface(1)%soil_temperature        ! soil temperature (K)
        if (gmi .or. saphir) then
            diagbuf(20) = gwp                           ! graupel water path
        else
            diagbuf(20) = surface(1)%soil_moisture_content ! soil moisture
        endif
        diagbuf(21) = surface(1)%land_type              ! surface land type
    else
        diagbuf(15) = tsavg5                         ! SST first guess used for SST retrieval
        diagbuf(16) = sstcu                           ! NCEP SST analysis at t
        diagbuf(17) = sstph                           ! Physical SST retrieval
        diagbuf(18) = sstnv                           ! Navy SST retrieval
        diagbuf(19) = dta                            ! d(ta) corresponding to sstph
        diagbuf(20) = dqa                            ! d(qa) corresponding to sstph
    endif
endif

```

```

        diagbuf(21) = dtp_avh                                ! data type
    endif
    if(lcw4crtm .and. sea) then
        ! diagbuf(22) = tpwc_amsua
        diagbuf(22) = scat                                 ! scattering index from AMSU-A
        diagbuf(23) = clw_guess                            ! integrated CLWP (kg/m**2) from
background
    else
        diagbuf(22) = surface(1)%vegetation_fraction     ! vegetation fraction
        diagbuf(23) = surface(1)%snow_depth                ! snow depth
    endif
    diagbuf(24) = surface(1)%wind_speed                  ! surface wind speed (m/s)

!
! Note: The following quantities are not computed for all sensors
if (.not.microwave) then
    diagbuf(25) = cld                                    ! cloud fraction (%)
    diagbuf(26) = cldp                                  ! cloud top pressure (hPa)
else
    if((lcw4crtm .and. sea) .or. gmi .or. amsr2) then
        if (gmi .or. amsr2) then
            diagbuf(25) = clw_obs                      ! clw (kg/m**2) from retrievals
        else
            diagbuf(25) = clwp_amsua                   ! cloud liquid water (kg/m**2)
        endif
        diagbuf(26) = clw_guess_retrieval             ! retrieved CLWP (kg/m**2) from simulated
BT
    else
        diagbuf(25) = clw                        ! cloud liquid water (kg/m**2)
        diagbuf(26) = tpwc                      ! total column precip. water (km/m**2)
    endif
endif

!
! For NST
if (nstinfo==0) then
    diagbuf(27) = r_missing
    diagbuf(28) = r_missing
    diagbuf(29) = r_missing
    diagbuf(30) = r_missing
else
    diagbuf(27) = data_s(itref,n)
    diagbuf(28) = data_s(idtw,n)
    diagbuf(29) = data_s(idtc,n)
    diagbuf(30) = data_s(itz_tr,n)
endif

if (lwrite_peakwt) then
    do i=1,nchanl_diag
        diagbufex(1,i)=weightmax(ich_diag(i))   ! press. at max of weighting fn (mb)
    end do
    if (goes_img) then
        do i=1,nchanl_diag
            diagbufex(2,i)=tb_obs_sdv(ich_diag(i))
        end do
    end if
else if (goes_img .and. .not.lwrite_peakwt) then
    do i=1,nchanl_diag
        diagbufex(1,i)=tb_obs_sdv(ich_diag(i))
    end do
end if

do i=1,nchanl_diag
    diagbufchan(1,i)=tb_obs(ich_diag(i))          ! observed brightness temperature (K)
    diagbufchan(2,i)=tbc(ich_diag(i))            ! observed - simulated Tb with bias correction (K)
    diagbufchan(3,i)=tbcnob(ich_diag(i))         ! observed - simulated Tb with no bias correction (K)
    errinv = sqrt(varinv(ich_diag(i)))
    diagbufchan(4,i)=errinv                      ! inverse observation error
    useflag=one
    if (iuse_rad(ich(ich_diag(i))) < 1) useflag=-one
    diagbufchan(5,i)= id_qc(ich_diag(i))*useflag      ! quality control mark or event indicator

    if (lcw4crtm) then

```

```

diagbufchan(6,i)=error0(ich_diag(i))
else
  diagbufchan(6,i)=emissivity(ich_diag(i))           ! surface emissivity
endif
diagbufchan(7,i)=tlapchn(ich_diag(i))               ! stability index
if (lcw4crtm) then
  diagbufchan(8,i)=cld_rbc_idx(ich_diag(i))        ! indicator of cloudy consistency
else
  diagbufchan(8,i)=ts(ich_diag(i))                  ! d(Tb)/d(Ts)
end if

if (lwrite_predterms) then
  predterms=zero
  do j = 1,npred
    predterms(j) = pred(j,ich_diag(i))
  end do
  predterms(npred+1) = cbias(nadir,ich(ich_diag(i)))

  do j=1,npred+2
    diagbufchan(ipchan_radiag+j,i)=predterms(j) ! Tb bias correction terms (K)
  end do
else ! Default to write out predicted bias
  do j=1,npred+2
    diagbufchan(ipchan_radiag+j,i)=predbias(j,ich_diag(i)) ! Tb bias correction terms (K)
  end do
  end if
end do

```

End of n-loop over obs

- release memory