A general purpose three-dimensional data assimilation software developed at CNR-ISAC

By
Stefano Federico, Rosa Claudia Torcasio, Claudio Transerici

$$2J(\nu) = \nu\nu^T + (y'_o - \mathrm{H}U\nu)^T R^{-1}(y'_o - \mathrm{H}U\nu)$$

$$\nabla J = \nu + U^T \mathrm{H}^T R^{-1}(y'_o - \mathrm{H}U\nu)$$

1. The CNR-ISAC 3DVar: a general overview

The CNR-ISAC 3DVar software is a general purpose 3DVar that can be used for research purposes, for teaching and for operations. The software is able to assimilate the following types of data: radio soundings, atmospheric motion vectors (AMV), radar reflectivity (with two different methods), ASCAT (Advanced Scatterometer) surface winds, WIVERN (**WI**nd **VE**locity **R**adar **N**ephoscope) winds, lightning, GNSS-ZTD (Global Navigation Satellite System - Zenith Tropospheric Delay), and rain rates in convective environments.
The software is general purpose and can be used with any model. An interface is provided for the WRF model in Interactive Data Language (IDL; https://www.nv5geospatialsoftware.com/Products/IDL; this software needs a license to be used) programming language, however you must provide the interface for your model (including WRF if you cannot run the IDL software). In this short manual we will consider that the 3DVar software is used with the WRF model.
The 3DVar software of CNR-ISAC is organised in two main components with executables installed in two different directories: **w3dvar_main.exe** installed in the folder **w3dvar_par**, and the software **w3dvar_node.exe** installed in the folder **w3dvar_par_node**. There is also a third directory (inmc) where you can find simple programs for computing the background error matrix in the horizontal and vertical directions (seesection “The background error matrix” for further details).
The software **w3dvar_main.exe** does the decomposition of the domain in *n* tiles for parallelization. The number of tiles can be chosen by the user (parameter nproc in the **w3dvar_main.f90** program of the **w3dvar_par** directory) and must be a squared number; 4 or 9 are recommended for 100–500 grid points points in WE and NS directions; 16 or 25 for larger number of grid points.
The **w3dvar_par** folder is also where the software is executed, where the input data are provided and where the output of 3DVar is produced. A simple program (**plot_3dvar.pro**) is provided to graph the software output. This code is written in IDL.
The software **w3dvar_par_node** does most of the work: after compilation the software **w3dvar_node.exe** must be copied in the folder **w3dvar_main** for the execution of the code.
The 3DVar is launched via the command **w3dvar_main.exe**; this command calls the code **w3dvar_node.exe**, collects the output form the nodes, and make the final output. This is an important point; the **w3dvar_par_node** directory is intended to be used for code developing.
The **w3dvar_main.exe** reads the date of the analysis from an external file called **date_exec**.
For each of the nodes, a file called **anl_reduced_+*node_number*.dat** is written in the **dat** subdirectory. This file contains the data from the background WRF output which are used for the analysis. The processes for the nodes are launched via the command
w3dvar_node.exe nodenumber
and the node number is always provided with three digits format (for example 001). The **w3dvar_main.exe** does this job.
For each of the nodes, when the execution of the process **w3dvar_node.exe** is finished, an output file called ***out_1d+3dvar_YYYYMMDDHHMinMin_nodenumber.dat*** is produced (YYYY is the year, MM is the month, DD is the day, HH is the hour, MinMin are the minutes). When this output is available for each of the nodes, the **w3dvar_main.exe** program recombines the output files of the nodes in one file for the whole domain, named
***out_1d+3dvar_YYYYMMDDHHMinMin.dat*.**
This is the final output of the code, containing the analysis.
The development of the 3DVar software started in 2010 and its reference paper is Federico (2013). In the original software, radiosoundings were the only data that could be assimilated. While the general architecture of the software is unchanged, differently from Federico (2013) the 3DVar doesn’t use anymore recursive filters, which have been substituted by gaussians, with length scales derived from the NMC method (Parrish and Derber, 1992; Barker et al., 2004).
The control variables assimilated in the software are: 1) water vapor mixing ratio, 2) air temperature, 3) rain mixing ratio, 4) zonal wind component, 5) meridional wind component. The

state vector is given by these variables in the order given above. The software uses the incremental formulation of the cost function, as described in Federico (2013).

2. How to compile

The 3DVar software is very easy to compile. All the software is written in Fortran 90. The is no Makefile. Once unzipped the software, two directories are created: **w3dvar_par** and **w3dvar_par_node**. In the **w3dvar_main** directory you will find the ./**compila_3dvar** script; once executed via the command:

$*./compila_3dvar*

The file **w3dvar_main.exe** will be generated. See the **compila_3dvar** file for setting the paths needed.
Similarly, in the **w3dvar_par_node** directory you find the script **compila_3dvar_node**. Once executed via the command:

$*./compila_3dvar_node*

The file **w3dvar_node.exe** will be generated. Again, once the executable **w3dvar_node.exe** is generated it must be put in the **w3dvar_par** directory for the execution of the 3DVar software.
After compilation, the software is run by the command:

$*./w3dvar_main.exe*

The **w3dvar_par** directory contains the following subdirectories: **dat, src, lib, dat_mw, dir_cappi, dir_gps, dir_light, dir_wind, ps, uh_transform**.
The **dat** subdirectory contains temporary files that are generated by the software at the runtime (for example data for the domain decomposition). The **src** subdirectory contains the software. The **lib** subdirectory contains libraries that are generated during the compilation of the software. The **dat_mw** subdirectory contains the rain-rates estimate from microwave observations that can be used for data assimilation. The subdirectory **dir_cappi** contains the radar reflectivity CAPPI (Constant Altitude Pain Position Indicator), which can be assimilated by the software. The subdirectory **dir_gps** contains the GNSS-ZTD data to be assimilated by the software. The subdirectory **dir_light** contains the lightning data to be assimilated by the software. The subdirectory **dir_wind** contains the wind data to be assimilated by the software (ASCAT, WIVERN, AMV). The subdirectory **ps** is intended for producing graphical output of the software. The directory **uh_transform** contains the transforms provided by the NMC method.
The **w3dvar_par_node** directory has the same structure of the subdirectories even if they are not used. This is motivated by the possibility, not yet implemented, to put the two software (**w3dvar_main.exe** and **w3dvar_node.exe**) together in one directory.

3. The w3dvar_main software

This software is used to launch the 3DVar via the **w3dvar_main.exe** command. The input data is very simple and it is provided in the file **date_exec**. The file **date_exec** is composed of two lines. In the first line it is provided the date of the analysis, while in the second line indicates the directory where the first guess file can be found.
An example of the **data_exec** file is provided below:

20200917120000
/home/user/WRFDAL/var/run/

The date use a 14 digits format YYYYMMDDHHMinMinSS.

The purpose of the **w3dvar_main** software is to read the variables needed by the 3DVar and to decompose the domain in *n* tiles where *n* must be a squared number (1,4,9,16,25,…). Note that 1 corresponds to a serial call of the 3DVar code. The parallelization of the code is made by writing simple ASCII files in the **dat** subdirectory and there is no need to compile with parallelization libraries (as MPICH). It is important to know that a halo region is defined for each tile. This halo region has a dimension of 3 times the lengthscale of the maximum background error decorrelation lengthscale. This means that as the maximum lengthscale becomes larger, the parallelization efficiency of the code decreases. In addition, the software cannot be run across different computers; it should be run on a single multicore machine.

3.1 The interface

An important element of the **w3dvar_main** software is that it calls the interface to generate suitable data for the 3DVar. In the current release of the software, the interface is provided by the IDL program **netcdf_wrf_for_3dvar.pro** . This program is intended to work with the WRF model. For other models you must create your interface. The **leggi_wrf** subroutine in the **leggi_vars.f90** routines (in the **src** directory) can be adapted for your case. The following inputs are requested: nx, ny, nz, xtn, ytn, zstar, longitude, latitude, orography, land fraction, geopotential height, temperature in kelvin, water vapour mixing ratio, relative humidity, reflectivity, pressure, rain mixing ratio, zenith total delay, dry zenith total delay, positive flashes, negative flashes, intracloud flashes, zonal velocity, meridional velocity. A short explanation is provided below:

The nx is the number of grid points in the WE direction; ny is the number of grid points in the SN direction, nz is the number of vertical levels. xtn is the distance along the x axis. For example, if you have a resolution of 3km in you model you can put xtn(1)=0., xtn(2)=3000., xtn(2)=6000., etc. xtn is a 1 dimensional vector and its size is nx. Similarly, ytn is the distance along the y axis. It is a 1 dimensional vector whose dimension is the number of grid points along the y direction (ny). Zstar is the average height of the model level. It is a one-dimension vector and has the dimension of the levels' number (nz). Longitude, latitude, orography and land fraction are all bi-dimensional vectors (nx,ny) (the first dimension is the WE direction, while the second dimension run in the SN direction) containing the relative parameter. The geopotential height, temperature, water vapour mixing ratio (kg/kg), relative humidity, pressure (in Pa), rain mixing ratio (kg/kg) and the wind components (m/s) are all three dimensional vectors with dimensions (nx,ny,nz), containing the parameter to be assimilated. Positive, negative and intracloud flashes are two dimensional vectors with (nx,ny) dimensions. If you do not intend to assimilate lightning you can provide all zeroes for these vectors.

In general, if you provide the above vectors to the 3dvar software, the code should be able to work with any model. This change can be done In the **leggi_wrf** subroutine which is contained in the **leggi_wrf.f90** routine in the **src** subdirectory of the **w3dvar_par** directory.

4. The w3dvar_node software

As stated above, while the **w3dvar_par** directory contains the code for the decomposition of the domain, for the parallelization of the code and for getting the output from the nodes and recomposing the final analysis,the assimilation is done by the software **w3dvar_node.exe**.

The software **w3dvar_node.exe** contains the list of the observations that can be assimilated (logical variables). To select the observations to assimilate it is necessary to set to *.true.* the corresponding variable (of course for any change in the fortran code a new compilation of the software is needed). Here is an example of the code to search and to adapt for your needs:

exist_light=.false.
exist_radar=.false.
sat_analisi=.true.
indirect_warm=.false.
wind_wivern=.false.

ascat_wind=.false.
wind=.false.
refl_wivern=.false.
amv=.true.

You can find this part of the code in the **w3dvar_node.f90** program of the **w3dvar_par_node**. In the above example, the 3DVar will check for the Atmospheric Motion Vector (AMV) data and for the rain-rates observed by satellites, all other data are not assimilated. In any case the software will check for the existence of the file containing the observations. If the file it is not found, the cost function associated with the specific observation will not be considered and the observation not assimilated. You can put all the above logical values to true and let the software check for the existence of observations and for the activation of the cost function. If a logical variable in the above list is set to false, the observation will not be checked at all and the cost function associated with this observation is not considered even if there is the file containing the observations.
Apart for the lightning and radar reflectivity data assimilation, for which the coding is different (future updates of the software will solve this issue of different coding), all other sources of data have a similar coding. In this short manual we consider the example of ASCAT surface wind data assimilation.
In the **w3dvar_node.f90** program of the **w3dvar_par_node** directory search for the lines:

if(ascat_wind) then
call elab_ascat(nxa,nya,nza,date_exec,anl%up,anl%vp,anl%glon,anl%glat,anl%zeta,dir_uv)
endif

Then go in the **src** subdirectory of the **w3dvar_par_node** directory and search for the **elab_ascat** subroutine trough the command *grep*.

*[stefano@amd1 src]$ grep elab_ascat *.f90*
elab_uv.f90:subroutine elab_ascat(nxa,nya,nza,date_exec,up,vp,glon,glat,zeta,dir_data)
elab_uv.f90:print,'*** subroutine elab_ascat ***'*

You can see that the **elab_ascat** subroutine is in the **elab_uv.f90** program in the **src** subdirectory. Then open the **elab_uv.f90** program and search for the **elab_ascat** subroutine and see what type of elaborations are done for each observation type.
In general a subroutine as **elab_ascat** does the following actions: a) read the datafile and determine which are the observations falling in the specific portion of domain; b) set the error covariance matrix for the observations; c) determine the vector O-B (observations minus background).

4.1 An example: preparing ASCAT data for data assimilation
Let us examine the process of assimilating ASCAT wind observations more in detail. The call to the subroutine **elab_ascat** is the following:

```
subroutine elab_ascat(nxa,nya,nza,date_exec,up,vp,glon,glat,zeta,dir_data)
```

nxa,nya,nza are the number of the grid points of the tile used by the specific process, date_exec is the date of the analysis (14 digits) up, vp are the model zonal and meridional wind components (three dimensional vectors of nxa,nya,nza), glon and glat are the longitude and latitude of the tile (two dimensional vectors of nxa,nya) zeta is the geopotential height and dir_data is the name of the directory where the software searchs for ASCAT data. Note that all vectors are known to the software at this point and come from the domain decomposition provided by the parallelization.
The following part of the software reads the observations and checks if they are included in the domain analyzed by the 3DVar:

```
open(11,file=trim(flname),status='old')
allocate(obs_lon(nobs),obs_lat(nobs),obs_quota(nobs),obs_up(nobs),obs_vp(nobs),valid(nobs))
allocate(obs_qc(nobs),obs_wsp(nobs),obs_wdir(nobs))
do i=1,nobs
read(11,'(a14,1x,7(f12.3))')
ch1,obs_lat(i),obs_lon(i),obs_up(i),obs_vp(i),obs_wsp(i),obs_wdir(i),obs_qc(i)
obs_quota(i)=10.
enddo
close(11)

valid(:)=0.

ic_net=0

do iobs=1,nobs
xlon=obs_lon(iobs)
xlat=obs_lat(iobs)
call nearest(glon,glat,nxa,nya,xlon,xlat,x_pos,y_pos) ! this routine determines the position of the
                                                      ! observation in the domain considered. It can
                                                      ! be found in the src subdirectory in the
                                                      ! sf_interp.f90 program
ii=nint(x_pos)
jj=nint(y_pos)
if(x_pos .gt. 1. .and. x_pos .lt. nxa-1 .and. y_pos .gt. 1.  &
.and. y_pos .lt. nya-1 .and. obs_up(iobs) .ne. amiss .and. obs_vp(iobs) .ne. amiss) then
ic_net=ic_net+1
valid(iobs)=1.
endif
enddo

nobs_all=nobs
nobs=ic_net

if(nobs .eq. 0) return
```

Note that if no observations are found the subroutine returns. Then we must populate the obs_ascat structure, which is defined in the the program **memory_3dvar.f90**. Continuing in the program **elab_ascat.f90**:

```
allocate(obs_ascat%lon(nobs),obs_ascat%lat(nobs),obs_ascat%quota(nobs),   &
obs_ascat%x_pos(nobs),obs_ascat%y_pos(nobs),obs_ascat%z_pos(nobs))
allocate(obs_ascat%erre_up(nobs,nobs),obs_ascat%erre_vp(nobs,nobs),   &
obs_ascat%erre_up_1(nobs,nobs),obs_ascat%erre_vp_1(nobs,nobs))
allocate(obs_ascat%up(nobs),obs_ascat%vp(nobs),obs_ascat%wsp(nobs)  &
,obs_ascat%wdir(nobs))
```

In the structure above, x_pos and y_pos are the position of the observation in the subdomain of the analysis program and are real numbers. The erre_up and erre_vp are the observations for matrices for the zonal and meridional wind components. They are square matrices of dimension nobs whose diagonal elements are the error associated with the observation and the off-diagonal elements are the error correlations among different observations. The erre_up_1 and erre_vp_1 are the inverse of erre_up and erre_vp matrices. As you can see from the above allocations, we define also the wsp and wdir, nevertheless they are not strictly necessary as the 3DVar uses the zonal and meridional wind components as control variables. In any case, the subroutines **convert_ang_aer_uv** and **convert_uv_ang_aer** give the possibility to convert from wsp and wdir to zonal and meridional wind components and vice-versa. These routines are included in the **sf_interp.f90** program in the **src** subdirectory of the **w3dvar_par_node** directory. Then, following in the **elab_ascat.f90** routine we find the code:

```
ic_net=0
do iobs=1,nobs_all
if(valid(iobs) .eq. 1.) then
ic_net=ic_net+1
obs_ascat%lon(ic_net)=obs_lon(iobs)
obs_ascat%lat(ic_net)=obs_lat(iobs)
obs_ascat%up(ic_net)=obs_up(iobs)
obs_ascat%vp(ic_net)=obs_vp(iobs)
obs_ascat%wdir(ic_net)=obs_wdir(iobs)
obs_ascat%wsp(ic_net)=obs_wsp(iobs)
obs_ascat%quota(ic_net)=obs_quota(iobs)
```

```
endif
enddo

obs_ascat%nobs=ic_net
nobs=ic_net

print*,'Osservazioni nette ASCAT: ',obs_ascat%nobs
```

The above part of the code transfers the observations to the obs_ascat structure. Then followsthe initialization of the observation error matrix and of its inverse:

```
!CICLO SULLE OSSERVAZIONI NETTE E CALCOLO DEI PARAMETRI NECESSARI ALLA 3DVAR
obs_ascat%erre_up_1(:,:)=0.
obs_ascat%erre_vp_1(:,:)=0.
obs_ascat%erre_up(:,:)=0.
obs_ascat%erre_vp(:,:)=0.
```

The option for printing the O-B statistics are activated if o_b_anl logical variable is set to true. This variable can be found at the start of the elab_ascat subroutine.

```
!------ O-B
if(o_b_anl) then
open(12,file='./ascat_o-b.dat',status='unknown')
write(12,*) nobs
endif
!-----
```

The following blocks does most of the job. First the vectors x_pos and y_pos of the obs_ascat structure are filled (the nearest subroutine can be found in the **sf_interp.f90** subroutine in the **src** subdirectory); then the observation error matrices are set (in this simple case an error of 2 m/s for each wind component is used) and the inverse of the error matches are computed. Note that the *obs_ascat%erre_up* and *obs_ascat%erre_vp* matrices are the square root of the error matrices, while *obs_ascat%erre_up_1* and *obs_ascat%erre_vp_1* are the inverse of the error matrices.

```
do iobs=1,nobs
xlon=obs_ascat%lon(iobs)
xlat=obs_ascat%lat(iobs)
call nearest(glon,glat,nxa,nya,xlon,xlat,x_pos,y_pos)
obs_ascat%x_pos(iobs)=nint(x_pos)
obs_ascat%y_pos(iobs)=nint(y_pos)
ii=nint(x_pos)
jj=nint(y_pos)
obs_ascat%z_pos(iobs)=1
kk=obs_ascat%z_pos(iobs)
obs_ascat%erre_up(iobs,iobs)=2.0
obs_ascat%erre_vp(iobs,iobs)=2.0  !
print*,iobs,nobs,obs_ascat%erre_up(iobs,iobs),obs_ascat%erre_vp(iobs,iobs)
obs_ascat%erre_up_1(iobs,iobs)=1./(obs_ascat%erre_up(iobs,iobs)**2.)
obs_ascat%erre_vp_1(iobs,iobs)=1./(obs_ascat%erre_vp(iobs,iobs)**2.)
!-------- O-B
if(o_b_anl) then
write(12,*) obs_ascat%lon(iobs),obs_ascat%lat(iobs),obs_ascat%up(iobs),up(ii,jj,kk),   &
            obs_ascat%vp(iobs),vp(ii,jj,kk)
endif
!----------
obs_ascat%up(iobs)=obs_ascat%up(iobs)-up(ii,jj,kk)      ! The model wind is that at the first level
obs_ascat%vp(iobs)=obs_ascat%vp(iobs)-vp(ii,jj,kk)
enddo
```

Finally, the output file for the O-B analysis is closed, if the O-B analysis is requested.

```
!----- O-B
if(o_b_anl) then
close(12)
!stop
endif
```

After providing the data to the 3DVar software, the cost function is activated and the subroutine is concluded.

```
cost_ascat=.true.
```

```
print*,'Attivo l'' assimilazione ASCAT:',cost_ascat
deallocate(obs_lon,obs_lat,obs_quota,obs_up,obs_vp,valid,obs_wdir,obs_wsp)

return
end
```

4.2 Assimilation of rain rate in convective environments

The assimilation of rain-rate in convective environments applies the simple cloud model of Torcasio et al., Remote Sens. 2024, 16(10), 1769; https://doi.org/10.3390/rs16101769. In this simple model the atmosphere is considered saturated from the lifting condensation level (LCL, computed from the model fields) to the -25°C isotherm, which is the top of the electrification layer. The only variable assimilated in this scheme is the water vapor mixing ratio.

The assimilation is managed by two programs of the **w3dvar_node.exe** software, which are in the subdirectory **src**. These two programs are:

elab_sat_rate_aero.f90
cost_function.f90

In addition to the **module_3dvar.f90** which contains the declaration of the observation structures and of the **memory.f90** module, which contains the definition of the logical variable **cost_aero,** which is activated in the case there are rain-rate observations that are above the rain threshold.

The structure used for saving the variables are of this type:

```
type aeromet_rrate
real, allocatable :: lon(:),lat(:),quota(:),x_pos(:),y_pos(:),z_pos(:),qs(:)
real, allocatable :: erre_qs(:,:),erre_qs_1(:,:),rrate(:)
character(len=14), allocatable :: date(:)
integer :: nobs
end type aeromet_rrate
```

The *lon* and *lat* variables are the longitude and the attitude of the observation, *quota* is the height of the observation from the geoid, *x_pos, y_pos, z_pos* are the positions of the observations in the model grid space (they are real numbers), *qs* is the mixing ratio of the observations (i.e. the saturation mixing ratio in the observation position computed from the model parameters), *erre_qs* is the square root of error matrix and *erre_qs_1* is the inverse of the error matrix.

The program **elab_sat_rate_aero.f90** reads the data and prepares the vectors for the cost-function, while the second program computes the cost-function and the gradient of the cost-function. The most important part of the program are detailed below:

The call to the subroutine is made from the **w3dvar_node.f90** program and is:

```
if(aero_rrate) then
call elab_sat_rate_aero(nx,ny,nz,nzr,anl%lcl,date_exec,anl%zeta,  &
anl%glon,anl%glat,anl%temp,anl%press,anl%rv,dir_light)
endif
```

where nx,ny,nz are the dimension of the subdomain integrated by the node, anl%lcl is the lightning condensation level (computed in the initial part of the program w3dvar_node.f90, search for the call to the **calcola_lcl** subroutine), *date_exec* is the date in the format YYYYMMDDHHMinMinSS (14 characters), *anl%zeta* is the geopotential height, *anl%glon*, is the longitude, *anl%glat* is the latitude, *anl%temp* is the temperature, *anl%press* is the pressure and *anl%rv* is the water vapor mixing ratio, *dir_light* is the directory where there are the observations of satellite derived rain-rate.

All the parameters a*nl%zeta, anl%glon,anl%glat,anl%temp,anl%press,anl%rv* are passed to the node by the *w3dvar_node.f90* program by the *w3dvar_main.f90* program, and refers to the subdomain that is considered by the specific node.

In the **elab_sat_rate_aero.f90** program there are two important parameters:

```
real, parameter :: rain_th=1.,t_25=248.15
```

the rain_th parameter is the minimum rain-rate threshold for the activation of the data assimilation. This parameter is very important and must be tuned for your specific application. The parameter t_25 is the -25°C isotherm expressed in Kelvin and indicate the top of the convective area. The observation file is named *aero_to3dv_YYYYMMDDHHMinMin.dat* and contains the following information: longitude, latitude, rain-rate, date in the format '(3(f13.5,1x),(a14))'.
Note that the date inside the file is not checked and it is assumed that all the observations refer to the time included in the name of the observation file indicated above. The following part of the code shows how the observations are read and put in the data structure of the 3DVar code:

```
ic=0
open(1,file=trim(dir_light)//flname,status='old')
do i=1,nrec
read(1,'(3(f13.5,1x),(a14))') x1,x2,x3,ch_date
obs_aero_rrate%lon(i)=x1
obs_aero_rrate%lat(i)=x2
obs_aero_rrate%rrate(i)=x3
obs_aero_rrate%date(i)=ch_date
xlon=obs_aero_rrate%lon(i)
xlat=obs_aero_rrate%lat(i)
call nearest(glon,glat,nx,ny,xlon,xlat,x_pos,y_pos) ! the subroutine nearest is called to find
                                                    ! the position in the horizontal of the
                                                    ! observation in the model
                                                    ! grid. x_pos and y_pos are real numbers.

ii=nint(x_pos)
jj=nint(y_pos)
obs_aero_rrate%x_pos(i)=ii
obs_aero_rrate%y_pos(i)=jj
if(ii .gt. 1 .and. ii .lt. nx .and. jj .gt. 1 .and. jj .lt. ny    &  ! check if the observation is
                                        ! inside the subdomain and the rain-rate is larger than
                                    ! the threshold. If so, the observation is retained.
   .and. obs_aero_rrate%rrate(i) .gt. rain_th) then
do k=1,nz
if(zeta(ii,jj,k) .gt. lcl(ii,jj) .and. temp(ii,jj,k) .gt. t_25) then !check if the level is between
the LCL and the 25°C isotherm.
if(mask(ii,jj,k) .eq. 1) then    !If the observation has already been considered this check skip the
                                 !observation.
goto 133
endif
mask(ii,jj,k)=1.
ic=ic+1
endif
enddo
endif
133 continue
enddo
close(1)
```

After, the new data structure obs_aero_rrate_fin is defined and the valid data are transferred to this structure:

```
naux=nint(sum(mask))

if(naux .eq. 0) return
allocate(obs_aero_rrate_fin%lon(naux),obs_aero_rrate_fin%lat(naux),obs_aero_rrate_fin%quota(naux))
allocate(obs_aero_rrate_fin%x_pos(naux),obs_aero_rrate_fin%y_pos(nrec),obs_aero_rrate_fin%z_pos(naux
))
allocate(obs_aero_rrate_fin%date(naux),obs_aero_rrate_fin%rrate(nrec),obs_aero_rrate_fin%qs(naux))
allocate(obs_aero_rrate_fin%erre_qs(naux,naux),obs_aero_rrate_fin%erre_qs_1(naux,naux))

obs_aero_rrate_fin%erre_qs(:,:)=0.
obs_aero_rrate_fin%erre_qs_1(:,:)=0.

obs_aero_rrate_fin%nobs=naux
ic=0
do j=1,ny
do i=1,nx
do k=1,nz
if(mask(i,j,k) .eq. 1.) then
ic=ic+1
!print*,ic,i,nrec
obs_aero_rrate_fin%lon(ic)=glon(i,j)
obs_aero_rrate_fin%lat(ic)=glat(i,j)
obs_aero_rrate_fin%quota(ic)=zeta(i,j,k)    ! The observations' positions are saved in the
obs_aero_rrate_fin%x_pos(ic)=i              ! structure both in the geographical and model grid space
obs_aero_rrate_fin%y_pos(ic)=j
```

```
obs_aero_rrate_fin%z_pos(ic)=k           ! After the mask is defined, we know also the vertical
                                         !  position of the observation in the model grid.
x1=rslif(press(ii,jj,k)*100.,temp(i,j,k))
obs_aero_rrate_fin%qs(ic)=max(rv(i,j,k),1.03*x1)-rv(i,j,k)
obs_aero_rrate_fin%erre_qs(ic,ic)=0.25*err_rv_z(k,k)   ! The observation error is 0.25 of the
corresponding background error. In this scheme we assume that
obs_aero_rrate_fin%erre_qs_1(ic,ic)=1./(obs_aero_rrate_fin%erre_qs(ic,ic)**2.). !R^-1
endif
enddo
enddo
enddo


if(obs_aero_rrate_fin%nobs .ge. 1) cost_aero=.true.  ! Finally the cost_aero logical variable is
                                                     ! activated and the program returns.
return
end
```

Considering the **cost_function.f90** program the code for the calculation of rain-rate observations contribution the is reported below:

```
rrate_aero: if(cost_aero) then
nobs_ind=obs_aero_rrate_fin%nobs
print*,'rrate_aeromet',nobs_ind,nvar,nxa,nya,nxyza,nxya
allocate(ni_prime_qs(nobs_ind),v1_qs(nobs_ind),   &
         v1_qs_aux(nobs_ind))
ival_qs=0
do iiw=1,nobs_ind
ival_qs=nint(obs_aero_rrate_fin%x_pos(iiw))+(nxa)*nint(obs_aero_rrate_fin%y_pos(iiw))
+nxya*nint(obs_aero_rrate_fin%z_pos(iiw))    ! position of the observation in the vector space.
ni_prime_qs(iiw)=ni_aux(ival_qs)
v1_qs(iiw)=obs_aero_rrate_fin%qs(iiw)-ni_prime_qs(iiw)
enddo
v1_qs_aux=matmul(obs_aero_rrate_fin%erre_qs_1,v1_qs).    !R-1*(yo'-UzUyUx ni)
aux_val=0.
do iiw=1,nobs_ind
aux_val=aux_val+v1_qs_aux(iiw)*v1_qs(iiw)                ! (yo'-UzUyUx ni)T R-1 (yo'-UzUyUx ni)
enddo
cost20=cost20+aux_val
print*,cost20
deallocate(ni_prime_qs,v1_qs,v1_qs_aux)
endif rrate_aero
```

where the most important passages are commented in red. In the computation of the ival_qs index, recall that this scheme assimilates the water vapor mixing ratio only, which is the first variable of the state vector **x** of the 3DVar code. The logical variable cost_aero is activated in the program **elab_sat_rate_aero.f90**, if there are observations to assimilate. In addition, the application of the transform U in the three spatial directions (U_z, U_y, U_z) is done in the first lines of the **cost_function.f90** program and it is applied to the while state vectors **ni**.
The code in the computation of the gradient is detailed below with the most important part of the code commented in red:

```
aero_rrate: if(cost_aero) then.          ! the logical variable cost_aero is activated in the
                                         ! elab_sat_rate_aero.f90 if there are rain rates to
                                         ! assimilate

nobs_ind=obs_aero_rrate_fin%nobs
print*,'AEROMET RRATE GRADIENT: ',nobs_ind,nvar,nxa,nya
allocate(ni_prime_qs(nobs_ind),v1_qs(nobs_ind),v2_qs(nobs_ind))
ival_qs=0
do iiw=1,nobs_ind
ival_qs=nint(obs_aero_rrate_fin%x_pos(iiw))+(nxa)*nint(obs_aero_rrate_fin%y_pos(iiw))
+nxya*nint(obs_aero_rrate_fin%z_pos(iiw))
ni_prime_qs(iiw)=ni_aux(ival_qs)                          ! Take the values of ni at the right
                                                          !positions
v1_qs(iiw)=obs_aero_rrate_fin%qs(iiw)-ni_prime_qs(iiw)    ! (yo'-ni)
enddo
v2_qs=matmul(obs_aero_rrate_fin%erre_qs_1,v1_qs)          ! R-1(yo'-ni)

aux_3d_aero_rr(:)=0.
do iiw=1,nobs_ind
  ival_qs=nint(obs_aero_rrate_fin%x_pos(iiw))+(nxa)*nint(obs_aero_rrate_fin%y_pos(iiw))
+nxya*nint(obs_aero_rrate_fin%z_pos(iiw))
aux_3d_aero_rr(ival_qs)=v2_qs(iiw)
```

```
enddo

aux_3d1_aero_rr(:)=0.
aux_1d(:)=0.

                                                    ! Transpose operators
ivar=1                                              ! water vapour mixing ratio only
do j=1,nya
do i=1,nxa
ni_z(:)=0.
do k=1,nza
ipts=(ivar-1)*nxyza+(k-1)*nxya+(j-1)*nxa+i
ni_z(k)=aux_3d_aero_rr(ipts)                         ! link with previous computation
enddo
if(ivar .eq. 1) then
aux_1d=matmul(transpose(uz_rv),ni_z)
else if (ivar .eq. 2) then
aux_1d=matmul(transpose(uz_temp),ni_z)
else if (ivar .eq. 3) then
aux_1d=matmul(transpose(uz_rain),ni_z)
else
print*,'Error in the variable selection.'
print*,'Stop in dfunc in the transport computation. AERI_RRATE.',ivar,nvar
stop
endif
do k=1,nza
ipts=(ivar-1)*nxyza+(k-1)*nxya+(j-1)*nxa+i
aux_3d1_aero_rr(ipts)=aux_1d(k)                          ! Uz JT HT R-1(yo'-ni)
!print*,i,j,k,ipts,aux_3d1(ipts)                        !  In this case the Jacobian of the
                                                        ! transform is the identity matrix
enddo
enddo
enddo

ivar=1         ! solo mixing ratio vapor
do k=1,nza
uhy=uhy_var(:,:,k,ivar)
do i=1,nxa
aux_1dy(:)=0.
aux_1dy_1(:)=0.
do j=1,nya
ipts=(ivar-1)*nxyza+(k-1)*nxya+(j-1)*nxa+i
aux_1dy(j)=aux_3d1_aero_rr(ipts)
enddo
aux_1dy_1=matmul(transpose(uhy),aux_1dy)
do j=1,nya
ipts=(ivar-1)*nxyza+(k-1)*nxya+(j-1)*nxa+i
aux_3d1_aero_rr(ipts)=aux_1dy_1(j)                  ! Uy Uz JT HT R-1*(yo'-ni)
enddo
enddo
enddo

ivar=1          ! solo mixing ratio vapor
do k=1,nza
uhx=uhx_var(:,:,k,ivar)
do j=1,nya
aux_1dx(:)=0.
aux_1dx_1(:)=0.
i1=(ivar-1)*nxyza+(k-1)*nxya+(j-1)*nxa
aux_1dx(1:nxa)=aux_3d1_aero_rr(i1+1:i1+nxa)
aux_1dx_1=matmul(transpose(uhx),aux_1dx)
aux_3d1_aero_rr(i1+1:i1+nxa)=aux_1dx_1(1:nxa)        ! Ux Uy Uz JT HT R-1*(yo'-ni)
enddo
enddo

deallocate(ni_prime_qs,v1_qs,v2_qs)

endif aero_rrate
```

5. The background error matrix

The formulation of the background error matrix closely follows the method of Barker et al. (2003). First the background error matrix is decomposed in the x,y and z directions. The background error matrix in the x and y direction is a decorrelation error matrix whose length scale are a function of the height. So, you need to provide these length-scales. These length-scales and the computation of the background error matrix in the x and y direction is provided in the subroutine

calcola_bckg_xy in the **w3dvar_node.f90** program. Few IDL programs are provided to compute the length-scales from the WRF model output via the NMC method. These programs are in the **inmc** directory are refer to the control variables used by the software (with the exception of the rain control variable for which a 10 km length-scale is used). These programs work with the WRF model and must be adapted for your needs. Note that length-scales coming from this software can be large and this can have a negative impact on the analysis; decrease them if necessary. This is dependent on the problem considered and trying different factor of reduction of the length-scales can be useful for the optimal setting of the code.
An eigenvalue-eigenvector decomposition is performed in the vertical direction. The outcome of this decomposition is passed to the *calcola_bckgz_new* subroutine of the **w3dvar_node.f90** program. Here you can also find some simple description of the background error matrices that can be overwritten by using the output of the *nmc_z.pro* program. This program is also written in IDL and can be also found in the **inmc** directory. Finally, a program to work with ensembles *nmc_z_ens.pro* is provided as an example of applying the software to determine the background error matrix in the vertical direction from an ensemble.
It is noted that while the background error matrices in the x and y directions are decorrelation error matrices, the error in the variable (specifies as a function of the height) is contained in the formulation of the vertical component of the background error matrix.

6. Convergence of the cost-function

The minimisation of the cost-function is implemented through the conjugate-gradient method using the routines of Press et al. (1997). The routines for the minimisation can be found in the **minimization_routines.f90** program in the **src** directory of the **w3dvar_par_node directory**. The cost-function and the gradient of the cost-function are provided in the **cost_function.f90** code in the **src** subdirectory of the **w3dvar_par_node** directory. Sometimes, the convergence of the minimization algorithm requires many iterations taking a long time. Based on the previous experience using the code, 15 iterations with the computation of the gradient of the cost function are enough for good results. Stated in other terms, the main program will produce an output after 15 computations of the cost-function gradient. If you want to change this behaviour go in the **minimization_routines.f90** program in the **src** subdirectory of the **w3dvar_node** directory and search for the string *NITER.* You will find NITER=15 and you can change the number of maximum iterations (15) to whatever number.

7. Software utilization and publications

The 3DVar software has been used in several papers. In addition to Federico (2013), in which the general physical basis of the software is explained, the following papers used this 3DVar applied to the assimilation of different observations:

- lightning data (Torcasio et al., 2023),
- radar reflectivity from ground-based radar (Federico et al., 2019; Federico et al., 2021; Avolio et al., 2025) and from satellites (Marra et al., 2019),
- zenith total delay (Mascitelli et al., 2019),
- and satellite-derived rain rate data (Federico et al., 2022; Torcasio et al., 2024).

In addition, the 3DVar software was used with both RAMS@ISAC and WRF numerical weather prediction (NWP) models. The software should work, in principle, with other NWP models, by providing the interface.
In these publications you will also find information about the software development and the expression of forward operators.

8. Questions and updates

Questions about the 3DVar software can be addressed to s.federico@isac.cnr.it or c.transerici@isac.cnr.it or rc.torcasio@isac.cnr.it. Errors should also be addressed to the above e-mails. The 3DVar software will be updated in the following years and a new release is expected

every one year. A web page, dedicated to the software, is available at the address: 150.146.138.33. Go there and click the link to be directed to the right page. You will also find simple routines for computing the vertical background error matrix and the de-correlation length-scales in the horizontal plane.
Finally, we plan to update the comments into the software and to express them in English and to enlarge the documentation of the software. Go to the 3Dvar software web page to stay updated.

9. WRF patches

The 3DVar code is general and can run, in principle, with any NWP model. We tested the 3DVar code with the RAMS@ISAC (Regional Atmospheric Modeling System at the Institute of Atmospheric Sciences of the National research Council) model and with the WRF model. We provide the patches for using the 3DVar code with WRF model version 4.1 for the thermodynamical variables: temperature, water vapor mixing ratio and rain mixing ratio. For the zonal and meridional wind components another behavior is needed (yet not included in this manual). The patches are included in the file WRF_patches.zip file provided with the 3DVar. They include the following files:

./Registry/Registry.EM_COMMON
./dyn_em/solve_em.F
./phys/module_microphysics_driver.F
./phys/module_mp_nudge_light.F
./run/namelist.input

After adding these patches to your version of the WRF model, the model must be cleaned (./clean -a command in the WRF directory) and compiled again as the Registry.EM_COMMON file is changed.
The ***out_1d+3dvar_YYYYMMDDHHMinMin.dat*** file is read by the program **solve_em.F** in the directory **./dyn_em/**. By default, the WRF looks to the existence of a 3DVar file every 30 minutes for the first six hours of run. To change for this behavior search for the following lines in ./dyn_em/solve_em.F file:

```
   flname_3dv='out_1d+3dvar_'//current_time_14(1:12)//'_f.dat'

   iflag_3dvar=0
  inquire(file=trim(grid%path_to_files)//trim(flname_3dv),exist=esiste)
       if(esiste .and. grid%assim_3dvar .eq. 1 .and. (current_time_14(11:14) .eq. '0000'  &
       .or. current_time_14(11:14) .eq. '3000') .and. curr_secs .le. 6.*3600.) then !the 3DVar file
is checked for the first 6 hours, and WRF checks for it at 00 and 30 minutes in these 6 hours.
       print*,'Leggo il file: ',trim(grid%path_to_files)//trim(flname_3dv)
       print*,'curr_secs....',curr_secs
       iflag_3dvar=1
       OPEN ( UNIT=14, FILE=trim(grid%path_to_files)//trim(flname_3dv),  &
       FORM='formatted', STATUS='old' )

       read(14,*) nxa
33     read(14,*) nya
34     read(14,*) nza
```

…… the reading of the file continues.

then, by changing the numbers above in red, you can custom the behaviors for your needs. Of course, after changing the file, the WRF model must be recompiled (in this case without cleaning).
The **./phys/module_microphysics_driver.F** file is modified to call the **./phys/module_mp_nudge_light.F** with the right arguments. The **module_mp_nudge_light.F** program does the 3DVar data assimilation by changing the temperature, water vapor mixing ratio and rain mixing ratio with the values provided by the 3DVar code in the ***out_1d+3dvar_YYYYMMDDHHMinMin.dat*** file. This is made in the subroutine r3dvar of the program **./phys/module_mp_nudge_light.F**. The call to the subroutine is as follows:

```
SUBROUTINE r3dvar(nx,ny,nz,qg,qv,rv_3d,temp,press,dz,dx,dy,dt)
```

After customizing and compiling the code, the only thing to do is to add the output file generated by the 3DVar, i.e. ***out_1d+3dvar_YYYYMMDDHHMinMin.dat***, into the directory where WRF expects the file. This directory is specified in the namelist.input file in the variable **path_to_file**:

path_to_files = '/Users/stefano/dati_light/'

Note also the the assim_3dvar variable in the namelist.input file must be set to 1 to activate the reading of the 3DVar file. In the namelist.input file set:

assim_3dvar = 1,

We recall that if the 3DVar is activated and the file is not found by the model at the runtime in the expected directory, the 3DVar is skipped and the WRF run continues without data assimilation by this package. A message is written in the rsl.out.0000 file showing if the file was found or not at the runtime.
We note also, that the patches added to the WRF model and discussed in this section do also the lightning data assimilation through the nudging. In this case the variables in the namelist that manage the assimilation are:

```
nudge_lightning                     = 0,0,0,
nudge_light_times                   = 0,
nudge_light_timee                   = 21600,
nudge_light_int                     = 900,
path_to_files                       = '/Users/stefano/dati_light/'
```

The variable **nudge_lightning** activates the lightning nudging if set to 1; if set to 0 the lightning data assimilation via nudging is deactivated. Note that this variable is a function of the domains set in the simulation. If you want to assimilate lightning only in the first domain and you are running two domains, set **nudge_lightning= 1,0,** . If you want to assimilate lightning only in the second domain and you are running two domains, set **nudge_lightning= 0,1,**. If you want to assimilate lightning in both domains and you are running two domains, set **nudge_lightning= 1,1**.
The variable **nudge_light_times** indicates the starting time of the lightning data assimilation, while the variable **nudge_light_timee** indicates when the lightning data assimilation must stop. The variable **nudge_light_int** sets the time interval between two lightning file. This time interval correspond to the time interval over which the lightning data are accumulated in each input file.
The **path_to_files** variable tells where the files with the lightning are located. This path is shared with the 3DVar data assimilation of this manual.

10. 3DVar Public Domain Notice

The 3DVar software was developed by Dr. Stefano Federico, Dr. Rosa Claudia Torcasio and Mr. Claudio Transerici. No proprietary rights are claimed, either statutory or otherwise, to this version and release of the software and consider the 3DVar software to be in the public domain for use by any person or entity for any purpose without any fee or charge. 3DVar is provided on an "AS IS" basis and any warranties, either express or implied, including but not limited to implied warranties of non-infringement, originality, merchantability and fitness for a particular purpose, are disclaimed. In no event shall the authors be liable for any damages, whatsoever, whether direct, indirect, consequential or special, that arise out of or in connection with the access, use or performance of the 3DVar software, including infringement actions.

References

Avolio E., Castorina G. Torcasio, R.C., Federico S.: A multi hazard extreme weather event in Southern Italy: Assessment and sensitivity tests of the WRF model, Atmos. Res., 315, 2025. https://doi.org/10.1016/j.atmosres.2024.107827.

Barker, D. M., Huang, W., Guo, Y.-R., and Bourgeois, A.: Athree- dimensional variational (3DVAR) data assimilation system for use with MM5. NCAR Tech. Note. NCAR/TN-453 1 STR, avail- able from UCAR Communications, P.O. Box 3000,Boulder, CO 80307, 68 pp., 2003.

Barker, D. M., Huang, W., Guo, Y.-R., and Xiao, Q. N.: A Three- Dimensional Variational Data Assimilation System For MM5: Implementation And Initial Results, Mon. Weather Rev., 132, 897–914, 2004.

Federico, S.; Torcasio, R.C.; Mascitelli, A.; Del Frate, F.; Dietrich, S. Preliminary Results of the AEROMET Project on the Assimilation of the Rain-Rate from Satellite Observations. In Computational Science and Its Applications—ICCSA 2022 Workshops; Gervasi, O., Murgante, B., Misra, S., Rocha, A.M.A.C., Garau, C., Eds.; ICCSA 2022, Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2022; Volume 13380.

Federico, S.; Torcasio, R.C.; Puca, S.; Vulpiani, G.; Comellas Prat, A.; Dietrich, S.; Avolio, E. Impact of Radar Reflectivity and Lightning Data Assimilation on the Rain- fall Forecast and Predictability of a Summer Convective Thunderstorm in Southern Italy. Atmosphere 2021, 12, 958.

Federico, S.; Torcasio, R.C.; Avolio, E.; Caumont, O.; Montopoli, M.; Baldini, L.; Vulpiani, G.; Dietrich, S. The impact of lightning and radar reflectivity factor data assimilation on the very short-term rainfall forecasts of RAMS@ISAC: Application to two case studies in Italy. Nat. Hazards Earth Syst. Sci. 2019, 19, 1839–1864.

Federico, S.: Implementation of a 3D-Var system for atmospheric profiling data assimilation into the RAMS model: initial results, Atmos. Meas. Tech., 6, 3563–3576, https://doi.org/10.5194/amt-6-3563-2013, 2013.

Marra, A.C.; Federico, S.; Montopoli, M.; Avolio, E.; Baldini, L.; Casella, D.; D'Adderio, L.P.; Dietrich, S.; Sanò, P.; Torcasio, R.C.; et al. The Precipitation Structure of the Mediterranean Tropical-Like Cyclone Numa: Analysis of GPM Observations and Numerical Weather Prediction Model Simulations. Remote Sens. 2019, 11, 1690.

Mascitelli, A.; Federico, S.; Fortunato, M.; Avolio, E.; Torcasio, R.C.; Realini, E.; Mazzoni, A.; Transerici, C.; Crespi, M.; Dietrich, S. Data assimilation of GNSS-ZTD into the RAMS model through 3D-Var: Preliminary results at the regional scale. Meas. Sci. Technol. 2019, 30, 055801.

Parrish, D. F. and Derber, J. C.: The National Meteorological Center's Spectral Statistical Interpolation analysis system, Mon. Weather Rev., 120, 1747–1763, 1992.

Press, W., Flannery B., Teukolsky S., and Vetterling, W.. Numerical Recipes in FORTRAN 77: The Art of Scientific Computing Cambridge University Press, 2 edition, (Sep 25, 1992)

Torcasio, R.C.; Papa, M.; Del Frate, F.; Dietrich, S.; Toffah, F.E.; Federico, S. Study of the Intense Meteorological Event Occurred in September 2022 over the Marche Region with WRF Model: Impact of Lightning Data Assimilation on Rainfall and Lightning Prediction. Atmosphere 2023, 14, 1152.

Torcasio, R.C.; Papa, M.; Del Frate, F.; Mascitelli, A.; Dietrich, S.; Panegrossi, G.; Federico, S. Data Assimilation of Satellite-Derived Rain Rates Estimated by Neural Network in Convective Environments: A Study over Italy. Remote Sens. 2024, 16, 1769. https://doi.org/10.3390/rs16101769