Report for IOMASA Deliverable 1.2.1:

Retrieval algorithm for total water vapour from AMSU-B data

Christian Melsheimer

IUP, University of Bremen

Contents

1 Basic Idea

The total water vapour algorithm relies on a satellite radiometer (SSM/T2, AMSU-B) measurement of the brightness temperature at three different frequencies ν_i, ν_j, ν_k at which the ground emissivity ε_s is similar but the water vapour absorption is different, $\kappa_i < \kappa_j < \kappa_k$ (κ_i is the water vapour mass absorption coefficient at frequency ν_i).
Then the following relation between the brightness temperatures at the three frequencies, $T_{b,i}$, $T_{b,j}$, $T_{b,k}$, and the total water vapour (TWV), W, can be derived [*Miao*, 1998; *Miao et al.*, 2001]

$$\log\eta_c = \ln\left(\frac{T_{b,i} - T_{b,j} - b_{ij}}{T_{b,j} - T_{b,k} - b_{jk}}\right) = c_0 + c_1 W \sec\theta \quad (1)$$

where θ is the sensor's viewing zenith angle, c_0 and c_1 are parameters depending on the water vapour absorption coefficients and the "bias" terms (b_{jk}, b_{ij}),

$$b_{ij} \approx \int_0^H \left[e^{\tau_i(z,H)\sec\theta} - e^{\tau_j(z,H)\sec\theta}\right] \frac{dT(z)}{dz}\, dz \quad (2)$$

contain the influence of the atmospheric temperature and water vapour profiles, which, however, is rather weak for typically low water vapour contents of the polar atmosphere.
Equation (

2 AMSU-B

AMSU (Advanced Microwave Sounding Unit) is a microwave radiometer on the new generation polar orbiting satellites of NOAA (National Oceanic and Atmospheric Administration), NOAA-15, NOAA-16 and NOAA-17. AMSU consists of two modules, AMSU-A and AMSU-B, which are very similar to the SSM/T1 and T2 (Special Sensor Microwave) on the DMSP (Defense Meteorological Satellite Program) satellites, but have a much better spatial resolution. The spatial resolution at nadir of AMSU-B is about 16 km, the swath width is about 2068 km.
AMSU-B has five channels in the range 89–182 GHz. Details are in the following table.

Table 1: The channels of AMSU-B ordered by increasing water vapour absorption coefficient.

Freq. [GHz]	89.0 (91.655)	150.0	183.31±7	183.31±3	183.31±1
AMSU-B Band	16	17	20	19	18
practical label	1	2	3	4	5

Channels 16 (89 GHz) and 17 (150 GHz) are window channels, the remaining three, 18, 19, and 20, are centred around the strong water vapour absorption line at 183.31 GHz, channel 18 being most, channel 20 least sensitive to water vapour. Note that we preferably label the five bands by the numbers 1 to 5, with increasing water vapour absorption. This labeling is identical to the names of the corresponding SSM/T2 bands.

3 Calibration of the Algorithm

The calibration of the algorithm, as mentioned above, needs simulated AMSU-B brightness temperatures and radiosonde profiles (temperature, humidity). For the simulations, we used the radiative transfer model ARTS (Atmospheric Radiative Transfer System), a modular, customizable program that has been (and still is) developed at IUP (Atmospheric Sounding Group). ARTS here uses the Rosenkranz absorption model (continuum and lines) for H_2O, O_2, N_2 (but can also use all other standard absorption models).
The calibration procedure does the following steps

1. - Use radiosonde profiles, integrate TWV from them
   - Simulate AMSU-B brightness temperatures T_1, T_2, T_3, T_4, T_5 (where the numbers denote the following frequencies: 89.0, 150.0, 183.3±1, 183.3±3, 183.3±7) for a range of ground emissivities ε_s for each radiosonde profile ($0.6 \leq \varepsilon_s \leq 1.0$ in 11 equidistant steps).
2. Select three channels i, j, k
3. Linear fit of ΔT_{ij} vs. ΔT_{jk} for each radiosonde profile (11 different emissivities), where $\Delta T_{ij} = T_{b,i} - T_{b,j}$
4. Find focal point $(F_{jk}, F_{ij}) = (\overline{b_{jk}}, \overline{b_{ij}})$ as point of least square distance from all fitted lines
5. Linear fit of $TWV \sec\theta$ vs. $\ln\eta_c = \ln\left(\frac{\Delta T_{ij} - F_{ij}}{\Delta T_{jk} - F_{jk}}\right)$ yields the required calibration constants C_0, C_1 for the retrieval algorithm:

$$TWV \sec\theta = C_0 + C_1 \ln\eta_c \tag{3}$$

The four calibration parameters were determined separately for different zenith angles θ.
By selecting different frequency triples (i, j, k) and different subsets of radiosonde profiles, the algorithm is adapted to different ranges of TWV:

- Low TWV ($<$ 1.5 kg/m^2): $(i, j, k) = (3, 4, 5)$
- Medium TWV (1.5 kg/m^2 $< TWV <$6 kg/m^2): $(i, j, k) = (2, 3, 4)$

The radiosonde profiles used were from WMO (World Meteorological Organization) radiosonde stations from the years 1996 until 2002. Only arctic stations close to the coast were used. For a list of the stations see appendix
The calibration parameter sets thus derived are listed in appendix

4 Total water vapour algorithm

The core of the algorithm is of course (
The decision which channels to use is decided by testing the signs of the numerator and denominator of the argument of the logarithm: They have both to be negative. This means that the condition

$$T_4 - T_5 < F_{4,5} \tag{4}$$

has to be met for the algorithm to be applied to channels 3, 4, and 5; otherwise, channels 2, 3, and 4 will be used. The algorithm cannot yield results any more if

$$T_3 - T_3 < F_{3,4} \tag{5}$$

In this case, the algorithm is set to yield the value “not a number” (NaN) which represents a TWV value above about 7 kg/m^2.
The IDL implementation of the algorithm and of the subroutine that reads the calibration parameters files can be found in appendix

A Arctic coastal radiosonde stations

```
**************************************************************************************************************
* WMO STATIONS NORTH OF 60 DEG NORTH LOCATED ON THE COAST FOR WHICH WE HAVE RADIOSONDE PROFILES (FROM DMI)*
**************************************************************************************************************

                  SVALBARD
                  ========
01004      7855'00"N 1156'00"E    8.0 NY-LESUND II

         ICELAND
         ========
04018 BIKF 6358'00"N 2236'00"W   49.0 KEFLAVIKURFLUGVLLUR        W coast

         GREENLAND
         =========
04202 BGTL 7632'00"N 6845'00"W   77.0 PITUFFIK/THULE AB.          coast/fiord
04320 BGDH 7646'00"N 1840'00"W   11.0 DANMARKSHAVN                coast
04360 BGAM 6536'00"N 3738'00"W   50.0 TASIILAQ/AMMASSALIK         coast

         RUSSIA, SIBERIA
        ================
20046      8037'00"N 5803'00"E   22.0 POLARGMO IM.E.T. KRENKELJA  Franz-Josef-Land, Hall I.
20292      7743'00"N 10418'00"E  15.0 GMO IM.E.K. FEDOROVA        Cape Tshelyuskin
20744      7222'00"N 5242'00"E   15.0 MALYE KARMAKULY             Nov. Zeml., SE coast
21504      7440'00"N 11256'00"E  57.0 OSTROV PREOBRAZENIJA        Khatanga estuary
21824      7135'00"N 12855'00"E   7.0 TIKSI                       coast, near mouth of Lena
22113      6858'00"N 3303'00"E   51.0 MURMANSK                    fiord
22217      6709'00"N 3221'00"E   25.0 KANDALAKSA                  sound of White Sea
22271      6753'00"N 4408'00"E    8.0 S(h)OJNA                    coast, mouth of White Sea/Arctic O.
22522      6459'00"N 3448'00"E    8.0 KEM'-PORT                   White Sea, coast
22550      6430'00"N 4044'00"E    8.0 ARHANGEL'SK- russia         White Sea, coast
23022 ULDD 6945'00"N 6142'00"E   49.0 AMDERMA                     Kara Sea, coast
25399      6610'00"N 16950'00"W   3.0 MYS UELEN                   Bering Strait, coast
25954      6021'00"N 16600'00"E   4.0 KORF                        Kamchatka, Coast

                ALASKA
               ========
70026 PABR 7118'00"N 15647'00"W  13.0 BARROW/W. POST W. ROGERS   N coast
70133 PAOT 6652'00"N 16238'00"W   3.0 KOTZEBUE, RALPH WIEN        sound, N Bering Strait
```

```
70200 PAOM 6430’00"N 16526’00"W   11.0 NOME                       S Bering Strait, coast

               CANADA
               ======
71072 CYMD 7614’00"N 11920’00"W   15.0 MOULD BAY - nt             W Arctic Archipel., coast
71081 CYUX 6847’00"N 8115’00"W    8.0 HALL BEACH - nu             Fox Basin, near Baffin I., coast
71082 CWLT 8231’00"N 6217’00"W   30.0 ALERT - nt                  Ellesmere I., N coast
71909 CYFB 6345’00"N 6833’00"W   34.0 IQALUIT - nu                bay, Baffin I.
71915 CYZS 6412’00"N 8322’00"W   64.0 CORAL HARBOUR - nt          N Hudson Bay, coast
71917 CWEU 7959’00"N 8556’00"W   10.0 EUREKA - nu                 fiord, Ellesmere I.
71924 CYRB 7443’00"N 9459’00"W   67.0 RESOLUTE BAY - nt           central Arct. Archip.. coast
71925 CYCB 6906’00"N 10508’00"W   27.0 CAMBRIGDE BAY - nt         S Arct. Archip., coast
```

B Calibration constants for the Arctic

B.1 Channels 3 (AMSU-B 20), 4 (AMSU-B 19), and 5 (AMSU-B 18) – Low TWV

```
# Calibration constants for TWV algorithm (per theta)
# for TWV range:      0.00000      1.80000
# and channels [i,j,k]:       3       4       5
# derived from simulated AMSU-B Tbs in /platte1/melsheim/twv-algo/result-
# number of thetas
# theta      C0             C1             px             py
          15
  1.667  5.7769835e-01  1.0241828e+00  4.8233519e+00  4.6147237e+00
  5.000  5.7787997e-01  1.0230488e+00  4.8368535e+00  4.6402364e+00
  8.333  5.7655001e-01  1.0231881e+00  4.8638391e+00  4.6919088e+00
 11.667  5.7526100e-01  1.0224001e+00  4.9042516e+00  4.7698755e+00
 15.000  5.7284611e-01  1.0220487e+00  4.9588661e+00  4.8770490e+00
 18.333  5.7066375e-01  1.0198721e+00  5.0289974e+00  5.0183492e+00
 21.667  5.6816792e-01  1.0159817e+00  5.1080680e+00  5.1865354e+00
 25.000  5.6442446e-01  1.0123206e+00  5.1928587e+00  5.3774190e+00
 28.333  5.6044602e-01  1.0047246e+00  5.2495990e+00  5.5340323e+00
 31.667  5.5856198e-01  9.9057138e-01  5.2545843e+00  5.6040668e+00
 35.000  5.5857372e-01  9.6918279e-01  5.1741438e+00  5.5097914e+00
 38.333  5.6449759e-01  9.3354315e-01  4.9440069e+00  5.1048703e+00
 41.667  5.6612736e-01  8.9602447e-01  4.6950302e+00  4.6297731e+00
 45.000  5.7348484e-01  8.3936876e-01  4.3526554e+00  3.9311612e+00
 48.333  5.7661337e-01  7.7232462e-01  4.0537467e+00  3.3229454e+00
```

B.2 Channels 2 (AMSU-B 17), 3 (AMSU-B 20), and 4 (AMSU-B 19) – Higher TWV

```
# Calibration constants for TWV algorithm (per theta)
# for TWV range:      0.00000      1.80000
# and channels [i,j,k]:       3       4       5
# derived from simulated AMSU-B Tbs in /platte1/melsheim/twv-algo/result-
# number of thetas
# theta      C0             C1             px             py
          15
  1.667  5.7769835e-01  1.0241828e+00  4.8233519e+00  4.6147237e+00
  5.000  5.7787997e-01  1.0230488e+00  4.8368535e+00  4.6402364e+00
  8.333  5.7655001e-01  1.0231881e+00  4.8638391e+00  4.6919088e+00
 11.667  5.7526100e-01  1.0224001e+00  4.9042516e+00  4.7698755e+00
 15.000  5.7284611e-01  1.0220487e+00  4.9588661e+00  4.8770490e+00
 18.333  5.7066375e-01  1.0198721e+00  5.0289974e+00  5.0183492e+00
```

21.667	5.6816792e-01	1.0159817e+00	5.1080680e+00	5.1865354e+00
25.000	5.6442446e-01	1.0123206e+00	5.1928587e+00	5.3774190e+00
28.333	5.6044602e-01	1.0047246e+00	5.2495990e+00	5.5340323e+00
31.667	5.5856198e-01	9.9057138e-01	5.2545843e+00	5.6040668e+00
35.000	5.5857372e-01	9.6918279e-01	5.1741438e+00	5.5097914e+00
38.333	5.6449759e-01	9.3354315e-01	4.9440069e+00	5.1048703e+00
41.667	5.6612736e-01	8.9602447e-01	4.6950302e+00	4.6297731e+00
45.000	5.7348484e-01	8.3936876e-01	4.3526554e+00	3.9311612e+00
48.333	5.7661337e-01	7.7232462e-01	4.0537467e+00	3.3229454e+00

C TWV algorithm (IDL)

```
pro read_calfile, calfile,scanangle,C0,C1,Fij,Fjk,RF,debug=debug,paramplot=paramplot
;+
;NAME:
;   read_calfile
;PURPOSE:
;   routine needed internally by twv_algo in order to read the file
;   with the TWV algorithm calibration parameters, and to interpolate
;   to the actual scanangles if necessary
;USAGE:
; read_calfile, calfile,scanangle,C0,C1,Fij,Fjk [,RF]
;INPUT:
;  calfile: name (string) of the file with the calibration parameters
;           (see twv_algo)
;  scanangle: array (float) of the scan angles of the instrument, in
;          degrees. If this is different from the incidence angles of
;          the calibration parameters, the latter are interpolated to
;          the scanangle grid.
; OUTPUT:
;  C0,C1,Fij,Fjk: named variables that will contain the arrays (one
;  value for each scan angle) of the calibration parameters
; OPTIONAL:
;  RF: output unit number for error reporting
; KEYWORDS:
; debug: as in twv_algo
; paramplot: if set, plot the calibration paramters versus scan angle
;HISTORY:
; written by C. Melsheimer, 28 April, 2004
; 2004-06-07 fixed plotting of calibration parameters in read_calfile
;
;-

if n_params() lt 7 then RF = -2 ; reporting output to stderr
if not keyword_set(debug) then debug = 0

;open parameter file
openr,PF,calfile,/get_lun

;initialize variables
n_za=0
commentsym = '#'
line = commentsym
while (strmid(line, 0, 1) eq commentsym) do begin
    readf, PF, line & if debug ge 3 then printf,RF,'"'+line+'"'
endwhile
parts = strsplit(line, /extract)
reads,parts[0],n_za &  if debug ge 2 then printf,RF,format='("n_za=",i)',n_za

za = fltarr(n_za)
```

```
Fij = fltarr(n_za)
Fjk = fltarr(n_za)
C0 = fltarr(n_za)
C1 = fltarr(n_za)

for i = 0,n_za-1 do begin
    readf, PF, line & if debug ge 3 then printf,RF,'"'+line+'"'
    parts = strsplit(line, /extract)
    za[i] = float(parts[0]) & if debug ge 3 then printf, RF, format='("za=",f7.3)',za[i]
    C0[i] = float(parts[1]) & if debug ge 3 then printf, RF, format='("C0=",e)',za[i]
    C1[i] = float(parts[2]) & if debug ge 3 then printf, RF, format='("C1=",e)',za[i]
    Fjk[i] = float(parts[3]) & if debug ge 3 then printf, RF, format='("Fjk=",e)',za[i]
    Fij[i] = float(parts[4]) & if debug ge 3 then printf, RF, format='("Fij=",e)',za[i]
endfor
free_lun,PF

;check:
if not array_equal(za, scanangle) then begin
    if debug ge 1 then printf, RF, 'C0 before interp.:',C0
    if debug ge 1 then printf, RF, 'C1 before interp.:',C1
    if debug ge 1 then printf, RF, 'Fjk before interp.:',Fjk
    if debug ge 1 then printf, RF, 'Fij before interp.:',Fij
;interpolate to Amsu-B.
    C0 = interpol(C0,za,scanangle)
    C1 = interpol(C1,za,scanangle)
    Fjk = interpol(Fjk,za,scanangle)
    Fij = interpol(Fij,za,scanangle)
    if debug ge 1 then printf, RF, 'C0 after interp.:',C0
    if debug ge 1 then printf, RF, 'C1 after interp.:',C1
    if debug ge 1 then printf, RF, 'Fjk after interp.:',Fjk
    if debug ge 1 then printf, RF, 'Fij after interp.:',Fij
endif

;plotting calibration parameters as function of angle
if keyword_set(paramplot) then begin
    param = [[C0],[C1],[Fij],[Fjk]]
    titlestring = ["CO","C1","Fij","Fjk"]
    addtitlestring = ["constant","linear in log(RCBT)","focal point y coord","focal point x coord"]
    unitstring = ["kg/m2","kg/m2","K","K"]
    device,retain=2
    for n=0,3 do begin
        window, n, retain=2, title=titlestring[n]+' (' + addtitlestring[n] + ')'
        plot,scanangle,param[*,n],linestyle=n,title=addtitlestring[n],$
             xtitle="angle [deg]",ytitle=titlestring[n] + ' ['+unitstring[n] + ']'
    endfor
endif
end                             ; of read_calfile
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
pro twv_algo, calfilebase, ch_2, ch_3, ch_4, ch_5, scanpos, twv,qqq,$; use_234,use_345,$
        debug=debug, reportfile=reportfile,use1234=use1234,$
        amsub=amsub, scanangle=scanangle,sat_eps=sat_eps, check=check,$
        subranges345=subranges345, subranges234=subranges234,$
         switch_twv=switch_twv
;+
;NAME:
;          twv_algo
;PURPOSE:
;calculate total water vapour (TWV) from three brightness
;temperatures given in three arrays, near the 182 GHz water vapour line
;using J.Miaos water vapour algorithm (for AMSU-B, SSM-T2), result is
;put into an array of the same dimension as the brightness temperature
;arrays
;
; numbering of AMSU-B (or other) Channels here:
; 1    2         3          4      5
;89   150    181.31+/-7   +/-3   +/-1  [GHz]
; 16   17       20         19     18   (officially AMSU)
;
;USAGE:
;    twv_algo, calfilebase, ch_2, ch_3, ch_4, ch_5, scanpos, twv
;
;INPUT:
; calfilebase:
;            base name of files (string) containing the calibration constants for the
;             algorithm, for channels 2,3,4 and 3,4,5, ending 'cal345.txt', 'cal234.txt', and
;             'cal345-01.txt', 'cal345-02.txt' etc. for the second pass calibration constants will be assumed.
;             Format of that file exactly as the output calfile from make_twv_algo.pro
; ch_2, ch_3, ch_4, ch_5:
;             The brightness temperatures (1-dim. float arrays)
;             measured in the four channels.
;             Dimension: number of scan positions per line * number of scan lines
;    scanpos: array containing the scan position index for each data point,
;             (should be of same size as ch_2 etc.) for AMSU-B,
;               0.,1., ... 44.
;OUTPUT:
;       twv:  The resulting TWV in an array of the same dimensions as
;             ch_2 etc.
;       qqq:  some dummy variable for testing, might be set to ch_23
;             or use_234 or anything in program text
;KEYWORDS:
;            debug: debugging level [default: 0],  can be 0, 1, 2, 3
;       reportfile: path/name (string) of the file for error and diagnostic
;                   output [default: standard error, i.e. the idl
;                   console]
;        scanangle: array containing the actual scan angles (in degrees,
;                   down-looking is 0.0) corresponding to the scan position
;                   indices from scanpos. If not set, AMSU-B scan
;                   angles are taken automatically.
```

```
;           /amsub: take AMSU-B scan angles
;           /check: if set, do not really calculate TWV, just set TWV in areas where
;                   the 345 algorithm would be applied to 1.0 and in areas where
;                   234 algorithm would be applied to 5.0
;        no_result: twv is will be set to this value where it cannot be
;                   retrieved because both algorithms are saturated
;                   [default: !values.f_nan]
;          sat_eps: Set global threshold for saturation,
;                   i.e. algorithm is saturated if a Tb difference is greater
;                   than sat_eps. If not set or set to -99, the 4 focal point
;                   coordinates are used.
;       switch_twv: TWV at which to switch from 345 to 234 algorithm
;       subranges345,
;       subranges234: arrays giving the subranges for the calibration
;                   parameters for the second pass.
;                   subranges345=[0.0,1.0,1.5] means that for
;                   channel345, 0.0..1.0 and 1.0..1.5. If not set or set to 0, no
;                   second pass is done
;       /use1234:  if set, use channels 1, 2 and 3 (AMSU 16, 17, 20) -
;                  highly experimental! Then, the command line arguments called
;                  ch_2, ch_3, ch_4, ch_5 will actually have to hold ch_1, ch_2,
;                  ch_3, ch_4 !
;TO DO:
;
;
;HISTORY: Christian Melsheimer, 8 Mar 2004, inspired by program by Jungang
;26 Apr - 4 May 2004: heavy development and testing.
;   7 - 10 June 2004: - modified saturation cutoff: now uses the focal point coordinates.
;                     - implemented recalculating TWV using different sets of calibration
;                       paramters depending on the value of TWV from the first calculation.
;       12 Nov, 2004: new keyword use1234 to experimentally use channels 1,2,3 and 2,3,4 instead of  2,3,4 and 3,4,5.

;-

;---------------------------
;0. Checking input, keywords
;---------------------------


;set the filename and logical unit for diagnostic and error reporting
if n_elements(reportfile) eq 0 then begin
    RF=-2                       ;this is stderr
endif else begin
    openw,RF,string(reportfile),/get_lun
endelse

; set numbers to this value if they should later be filtered out:
; here: TWV cannot be retrieved because both algos are saturated
if n_elements(NO_RESULT) eq 0 then begin
    NO_RESULT = !values.f_nan   ;+99.
```

```
endif

; limit when a Tb difference is considered > 0 (i.e. saturation):
if n_elements(SAT_EPS) eq 0 then SAT_EPS = -99
if SAT_EPS eq -99 then begin
    printf,RF, "Using focal point coordinates as saturation threshold"
endif else begin
    printf,RF,"Using global saturation threshold SAT_EPS of", SAT_EPS
    SAT23_h = SAT_EPS
    SAT34_h = SAT_EPS
    SAT34_l = SAT_EPS
    SAT45_l = SAT_EPS
endelse

; TWV value where the two algorithms are switched
if n_elements(SWITCH_TWV) eq 0 then begin
    SWITCH_TWV = 1.5
endif

; set TWV to this value if it is NaN or infinite
if n_elements(UNDEF) eq 0 then begin
    UNDEF = -99.
endif

; set TWV to this value if it is retrieved as negative
if n_elements(NEG) eq 0 then begin
    NEG = -9.
endif

;use AMSU-B scan angles if nothing is given
if not keyword_set(scanangle) then amsub=1

if keyword_set(amsub) then begin
;create array of AMSU-B zenith angles:
    rtheta= 48.95D - 1.1D * dindgen(45)
    scanangle = fltarr(90)
    scanangle[0:44] = rtheta
    scanangle[45:89] = reverse(rtheta)
end

;number of different scan angles:
n_scan = (size(scanangle,/dim))[0]

;compare with scan angles from AMSU data.
;if debug ge 1 then printf, RF, format='("scan=",f,"deg ")',scan

;

;;;;OBSOLETE, since we alwayw use both algorithms!
;if keyword_set(which_channels) then which=which_channels else which = [3,4,5]
```

```
;printf,RF, ’Using channels’, which

if not keyword_set(debug) then debug = 0
;check array sizes
if debug ge 1 then print, ’size of ch_i:  ’, size(ch_i)
if  not ( array_equal(size(ch_i),size(ch_j) ) and  array_equal(size(ch_j),size(ch_k)))  then begin
    printf,RF,’ERROR! The 3 Tb arrays have different size or type!’
    printf,RF, ’Aborting’
    return
endif

;;FOLLOWING DOESNT WORK SINCE Tbs ETC. ARE IN 1-D ARRAY!
;check if number of scan angles matches one tb array dimension
;size_tb=size(ch_i,/dimensions)
;size_scan = size(scan, /dimensions)
;if size_tb[0] ne size_scan[0] then begin
;  printf,RF,’ERROR! Mismatch of number of scan angles with tb array dims!’
;  printf,RF, ’Aborting’
;  return
;endif

;n_scan = size_tb[0]
;n_lines = size_tb[1]
;if debug ge 2 then printf,RF,n_scan,n_lines;n_scaformat=’("n_scan=",
;twv = fltarr(n_scan,n_lines)


;-----------------------------
;1. reading calibration parameter files
;----------------------------
; parse subranges

; check if subranges are given
if n_elements(subranges345) eq 0 then subranges345 = 0 ; if keyword not present, set it to zero
if n_elements(subranges234) eq 0 then subranges234 = 0 ; if keyword not present, set it to zero

; put them into a structure
subranges = {a345:subranges345, a234: subranges234}
;structure for number of calibration pamameter sets: 1 for the first pass, and then as many for the second
; pass as there are subranges
n_calsets = {a345:0, a234:0}
;if keyword /use1234 set, use channels 234 and 123, else the normal
;345, 234 combination
if keyword_set(use1234) then aname = [’234’, ’123’] else aname = [’345’, ’234’]
for i = 0,1 do begin            ; i=0 is a345, i=1 is a234
    if size(subranges.(i), /n_dim) eq 0 then begin
        ;subranges is a scalar, i.e. no subranges
        printf, RF, "No subranges, just doing one pass"
        n_calsets.(i) = 1       ; just first pass, no subranges
    endif else begin
```

```
        n_calsets.(i) = n_elements(subranges.(i))
        ; because n numbers define n-1  subranges, plus set for the first pass
      ;check if there are at least 2 numbers in subranges
        if n_calsets.(i) le 1 then begin
            printf, RF,"WARNING: only one number in subranges"+aname[i]+"! Ignoring it (no 2nd pass)"
            ;return
        endif
;check that subranges is sorted
        if not array_equal(subranges.(i), subranges.(i)[sort(subranges.(i))]) then begin
            printf, RF, "ERROR: subranges"+aname[i]+" not sorted in ascending order! Aborting."
            return
        endif
    endelse
endfor

;initialize structure holding calfile names
calfile = {a345:strarr(n_calsets.a345), a234:strarr(n_calsets.a234)}

;initialize structures holding the four calibration parameters:
C0 = {a345:fltarr(n_calsets.a345,n_scan), a234:fltarr(n_calsets.a234,n_scan)}
C1 = {a345:fltarr(n_calsets.a345,n_scan), a234:fltarr(n_calsets.a234,n_scan)}
Fy = {a345:fltarr(n_calsets.a345,n_scan), a234:fltarr(n_calsets.a234,n_scan)}
Fx = {a345:fltarr(n_calsets.a345,n_scan), a234:fltarr(n_calsets.a234,n_scan)}

;construct calfile names and read them:
; calfilebase + "-cal345.txt" and calfilebase + "-cal234.txt"  for the first pass
; and calfilebase + "-cal345-01.txt",  calfilebase + "-cal345-02.txt" etc. for the first and second subrange etc.
;   of the second pass
;;;; ATTENTION: when using channels 1 to 4 (keyword /use1234), 234 has is replaced by
;;;; 123, and 345 by 234 (see definition of variable aname above)!
for i = 0,1 do begin            ; i=0 is a345, i=1 is a234
    nn=indgen(n_calsets.(i))    ;0 1 2 3 ...
    sn = '-' + string(nn,format='(i2.2)') ; '-00' '-01' '-02' '-03' ...
    sn[0] = ""
    for j = 0, n_calsets.(i) -1 do begin
        calfile.(i)[j] = calfilebase + '-cal' + aname[i] + sn[j] + '.txt'
        ;test if such a calfile can be found:
        if not file_test( calfile.(i)[j], /regular) then begin
            printf,RF,"ERROR! Calibration parameter file "+ (calfile.(i))[j] + " not found or not a regular file!"
            printf, RF, "aborting."
            return
        endif
        if debug ge 1 then printf,RF," reading calfile: " +   (calfile.(i))[j]
        read_calfile,calfile.(i)[j], scanangle, tC0, tC1, tFy, tFx, RF ; 't' for temporary
        C0.(i)[j,*] = tC0
        C1.(i)[j,*] = tC1
        Fy.(i)[j,*] = tFy
        Fx.(i)[j,*] = tFx
    endfor
endfor
```

```
if debug ge 2 then printf,RF, C0

if SAT_EPS eq -99 then begin
;calculate saturation thresholds from focal points:
; take focal point for nadir view
    scaling=1.0
    SAT23_h = Fy.a234[0,0]*scaling;F23_h[0]*scaling
    SAT34_h = Fx.a234[0,0]*scaling;F34_h[0]*scaling
    SAT34_l = Fy.a345[0,0]*scaling;F34_l[0]*scaling
    SAT45_l = Fx.a345[0,0]*scaling;F45_l[0]*scaling
endif

;---------------
; 2. prepare Tbs
;---------------

dt_23 = ch_2 - ch_3
dt_34 = ch_3 - ch_4
dt_45 = ch_4 - ch_5


;check sign?
neg_23 = where (dt_23 le 0,count_23,complement=pos_23,ncompl=ncount_23)
neg_34 = where (dt_34 le 0,count_34,complement=pos_34,ncompl=ncount_34)
neg_45 = where (dt_45 le 0,count_45,complement=pos_45,ncompl=ncount_45)
if debug ge 1 then printf,RF,'no of points where 23, 34, 45, negative:',count_23, count_34, count_45
if debug ge 1 then printf,RF,'no of points where 23, 34, 45, positive:',ncount_23, ncount_34, ncount_45

;------------------------------------------------------------
;3. applying the algorithm(s) the first time ("First pass")
;-------------------------------------------------------------

;get indices where the two algorithms are used.
use_345 = where (dt_45 lt SAT45_l and dt_34 lt SAT34_l)
use_234 = where (dt_45 ge SAT45_l and dt_34 lt SAT34_h and dt_23 lt SAT23_h)
if debug ge 2 then help,use_234,use_345

;qqq=dt_34
;qqq[use_234]=-20.


;create twv array of same dimensions as Tbs:
twv = make_array(size(dt_34,/dim),/float,value=NO_RESULT)
help,twv
;print,twv[1:100]


qqq=use_234
```

```
cos_scan = cos(!DTOR*scanangle)
if debug ge 2 then print, 'cos_scan',cos_scan
;help

;calculate twv with 345 algorithm for all indices where this should work (use_345, see above)

if use_345[0] ne -1 then begin
    if keyword_set(CHECK) then twv[use_345] = 0.5 else begin
;     twv[use_345] = ( C0_l[scanpos[use_345]] + C1_l[scanpos[use_345]] $
;                      * alog((dt_34[use_345] - F34_l[scanpos[use_345]])/(dt_45[use_345] - F45_l[scanpos[use_345]])) )$
;       * cos_scan[scanpos[use_345]]
        twv[use_345] = ( C0.a345[0,scanpos[use_345]] + C1.a345[0,scanpos[use_345]] $
                         * alog((dt_34[use_345] - Fy.a345[0,scanpos[use_345]])/(dt_45[use_345] - Fx.a345[0,scanpos[use_345]])) )$
          * cos_scan[scanpos[use_345]]
;check where this yields more than SWITCH_TWV (e.g., 1.5 kg/m^2)
        above_switch = where (twv gt SWITCH_TWV and twv lt 20.)
;and recalculate with 234 algorithm
        if debug ge 1 then help,use_234
        use_234 = [use_234,above_switch]
    endelse
endif ; use_345 was not -1

if debug ge 2 then help,use_234,use_345

;calculate twv with 234 algorithm for all indices where this should work (use_234, see above)
if use_234[0] ne -1 then begin
    if keyword_set(CHECK) then twv[use_234] =6.5 else begin
;     twv[use_234] = ( C0_h[scanpos[use_234]] + C1_h[scanpos[use_234]] $
;                      * alog((dt_23[use_234] - F23_h[scanpos[use_234]])/(dt_34[use_234] - F34_h[scanpos[use_234]])) )$
;       * cos_scan[scanpos[use_234]]
        twv[use_234] = ( C0.a234[0,scanpos[use_234]] + C1.a234[0,scanpos[use_234]] $
                         * alog((dt_23[use_234] - Fy.a234[0,scanpos[use_234]])/(dt_34[use_234] - Fx.a234[0,scanpos[use_234]])) )$
          * cos_scan[scanpos[use_234]]
    endelse
endif ; use_234 was not -1


;------------------------------------------------------------------------
;4. if desired, applying the algorithm(s) a second time ("2nd pass"), with
;several subranges.
;------------------------------------------------------------------------
if not keyword_set(CHECK) then begin
    printf,RF, "twv_algo: 2nd pass calculation"
; run through both algorithms:
    for i = 0,1 do begin        ; i=0 is a345, i=1 is a234
;test if subranges were given
        if n_elements(subranges.(i)) ge 2 then begin
; recalculating TWV from Tbs depending in which subrange the first pass TWV is
            for j = 1 , n_calsets.(i)-1 do begin
```

```
                use_subrange = where(twv lt subranges.(i)[j] and twv ge subranges.(i)[j-1])
                if debug ge 1 then begin
                    printf,RF,format='("subrange no. ",i2.2,": twv=[",f3.1,",",f3.1,"] ")',$
                      j,subranges.(i)[j-1],subranges.(i)[j]
                    printf,RF,"no of twv values in this subrange:", n_elements(use_subrange)
                endif
                if i eq 0 then begin ;algo345
                    RCBT = alog((dt_34[use_subrange] - Fy.(i)[j,scanpos[use_subrange]])/$
                                (dt_45[use_subrange] - Fx.(i)[j,scanpos[use_subrange]]))
                endif else if i eq 1 then begin ; algo234
                    RCBT = alog((dt_23[use_subrange] - Fy.(i)[j,scanpos[use_subrange]])/$
                                (dt_34[use_subrange] - Fx.(i)[j,scanpos[use_subrange]]))
                endif
                twv[use_subrange] = ( C0.(i)[j,scanpos[use_subrange]] + C1.(i)[j,scanpos[use_subrange]] $
                                      * RCBT ) * cos_scan[scanpos[use_subrange]]
            endfor
        endif
    endfor
endif

neg_twv = where (twv lt 0.)
help,neg_twv

;twv[neg_twv] = NEG

;undef_twv = where (not finite(twv)) ; i.e. where twv = NaN or infinity
;help,undef_twv
;twv[undef_twv] = UNDEF

end
```

References

Miao, J., K. Kunzi, G. Heygster, T. A. Lachlan-Cope, J. Turner, Atmospheric water vapor over Antarctica derived from SSM/T2 data, *J. Geophys. Res.*, vol. 106, no. D10, p. 10187–10203, 2001.

Miao, J., *Retrieval of Atmospheric Water Vapor Content in Polar Regions Using Spaceborne Microwave Radiometry*, Dissertation Univ. Bremen, Fachbereich 1 (Physik und Elektrotechnik) and: Reports on Polar Research 289/1998, 109 pp., Alfred Wegener Institute for Polar and Marine Research, Bremerhaven, Germany, 1998.