Technical Note on:

Software Architecture and Algorithms Definitions

25 March 1998

Issue: 3
Revision: A
(Compliant with ORM_ABC Vers. 1.1)

Delivery of the study:
"Development of an Optimised Algorithm for Routine p,T Retrieval from MIPAS Limb Emission Spectra".

Prepared by:

Name	Institute
B. Carli	IROE-CNR
A. Gignoli	FMA
M. Höpfner	IMK
P. Raspollini	FMA
M. Ridolfi	IROE-CNR

Approved by:

Name	Institute
M. Carlotti	University of Bologna

TABLE OF CONTENTS

- List of modified sections

The following table contains the list of the sections which have been modified in the current revision of the TN with respect to Issue 3 and the description of the main modifications in each section.

Section		Main modifications
2.1	RETR_PT	Updated calls to the modified routines
2.2.10	GUESSPAR_PT	Modified interface, updated calls to mwcont_pt and ficarra_pt
2.2.11	FWDMDL_PT	Updated calls to the modified routines
2.2.11.1	MKPLEV_PT	Added intialisation of variable *ifails*
2.2.11.11	CROSS_PT	Use of 'compressed' grid implemented, modified interface
2.2.11.18	SPECTRUM_PT	Modified interface, use of 'compressed' grid and direct interpolation / convolution implemented.
2.2.12	ABCALC_PT	*rb* ---> *rbt* and run-time optimisations.
2.2.15	NEWPAREST_PT	*rb* ---> *rbt* and re-organisation of the do-loops.
2.2.16	UPDPROF_PT	Modified interface, updated calls to mwcont_pt and ficarra_pt
2.2.19	OUTPUT_PT	Added call to cont_char_pt
2.2.26	MWCONT_PT	The description has been improved, added calculation of *lccmat*, modified interface
2.2.27	FICARRA_PT	The description has been improved, changed interpolation method, modified interface
2.2.28	JACLOSCALC	The module structure has been simplified
2.2.30	READ_IRRGRID_PT	Added calculation of variables *nused1*, *rsan* and *ilim*; modified interface
2.2.31	READ_LOOKUP_PT	Storage of the look-up tables on the 'compressed' grid, storage of *tab*, modified interface.
2.2.32	DECOMPR_PT	Handling of different kinds of tabulation for the LUTs.
2.2.34	CONT_CHAR_PT	New module for retrieved continuum parameters characterisation
2.3	Variables and parameters ...	Added the new parameter *imxsi2* and the new variables: *iigrid, igridc, ilim, lccmat, nused1, rsan, tab.* Changed dimension of *ru, rcross, rcrosspert.*
3.1	RETR_VMR	Updated calls to the modified routines
3.2.10	GUESSPAR_VMR	Modified interface, updated calls to mwcont_vmr and ficarra_vmr
3.2.11.16	CROSS_VMR	Use of 'compressed' grid implemented, modified interface
3.2.11.26	SPECTRUM_VMR	Modified interface, use of 'compressed' grid and direct interpolation / convolution implemented.
3.2.12	ABCALC_VMR	*rb* ---> *rbt* and run-time optimisations.
3.2.15	NEWPAREST_VMR	*rb* ---> *rbt* and re-organisation of the do-loops.
3.2.16	UPDPROF_VMR	Modified interface, updated call to ficarra_pt
3.2.19	OUTPUT_VMR	Modified interface, added call to cont_char_vmr
3.2.33	CONT_CHAR_VMR	New module for retrieved continuum parameters characterisation
3.3	Variables and parameters ...	Added the new parameter *imxsi2* and the new variables: *iigrid, igridc, ilim, lccmat, nused1, rsan, tab.* Changed dimension of *ru, rcross.*

Routines *spe_int_pt.f* and *spe_int_vmr.f* have been dropped from the ORM_ABC_V1.1 code and their description have been replaced by the description of the new routines *cont_char_pt.f* and *cont_char_vmr.f* in sections 2.2.34 and 3.2.33 respectively.

- Introduction

MIPAS (Michelson Interferometer for Passive Atmospheric Sounding) is an ESA developed instrument to be operated on Board ENVISAT-1 as part of the first Polar Orbit Earth Observation Mission program (POEM-1). MIPAS will perform limb sounding observations of the atmospheric emission spectrum in middle infrared region. Concentration profiles of numerous trace gases can be derived from MIPAS observed spectra.

According to the current baseline ESA data processing will routinely retrieve from MIPAS measurements altitude profiles of atmospheric pressure and temperature (p,T), and of volume mixing ratio (VMR) of five high priority species (O_3, H_2O, HNO_3, CH_4 and N_2O). The retrieval of these parameters from calibrated spectra (Level 1b data) is performed by the Level 2 processor.

Level 2 processing is expected to be a critical part of the Payload Data Segment (PDS) because of both the long computing time that may be required and the need for a validated algorithm capable of producing accurate and reliable results.

The study for the "Development of an Optimised Algorithm for Routine P, T and VMR Retrieval from MIPAS Limb Emission Spectra" is meant to provide a scheme for Level 2 analysis, suitable for implementation in ENVISAT PDS and optimised for the requirements of speed and accuracy. The result of the study will be used by industry as an input for the development of the industrial prototype of Level 2 code.

In this document the software architecture and algorithms of Level 2 scientific code are described following the guidelines provided by ESA. The descriptions are given both the Optimised Forward Model (OFM) and the Optimised Retrieval Model (ORM).

- Purpose of the document

The purpose of this document is to define architecture and algorithms of level 2 scientific processor. The document should provide sufficient information in order to fully understand the functionality of the different modules constituting the scientific code delivered to ESA.

- Applicable and reference documents

The applicable and reference documents of the present technical note are listed below.

Applicable documents:			
No	**Document**	**Issue**	**Title**
AD 1	PO-TN-BOM-GS-0010	1	MIPAS Input/Output Data Definition
AD 2	PO-RS-ESA-GS-0177	1	MIPAS Level 2 Processing Input/Output Data Definition
AD3	PF-TN-ESA-GS-0009	3.1	ENVISAT Payload to Target Parameters Calculation Software Interface and Installation Guide
AD4	PPF-TN-ESA-GS-0006	3.1	ENVISAT Orbit Propagator
AD5	PO-TN-ESA-GS-0242	5.0	ENVISAT-1 Product Format Guidelines
AD6	TN-IROE-RSA9601	2	High level algorithm definition on physical and mathematical optimisations
AD7	TN-IROE-RSA9501	4	High Level Software Architecture and Retrieval Modules Interfaces

Reference documents:			
No	**Document**	**Issue**	**Title**
RD1	PO-RS-DOG-GS-0001	draft	MIPAS Level 2 Processor Prototyping, Software Requirements Document
RD2		2	W.H.Press et all: 'Numerical Recipes in FORTRAN' Sedcond edition (1992)
RD3	PO-TN-OXF-GS-0010	?	Generation of Optimised Spectral Grids
RD4	PO-TN-OXF-GS-0011	?	Generation of compressed look-up tables

- Acronyms

The acronyms used in the present technical note are listed below:

AILS	Apodized Instrument Line Shape
FOV	Field Of View
HW	Half Width
ILS	Instrument Line Shape
PDS	Payload Data Segment
UTC	Universal Time Coordinated
VC	Variance - Covariance
VCM	Variance Covariance Matrix
VMR	Volume Mixing Ratio
ZPD	Zero Path Difference
r.u.	Radiance Units: $nW / (cm^2 * sr * cm^{-1})$
MW	Microwindow

1. Level 2 scientific processor

The final goal of Level 2 scientific processor is to retrieve atmospheric pressure and temperature distributions and Volume Mixing Ratio (VMR) profiles of the five high priority chemical species measured by MIPAS. The code performs six retrievals: the first is called p, T retrieval, the other five retrievals are called VMR retrievals and are performed by the same VMR retrieval module. VMR retrievals must be performed after the p,T retrieval has been completed..
In the ORM code Vers.2 (and later versions), the six retrievals are performed by a single main program which contains the calls to p,T and VMR retrieval modules. The actual sequence of operations is shown in the logical scheme reported in Fig. 1. The time sequence of the six retrievals is established by the need of exchanging the internal data sets described in Sect. 2.1.4 of AD7.

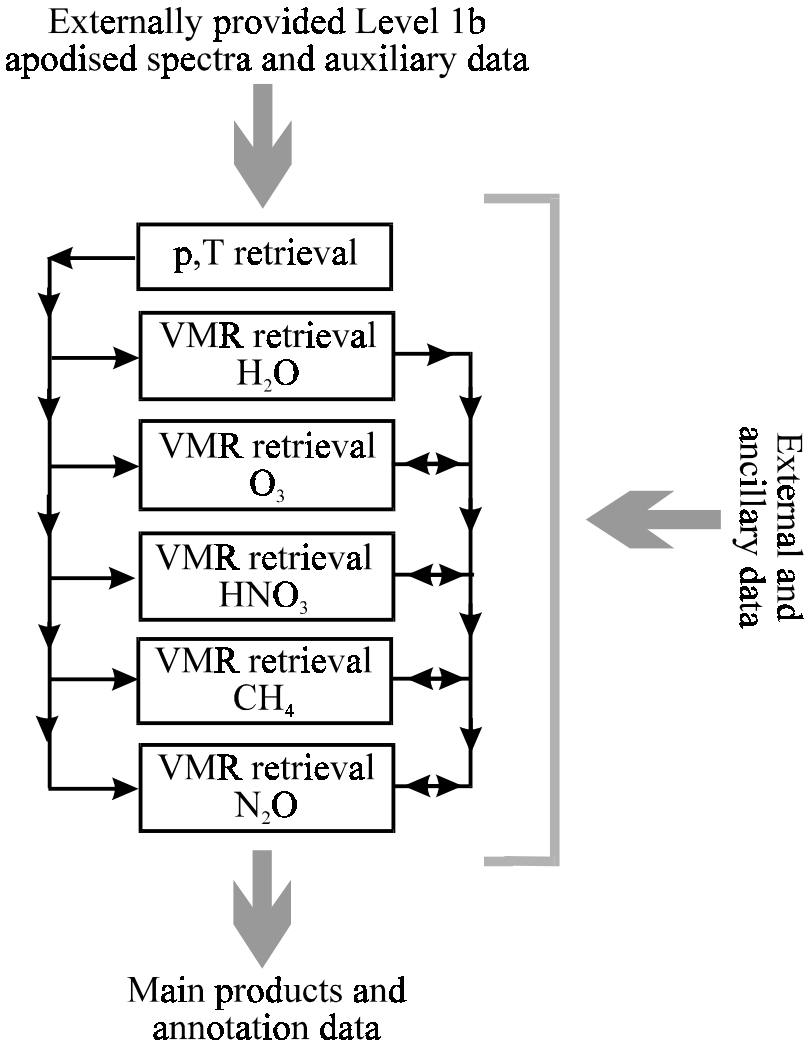


Fig. 1 Logical scheme of the ORM.

This sequence of operations, in the scientific code, is carried-out by the main program module named ‘orm.f’. Hereafter we report the FORTRAN code of this module. The source is self explanatory, thanks to the comments within the lines. Please note that this module is also used to read from the environment the variable defining the location of I/O directories.

```
      program orm
      implicit none
      logical lifptwassucc
      character siod*80, sidir*90, sodir*90
      integer*4 iodl
      common/iopaths/ sidir,sodir,iodl

* Reads the location of I/O directories from the environment
* variable ORM_IODIR and performs some checks on it:

      call getenv ("ORM_IODIR",siod)
      iodl = index(siod," ")-1              ! computes the length of 'siod'
      if (iodl.gt.80) then
        write(*,*)' --- FATAL ERROR in main orm ---'
        write(*,*)'Path of the I/O directories too long !'
        stop
      end if
      if (siod(iodl:iodl).ne.'/') then
        write(*,*)' --- FATAL ERROR in main orm ---'
        write(*,*)'The environment variable ORM_IODIR'
        write(*,*)'must end by / '
        stop
      end if
      sidir = siod(1:iodl)//'INP_FILES/'
      sodir = siod(1:iodl)//'OUT_FILES/'
      iodl = iodl + 10
*
* sidir = full name of the Input directory,
* sodir = full name of the Output directory,
* iodl = number of characters of sidir and sodir
*
      write(*,*)'ORM input directory = ',sidir(1:iodl)
      write(*,*)'ORM output directory = ',sodir(1:iodl)
*
* Removes out-dated dump files:
      write(*,*)'Removing out_dated dump files ...'
      call system('rm '//sidir(1:iodl)//'*_dump.dat')
      call system('rm '//sodir(1:iodl)//'*_dump.dat')
      write(*,*)'Done !!'
*
* Runs p,T retrieval:
      call retr_pt(lifptwassucc)
*
* lifptwassucc = TRUE   ---> p,T retrieval was successful
     * lifptwassucc = FALSE ---> p,T retrieval was unsuccessful
     * if p,T retrieval was unsuccessful the ORM stops (the operational code has to jump to next scan)
      if (.not.lifptwassucc) then
        write(*,*) 'p,T retrieval was not successful,'
        write(*,*) 'VMR retrievals will not be performed'
        stop 'p,T retieval was unsuccessful!!'
      end if
* Copies p,T retrieved profiles in the ORM input directory
      call system('cp '//sodir(1:iodl)//'pt_dump.dat '//sidir(1:iodl)//'pt_dump.dat')

* Runs H2O retrieval:
      call retr_vmr(1)
* Copies H2O retrieved profile in the ORM input directory
      call system('cp '//sodir(1:iodl)//'h2o_dump.dat '//sidir(1:iodl)//'h2o_dump.dat')
```

```
* Runs O3 retrieval:
    call retr_vmr(3)
* Copies O3 retrieved profile in the ORM input directory
     call system('cp '//sodir(1:iodl)//'o3__dump.dat '//sidir(1:iodl)//'o3__dump.dat')

* Runs HNO3 retrieval:
    call retr_vmr(12)
* Copies HNO3 retrieved profile in the ORM input directory
    call system('cp '//sodir(1:iodl)//'hno_dump.dat '//sidir(1:iodl)//'hno_dump.dat')

* Runs CH4 retrieval:
    call retr_vmr(6)
* Copies CH4 retrieved profile in the ORM input directory
    call system('cp '//sodir(1:iodl)//'ch4_dump.dat '//sidir(1:iodl)//'ch4_dump.dat')

* Runs N2O retrieval:
    call retr_vmr(4)
* All the required retrievals have been completed !'
    end
```

2. Software architecture and algorithms of p,T retrieval scientific code

In this section the software architecture and the algorithms used in p,T retrieval module are specified. Section 2.1 shows the high level flow diagram of the calls between main modules and the detailed calling tree. The tree of calls of each module, its I/O data and the algorithms are described in section 2.2. Section 2.3 contains a description of parameters and variables used by the modules constituting p,T retrieval.

2.1 High level flow diagram of calls

Fig. 2 shows the high level flow diagram of calls of the p,T retrieval module. Each box corresponds to a single main module of the program. The FWDMDL_PT module is however an exception and it contains more than one main module.

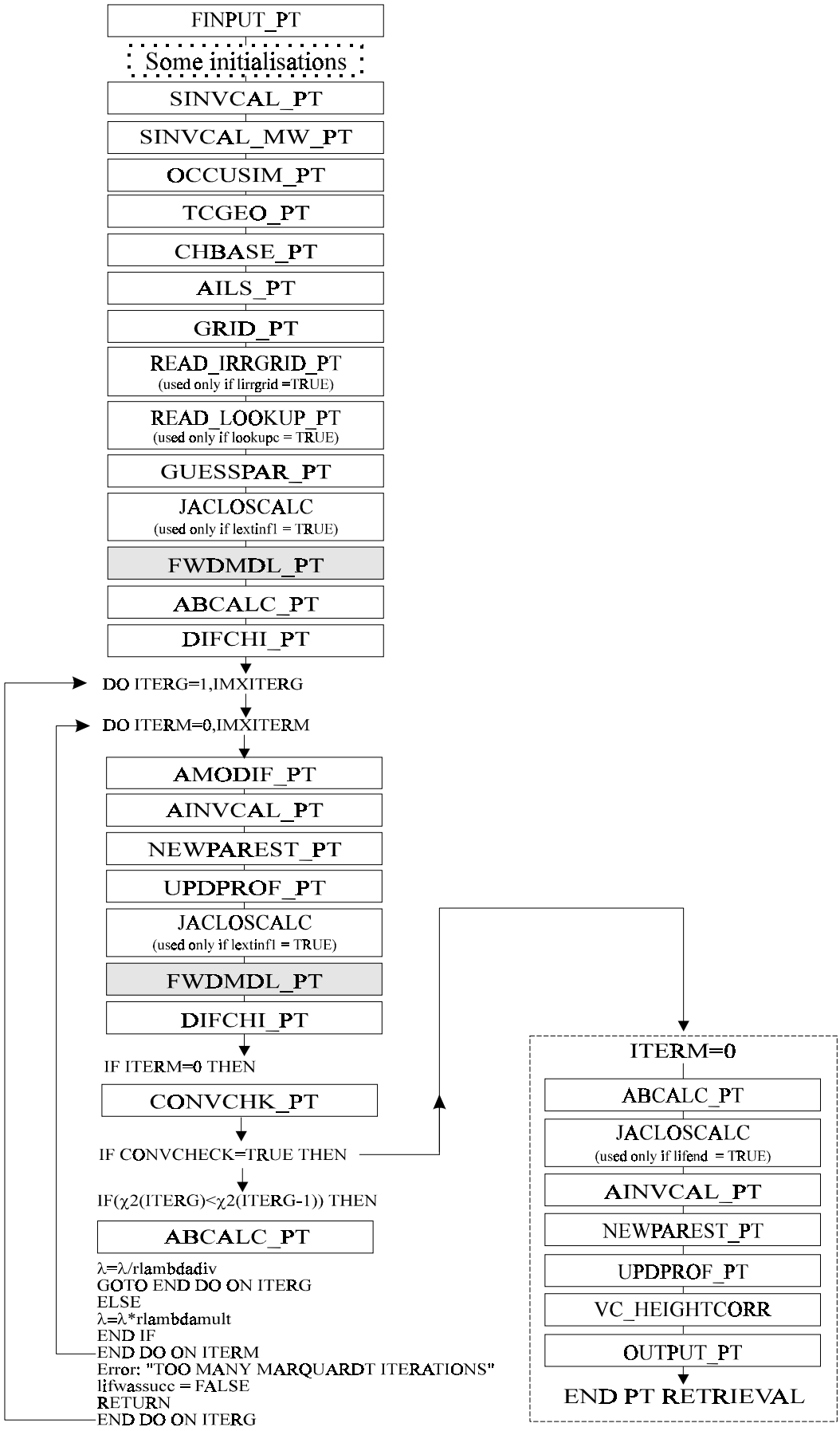


Fig. 2 High Level flow diagram of p-T retrieval module

The operations described in the flow diagram of Fig.2 are carried-out by the program module 'RETR_PT'. Hereafter, the calling tree of this module is described. The meaning of tree symbols is [1]:

. ==> terminal node in the tree
* ==> external procedure
> ==> subtree node, expanded below
+ ==> multiply called terminal node
] ==> procedure calling only externals
? ==> module is in IF clause
(==> module is in DO loop

```
RETR_PT ]
    |-----INPUT_PT *
    |-----SINVCAL_PT *
    |-----SINVCAL_MW_PT *
    |-----OCCUSIM_PT *
    |-----TCGEO_PT *
    |-----CHBASE_PT *
    |-----FAILS_PT *
    |-----GRID_PT *
    |?----READ_LOOKUP_PT*
    |?----READ_IRRGRID_PT*
    |-----GUESSPAR_PT *
    |?----JACLOSCALC*
    |-----FWDMDL_PT *
    |-----ABCALC_PT *
    |-----DIFCHI_PT *
    |((---AMODIF_PT *
    |((---AINVCAL_PT *
    |((---NEWPAREST_PT *
    |((---UPDPROF_PT *
    |((?--JACLOSCALC*
    |((---FWDMDL_PT *
    |((?--DIFCHI_PT *
    |((?--CONVCHK_PT *
    |((---ABCALC_PT *
    |((---OUTPUT_PT *
    |((?--JACLOSCALC*
    |((---ABCALC_PT *
    |((---AINVCAL_PT *
    |((---NEWPAREST_PT *
    |-----UPDPROF_PT *
    |-----VC_HEIGHTCORR*
    |-----OUTPUT_PT *
```

The complete structure of 'retr_pt' is described in Fig. 3. Frames drawn with a continuous line represent the main modules, i.e source (.f) and object (.o) files. Outlined frames refer to submodules

[1] This structure chart is an output of the freeware floppy and flow programs developed by Julian J. Bunn at CERN Computer Centre

contained in one of the main modules. For a detailed description of program modules see section 2.2.

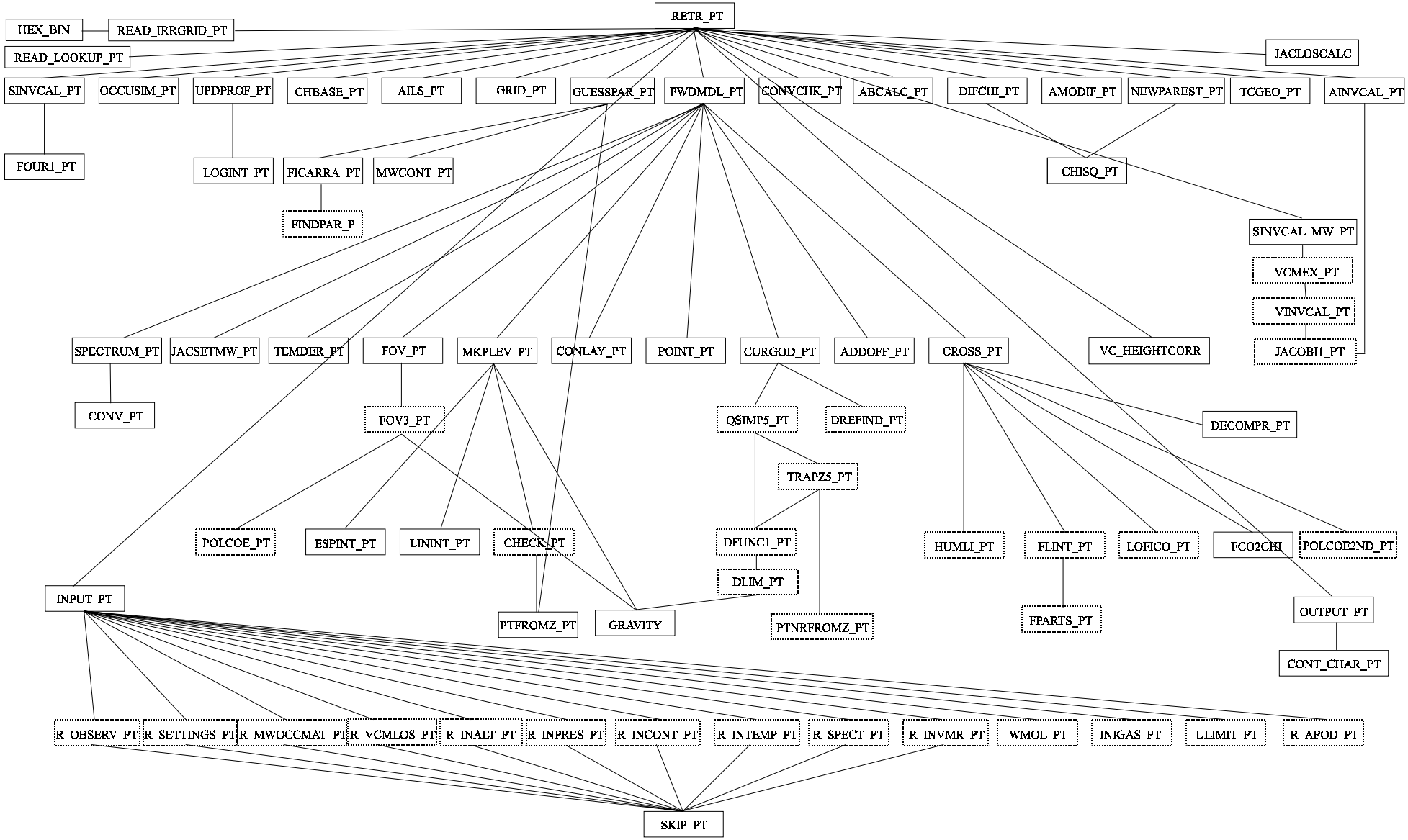


Fig. 3 Complete structure of p-T retrieval module

Since in the 'RETR_PT' module performs also some initialisations and other operations which are marginal with respect to the flow of the calls, we report here the details of the algorithms contained in RETR_PT module.

Variables echanged with external modules

The INPUT_PT routine which is called by RETR_PT at the beginning of the flow, reads and sets-up all the program variables used by the different program modules. The variables are passed from INPUT_PT to RETR_PT through common statements.

Detailed description

We report hereafter an extract of the FORTRAN source code of RETR_PT module, where the performed operations are explained in detail. Please note that the program lines included whithin the 'special' comment lines '* +++++++++++++++++' are not to be included in the level 2 prototype code because are used only for debugging purposes.

```
      subroutine retr_pt (lifwassucc)
      implicit none
      include 'parameters_pt.inc'
*
Declaration of variables and common statements follow (see the source code in retr_pt.f).
*
* Reads input files:
      rdtime = etime(rtar)
      write(*,*)'before input'
      call input_pt
      rdtime = etime(rtar)
      write(*,*)'E_Time after input_pt (s) = ',rtar(1)+rtar(2)
*
* Reads the environment variable TEP which establishes
* whether this is a test run for writing-out variables at the TEPs:
      call getenv('TEP',step)

      if (lextinf1) then
        write(*,*)'Using a-priori LOS info during the retrieval'
        lifend = .false.
      end if
      if (lifend)
     &   write(*,*)'Using a-priori LOS info after the retrieval'

      write(*,*)'lfit = ',(lfit(j),j=1,ilimb)
*
      write(*,'(/a)')'lokku = '
      do j=1,ilimb
        write(*,*) (lokku(j,k),k=1,nselmw)
      end do
*
* Some initialisations:
*
      lifwassucc = .true.       ! p,T retrieval is assumed to be successful at the beginning
      rlambda=rlambdain     ! initial value of Marquardt damping factor
      rdt=2.D0                  ! temperture increment for calculation of derivatives wrt temperature
      iterg=0                   ! initialisation of gauss iterations index
*
* Initialisation of the instrumental offset
      do k=1,nselmw
        roffs(k)=0.D0
```

```
      end do
* initial guess tangent altitudes
      do j=1,imxgeo
        rztanginit(j)=rztang(j)
      enddo
*
      write(*,*)'Current latitude (deg.) = ',rlat
*********************************************************************
* Atmospheric continuum profiles are scaled and set to zero where
* required:
      do k=1,ipro
       if (rzprof(k).gt.rzc0) then
         do j=1,nselmw
          rcprof(k,j) = 0.D0
         end do
       else
         do j=1,nselmw
          rcprof(k,j)=rcprof(k,j)*1.D30
         end do
       end if
      end do
*********************************************************************
* Setup of upper continuum limit (nucl) and umbrella radius:
*
      nucl = 0
      do j=1,ilimb
        if(rztang(j).gt.rucl.and.rucl.ge.rztang(j+1)) nucl=j
      end do
      write(*,'(a,f5.2,2x,i4,2x,f5.2)')
     &          'rucl, nucl, rperc = ',rucl,nucl,rperc

* +++++++
* Please note that this is only a re-initialisation which is not
* to be performed in the operational code:
      do j=1,nselmw
       do k=1,ilimb
         rconint(k,j) = 10.D0
       end do
      end do
* +++++++
* +++++++
* writing into a file for plotting continuum profiles:
      open(50,file=sodir(1:iodl)//'pt_rcprof_ref.dat',
     &     status='unknown')
      do j=1,ipro
        write(50,'(25e20.5)')rpprof(j),(rcprof(j,k),k=1,nselmw)
      end do
      close(50)
* +++++++
*********************************************************************
      write(*,*)'before sinvcal'
      call sinvcal_pt(rapod,napod,rzerof,rapod_sigma,nailsdp,
     &                rvcmobinvopt)
      write(*,*)'before sinvcal_mw'
      call sinvcal_mw_pt(rapod_sigma,nailsdp,rvcmobinvopt,
     &                   nselmw,nsam,rzerof,rvcmobinv)
*
* TEP_01_PT
      if(step.eq.'test')
```

```
     & call tep_01_pt(nselmw,nsam,rvcmobinv)
*
      write(*,*)'before occusim'
      call occusim_pt(rztang,ilimb,imaingas,lokku,nselmw,lfit,
     &          rbase,rzsi,nsam,igeo,lfitgeo,ipar,iocsim,ilimbmw,
     &          irowmw,iobs,rzpar)
*
      write(*,*)'Before tcgeo'
      call tcgeo_pt(lfitgeo,ipar,igeo,lfit,ilimb,nucl,
     &               igeotder,igeocder)
*
      write(*,*)'before chbase'
      call chbase_pt(rzprof,rtprof,rpprof,rvmrprof,rcprof,ipro,igas,
     &          nselmw,rztang,ilimb,rzpar,ipar,rlat,lfit,
     &          rzbase,rtbase,rpbase,rvmrbase,rcbase,ibase,lparbase)
*
*
* ++++++++
      open(44,file=sodir(1:iodl)//'pt_inguessb_ztp.dat',
     &        status='unknown')
      do j = 1,ibase
        write(44,'(3e15.5)')rzbase(j),rtbase(j),rpbase(j)
      end do
      close(44)
      open(46,file=sodir(1:iodl)//'pt_inguess_ztang.dat',
     &        status='unknown')
      open(45,file=sodir(1:iodl)//'pt_inguess_ptang.dat',
     &        status='unknown')
      do j = 1,ilimb
        write(46,*)j,rztang(j)
        write(45,'(i5,e15.5)')j,rpbase(ibase-1-ilimb+j)
      end do
      close(46)
      close(45)

      open(50,file=sodir(1:iodl)//'pt_rcbase_ing.dat',
     &     status='unknown')
      do j=1,ibase
        write(50,'(25e20.5)')rpbase(j),(rcbase(j,k),k=1,nselmw)
      end do
      close(50)
* ++++++++
*
*
      write(*,*)'before fails'
      call fails_pt(nselmw,nailsdp,nrd,rails,delta,dstep,rils,iadd,
     &            nils,rintils)
*
      write(*,*)'before grid'
      call grid_pt(nselmw,nsam,nrd,dstep,ifspmw,
     &          iadd,delta,iline,dsilin,ioutin,isigma,dsigma)
*
      rdtime = etime(rtar)
      r1 = rtar(1)+rtar(2)
      if (lirrgrid) then
         write(*,*)'before read_irrgrid'
         call read_irrgrid_pt(lirrgridmw,smw,nselmw,
     &                 dsigma,isigma,delta,nrd,igeo,iigrid,
     &                 cint,igridc,nused1,rsan,ilim,rils,
```

```
     &                nils,nsam)
      else
        do imw=1,nselmw
          lirrgridmw(imw)=.false.
        end do
      end if
      rdtime = etime(rtar)
      r1 = rtar(1)+rtar(2) - r1
      write(*,*)'E_Time required for read_irrgrid (s): ',r1
*
      rdtime = etime(rtar)
      r1 = rtar(1)+rtar(2)
      write(*,*)'E_Time before read_lookup (s): ',r1
      write(*,*)'before read_lookup'
      if (lookupc) call read_lookup_pt(ilookupmw,lmgas,smw,nselmw,
     &           igasmw,igashi,igasnr,dsigma,isigma,
     &           nll,npl,rp1l,rdpl,ntl,rt1l,
     &           rdtl,ru,rkl,iigrid,lirrgridmw,tab)
      rdtime = etime(rtar)
      r1 = rtar(1)+rtar(2) - r1
      write(*,*)'E_Time required for read_lookup (s): ',r1
*
      write(*,*)'before guesspar'
      call guesspar_pt(rzbase,rtbase,rpbase,rcbase,ibase,nselmw,rlat,
     &       rzpar,ipar,rztang,ilimb,lfit,lokku,lparbase,rperc,rconint,
     &       rxpar,itop,icontpar,rjaccon,isaved,dstep,nsam,
     &       ifspmw,nucl,lcfit,lccmat,ifco)
*
      if(lextinf1) call jacloscalc(rxpar,ilimb,ipar,itop,rztang,
     &                      rlat,rjaclos)
*
* TEP_02_PT:
      if(step.eq.'test')
     &   call tep_02_pt(ibase,igeo,ipar,igas,ilimb,itop,
     &   nselmw,igasmw,isigma,igeocder,igeotder,rzbase,
     &   rtbase, rpbase,rcbase,rvmrbase,rztang,rzsi,lfit,
     &   lfitgeo,lokku,iocsim,rxpar)
*
      rdtime = etime(rtar)
      r1 = rtar(1)+rtar(2)
      write(*,*)'E_Time before first call to fwdmdl_pt (s): ',r1
      write(*,*)'before fwdmdl'
      call fwdmdl_pt(rzsi,igeo,rzbase,rtbase,rpbase,rvmrbase,
     &               rcbase,ibase,rulatm,rwmolref,dsigm0,
     &               rhw0ref,rmaxtv1,rmaxtv2,rzt12,rhwvar,
     &               igas,rexphref,rincz,redfact,rlat,
     &               lfitgeo,rdt,lparbase,nselmw,iept,rearad,
     &               deps,isigma,dsigma,delta,iocsim,igasmw,
     &               ruplin,rlolin,iline,icode,rint0,relow,rhw0,
     &               dsilin,ioutin,igasnr,rexph,rwmol,igashi,
     &               iiso,ninterpol,nsam,nils,rils,rintils,nrd,
     &               iadd,ilimbmw,lokku,nucl,ilimb,igeocder,
     &               igeotder,rjaccon,roffs,rbase,rsl,icontpar,
     &               ipar,irowmw,
     &               ilookupmw,lmgas,smw,nll,npl,
     &               rp1l,rdpl,ntl,rt1l,rdtl,ru,rkl,tab,
     &               rjacob,rspfov,iigrid,cint,lirrgridmw,
     &               igridc,nused1,rsan,ilim)
      rdtime = etime(rtar)
```

```
      r1 = rtar(1)+rtar(2) -r1
      write(*,*)'E_Time required for fwdmdl_pt (s): ',r1
*
      rdtime = etime(rtar)
      r1 = rtar(1)+rtar(2)
      write(*,*)'before abcalc_pt'
      call abcalc_pt(rjacob,rvcmobinv,ra,rbt,iobs,
     &  itop,nselmw,ilimbmw,nsam,rnoise,ilimb,lokku,
     &  rjaclos,ipar,rinvclos,lextinf1,rblos,lifend,.FALSE.)
      rdtime = etime(rtar)
      r1 = rtar(1)+rtar(2) -r1
      write(*,*)'E_Time required for abcalc_pt (s): ',r1
*
      rdtime = etime(rtar)
      r1 = rtar(1)+rtar(2)
      write(*,*)'before difchi'
      call difchi_pt(iobs,itop,robs,rspfov,rvcmobinv,rnoise,
     &          nsam,nselmw,ilimb,lokku,
     &          ilimbmw,iterg,rnres,rchisq,rchisqp,
     &          rztang,rdzeng,lextinf1,rnreslos,rinvclos)
      rdtime = etime(rtar)
      r1 = rtar(1)+rtar(2) -r1
      write(*,*)'E_Time required for difchi_pt (s): ',r1

       write(*,*)'Retrieval of: p,T'
       write(*,*)'iterg,rchisq=',iterg,rchisq(iterg)
       rdtime = etime(rtar)
       write(*,*)'E_Time before starting iterations (s): ',
     &          rtar(1)+rtar(2)
*
*  Begin of the do-loop on macro-iterations
*
      do 10 iterg=1,imxiterg

      rdtime = etime(rtar)
      rtit = rtar(1)+rtar(2)    ! initialisation of E_Time per iteration

      write(*,'(//a,/a/)')c1,c1
      write(*,'(a,i2/)')'Starting GAUSS macro-iteration N. ',iterg
      write(*,*)'iterg=',iterg
*
*  Begin of do-loop on micro-iterations
*
      do 20 iterm=0,imxiterm

      write(*,'(a,i3)')'Macro-iteration N. ',iterg
      write(*,'(a,i2)')'Marquardt micro-iteration index: '
     &               ,iterm
      write(*,*)'iterm=',iterm
      write(*,'(a,e10.3)')'Lambda =  ',rlambda
*
      rdtime = etime(rtar)
      r1 = rtar(1)+rtar(2)
         write(*,*)'before amodif_pt'
         call amodif_pt(ra,rlambda,itop,ipar,icontpar)
      rdtime = etime(rtar)
      r1 = rtar(1)+rtar(2) - r1
      write(*,*)'E_Time required for amodif_pt (s) = ',r1
*
```

```
      rdtime = etime(rtar)
      r1 = rtar(1)+rtar(2)
         write(*,*)'before ainvcal_pt'
         call ainvcal_pt (ra,itop,rainv)
      rdtime = etime(rtar)
      r1 = rtar(1)+rtar(2) - r1
      write(*,*)'E_Time required for ainvcal_pt (s) = ',r1
*
      rdtime = etime(rtar)
      r1 = rtar(1)+rtar(2)
         write(*,*)'before newparest'
      call newparest_pt(rainv,rbt,rnres,rxparold,itop,iobs,
     &                  iterm,rjacob,rxpar,rlinchisq,
     &                  rvcmobinv,rnoise,nsam,nselmw,ilimbmw,
     &                  ilimb,lokku,rblos,rnreslos,lextinf1,
     &                  ipar,rjaclos,rinvclos,lifend,.FALSE.)
      rdtime = etime(rtar)
      r1 = rtar(1)+rtar(2) - r1
      write(*,*)'E_Time required for newparest_pt (s) = ',r1
*
************************************************************************
* Constraining p,T and continua in phisically meaningful ranges:
*
      do j=1, 2*ipar
        if (rxpar(j).lt.0.d0) then
           rxpar(j)=0.d0
           write(*,*)'WARNING: constrained rxpar at j =',j
        end if
      end do
      do j=2*ipar+1, 2*ipar+icontpar
        if (rxpar(j).lt.-100.d0) rxpar(j)=-100.D0
        if (rxpar(j).gt.1.D+30) rxpar(j)=1.D+30
      end do
*
************************************************************************

      print*,'CHISQ in linear approx. rlinchisq = ',rlinchisq
      write(*,'(a)')' Actual values of rxpar :'
      write(*,'(6e13.4)')(rxpar(j),j=1,itop)

      write(*,*)'before updprof'
      call updprof_pt(rxpar,itop,ipar,rzpar,rzbase,rtbase,rpbase,
     &                ibase,rcbase,nselmw,rvmrbase,igas,roffs,
     &                lparbase,rlat,rztang,ilimb,rzsi,igeo,
     &                rbase,lokku,ilimbmw,icontpar,
     &                isaved,nsam,ifspmw,dstep,rjaccon,
     &                nucl,rperc,rconint,lcfit,lccmat)
*
      if(step.eq.'test')
     & call tep_07_pt(ibase,icontpar,rzbase,rtbase,
     &          rpbase,lparbase,nselmw,rcbase,igas,rvmrbase,
     &          roffs,rztang,ilimb,rzsi,igeo,isaved,
     &          rjaccon,lcfit)
*
* ++++++++
         write(*,*)' rzbase, rtbase, rpbase, ibase = ',ibase
         do j=1,ibase
          write(*,*)rzbase(j),rtbase(j),rpbase(j)
         end do
```

```
        write(*,*)' rzsi,  igeo = ',igeo
        write(*,'(6f10.4)')(rzsi(j),j=1,igeo)
*
*
      open(81,file=sodir(1:iodl)//
     &     'pt_retr_ztp.dat',
     &     status='unknown')
       k=ipar
      do j=1,ibase
        if (lparbase(j)) then
          k=k+1
         write(81,'(5e13.5)')rzbase(j),rpbase(j),rtbase(j),
     &    rtbase(j)-sqrt(rainv(k,k)),
     &    rtbase(j)+sqrt(rainv(k,k))
        else
         write(81,'(5e13.5)')rzbase(j),rpbase(j),rtbase(j),0.,0.
        end if
      end do
      close(81)
*
      open(86,file=sodir(1:iodl)//
     &     'pt_retr_ztang.dat',
     &      status='unknown')
      open(87,file=sodir(1:iodl)//
     &     'pt_retr_ptang.dat',status='unknown')
      do j=1,ilimb
      write(86,'(i3,f15.5)')j,rztang(j)
      write(87,*)j,rpbase(ibase-1-ilimb+j),
     &      sqrt(abs(rainv(j,j)))
      end do
      close(86)
      close(87)

      open(50,file=sodir(1:iodl)//'pt_rcbase_ret.dat',status='unknown')
      do j=1,ibase
       write(50,'(25e20.5)')rpbase(j),(rcbase(j,k),k=1,nselmw)
      end do
      close(50)
*
* ++++++++

      if(lextinf1) call jacloscalc(rxpar,ilimb,ipar,itop,rztang,
     &                        rlat,rjaclos)

      rdtime = etime(rtar)
      r1 = rtar(1)+rtar(2)
*
        write(*,*)'before fwdmdl'
        call fwdmdl_pt(rzsi,igeo,rzbase,rtbase,rpbase,rvmrbase,
     &               rcbase,ibase,rulatm,rwmolref,dsigm0,
     &               rhw0ref,rmaxtv1,rmaxtv2,rzt12,rhwvar,
     &               igas,rexphref,rincz,redfact,rlat,
     &               lfitgeo,rdt,lparbase,nselmw,iept,rearad,
     &               deps,isigma,dsigma,delta,iocsim,igasmw,
     &               ruplin,rlolin,iline,icode,rint0,relow,rhw0,
     &               dsilin,ioutin,igasnr,rexph,rwmol,igashi,
     &               iiso,ninterpol,nsam,nils,rils,rintils,nrd,
     &               iadd,ilimbmw,lokku,nucl,ilimb,igeocder,
     &               igeotder,rjaccon,roffs,rbase,rsl,icontpar,
```

```
     &              ipar,irowmw,
     &              ilookupmw,lmgas,smw,nll,npl,
     &              rp1l,rdpl,ntl,rt1l,rdtl,ru,rkl,tab,
     &              rjacob,rspfov,iigrid,cint,lirrgridmw,
     &              igridc,nused1,rsan,ilim)
*
      rdtime = etime(rtar)
      r1 = rtar(1)+rtar(2) - r1
      write(*,*)'E_Time required fwdmdl_pt (s) = ',r1
*
* Saving the old residuals:
        do j=1,iobs
          rnresold(j) = rnres(j)
        end do
        do j=1,ilimb-1
          rnreslosold(j)=rnreslos(j)
        end do

        rdtime = etime(rtar)
        r1 = rtar(1)+rtar(2)
         write(*,*)'before difchi'
         call difchi_pt(iobs,itop,robs,rspfov,rvcmobinv,rnoise,
     &          nsam,nselmw,ilimb,lokku,
     &          ilimbmw,iterg,rnres,rchisq,rchisqp,
     &          rztang,rdzeng,lextinf1,rnreslos,rinvclos)
        rdtime = etime(rtar)
        r1 = rtar(1)+rtar(2) - r1
        write(*,*)'E_Time required difchi_pt (s) = ',r1
*
      if(step.eq.'test')
     & call tep_08_pt(iobs,iterg,rnres,rchisq,
     &      ilimb,nselmw,rchisqp)
*
         write(*,*)'Retrieval of: p,T'
         write(*,*)'iterg,rchisq=',iterg,rchisq(iterg)
*
         if (iterm.eq.0) then
*
           write(*,*)'before convchk'
           call convchk_pt(rchisq,iterg,rlinchisq,rxpar,rxparold,
     &     ipar,itop,iobs,rlambda,rconvc(1),rconvc(2),rconvc(3),lconverg)
*
           if (lconverg) then
            write(*,'(a)')'The convergence criteria are now verified,'
     &       //' exiting from iterations :-) '
            goto 30
           end if
         end if
*
*******
         if ((rchisq(iterg).le.rchisq(iterg-1))
     &       .or.(rchisq(iterg).lt.1.0)) then
*
      rdtime = etime(rtar)
      r1 = rtar(1)+rtar(2)
           write(*,*)'before abcalc_pt'
      call abcalc_pt(rjacob,rvcmobinv,ra,rbt,iobs,
     &  itop,nselmw,ilimbmw,nsam,rnoise,ilimb,lokku,
     &  rjaclos,ipar,rinvclos,lextinf1,rblos,lifend,.FALSE.)
```

```
      rdtime = etime(rtar)
      r1 = rtar(1)+rtar(2) - r1
      write(*,*)'E_Time required abcalc_pt (s) = ',r1
           rlambda=rlambda/rlambdadiv
           goto 15
         else
           rlambda=rlambda*(rlambdamul-1.D0)/(1.D0+rlambda)
           do j=1,itop
             rxpar(j) = rxparold(j)
           end do
           do j=1,iobs
             rnres(j) = rnresold(j)
           end do
           do j=1,ilimb-1
             rnreslos(j)=rnreslosold(j)
           end do
         end if
20     continue
       write(*,*)'Too many MARQUARDT micro-iterations'
       lifwassucc = .false.
       goto 177
15       call output_pt(rxpar,ipar, icontpar,rainv,
     &   rztang,rztanginit,rvchcorr,
     &   nsam,robs,rspfov,rchisq,iobs,itop,iterg,iterm,rlambda,
     &   rlinchisq,ilimb,igeo,nselmw, rchisqp,slab,lokku,.true.,
     &   lcfit,lccmat,nucl)

      rdtime  = etime(rtar)
      rtit = rtar(1)+rtar(2) - rtit
      write(*,*)'E_Time spent in G. it. ',iterg,' was (s): ',rtit
10     continue
*
      write(*,'(//a,i3/)')'Maximum N. of allowed macro-iterations'
     & //' has been reached, iterg = ',iterg-1
30     continue
*
      write(*,'(a)')'Exited from iterations, producing now the output.'
*
      if(lifend)
     &   call jacloscalc(rxpar,ilimb,ipar,itop,rztang,
     &                   rlat,rjaclos)

      rdtime = etime(rtar)
      r1 = rtar(1) +rtar(2)
      write(*,*)'before abcalc_pt'
      call abcalc_pt(rjacob,rvcmobinv,ra,rbt,iobs,
     &  itop,nselmw,ilimbmw,nsam,rnoise,ilimb,lokku,
     &  rjaclos,ipar,rinvclos,lextinf1,rblos,lifend,.TRUE.)
      rdtime = etime(rtar)
      r1 = rtar(1)+rtar(2) - r1
      write(*,*)'E_Time required abcalc_pt (s) = ',r1
*
      rdtime = etime(rtar)
      r1 = rtar(1)+rtar(2)
         write(*,*)'before ainvcal_pt'
         call ainvcal_pt (ra,itop,rainv)
      rdtime = etime(rtar)
      r1 = rtar(1)+rtar(2) - r1
      write(*,*)'E_Time required ainvcal_pt (s) = ',r1
```

```
*
      rdtime = etime(rtar)
      r1 = rtar(1)+rtar(2)
         write(*,*)'before newparest'
      call newparest_pt(rainv,rbt,rnres,rxparold,itop,iobs,
     &                  iterm,rjacob,rxpar,rlinchisq,
     &                  rvcmobinv,rnoise,nsam,nselmw,ilimbmw,
     &                  ilimb,lokku,rblos,rnreslos,lextinf1,
     &                  ipar,rjaclos,rinvclos,lifend,.TRUE.)
      rdtime = etime(rtar)
      r1 = rtar(1)+rtar(2) - r1
      write(*,*)'E_Time required newparest_pt (s) = ',r1
*
**********************************************************************
* Constraining p,T and continua in phisically meaningful ranges:
*
      do j=1, 2*ipar
        if (rxpar(j).lt.0.d0) then
          rxpar(j)=0.d0
          write(*,*)'WARNING: constrained rxpar at j =',j
        end if
      end do
      do j=2*ipar+1, 2*ipar+icontpar
        if (rxpar(j).lt.-100.d0) rxpar(j)=-100.D0
        if (rxpar(j).gt.1.D+30) rxpar(j)=1.D+30
      end do
*
**********************************************************************

      write(*,'(/a)')'Final estimate of the vector rxpar:'
      write(*,'(6(1pe12.4))')(rxpar(j),j=1,itop)
      write(*,'(/a)')'Final estimate of the square errors on rxpar:'
      write(*,'(6(1pe12.4))')(rainv(j,j),j=1,itop)
*
         write(*,*)'before updprof'
      call updprof_pt(rxpar,itop,ipar,rzpar,rzbase,rtbase,rpbase,
     &                  ibase,rcbase,nselmw,rvmrbase,igas,roffs,
     &                  lparbase,rlat,rztang,ilimb,rzsi,igeo,
     &                  rbase,lokku,ilimbmw,icontpar,
     &                  isaved,nsam,ifspmw,dstep,rjaccon,
     &                  nucl,rperc,rconint,lcfit,lccmat)
*
      if(step.eq.'test')
     & call tep_07_pt(ibase,icontpar,rzbase,rtbase,
     &          rpbase,lparbase,nselmw,rcbase,igas,rvmrbase,
     &          roffs,rztang,ilimb,rzsi,igeo,isaved,
     &          rjaccon,lcfit)
*
* +++++++
      open(81,file=sodir(1:iodl)//
     &     'pt_retr_ztp.dat',status='unknown')
       k=ipar
      do j=1,ibase
        if (lparbase(j)) then
          k=k+1
          write(81,'(5e13.5)')rzbase(j),rpbase(j),rtbase(j),
     &    rtbase(j)-sqrt(rainv(k,k)),
     &    rtbase(j)+sqrt(rainv(k,k))
        else
```

```
          write(81,'(5e13.5)')rzbase(j),rpbase(j),rtbase(j),0.,0.
        end if
      end do
      close(81)
*
      open(86,file=sodir(1:iodl)//
     &     'pt_retr_ztang.dat',
     &      status='unknown')
      open(87,file=sodir(1:iodl)//
     &     'pt_retr_ptang.dat',status='unknown')
      do j=1,ilimb
      write(86,'(i3,f15.5)')j,rztang(j)
      write(87,*)j,rpbase(ibase-1-ilimb+j),
     &      sqrt(abs(rainv(j,j)))
      end do
      close(86)
      close(87)

      open(50,file=sodir(1:iodl)//'pt_rcbase_ret.dat',status='unknown')
      do j=1,ibase
        write(50,'(25e20.5)')rpbase(j),(rcbase(j,k),k=1,nselmw)
      end do
      close(50)

*
* write the plot-file of the final simulations
*
      open(50,file=sodir(1:iodl)//'pt_sim.dat',status='unknown')
      do k=1,nselmw
         do l=1,nsam(k)
          rummy=dsigma(1,k)+(l-1)*.025D0+iadd*delta
          write(50,'(f9.4,30(1pe11.3))')rummy,
     &    (rspfov(l,j,k),j=1,ilimbmw(k))
         end do
      end do
      close (50)
*
* +++++++
*
      call vc_heightcorr(rxpar,ilimb,ipar,rainv,
     &              rztang,rztanginit,rhcorr,rvchcorr)

       call output_pt(rxpar,ipar, icontpar,rainv,
     &  rztang,rztanginit,rvchcorr,
     &  nsam,robs,rspfov,rchisq,iobs,itop,iterg,iterm,rlambda,
     &  rlinchisq,ilimb,igeo,nselmw, rchisqp,slab,lokku,.false.,
     &  lcfit,lccmat,nucl)
*
****
* Writing p,T retrieved values into a dump file:
*

      open(34,file=sodir(1:iodl)//'pt_dump.dat',
     &       form='unformatted',status='unknown')
      write(34) ibase
      write(34) imxpro
      write(34) imxgeo
      write(34) ilimb
      write(34) rpbase
```

```
      write(34) rtbase
      write(34) rzbase
      write(34) rztang
      close (34)

*
****
*
177   continue
      close I/O files, units from 11 to 30 but 14
      close (40)
* Closes TEP files:
      if (step.eq.'test') then
        close tep files, units from 71 to 78
      end if
**
      return
      end
```

2.2 Algorithms and architecture of p,T retrieval modules

In this section the architecture and the algorithms of the p,T retrieval program are described. Whenever not explicitly declared, the first character of variable names is chosen according to this convention:

INTEGER*4	n (1)
INTEGER*4	I
REAL*8	r (2)
REAL*8	d
COMPLEX*8	c
COMPLEX*16	x
CHARACTER	s
LOGICAL	l

Note:
1) Probably it is possible to use this variable as INTEGER*2 without degrading the numerical accuracy of the program.
2) Probably it is possible to use this variable as REAL*4 without degrading the numerical accuracy of the program.

Each section refers to one main module containing many sub-modules. The sub-module with the same name of the main module is the first called by each main module and contains the main calling tree. Each sub-module description contains:

1. The tree of the calls
2. A short description
3. A table containing the sorted list of all the variables of the module interface. The subscripted variables are be modified by the call. Variable description refers to the use of the variable inside the module. For a detailed description of the variables see section 2.3
4. A description of algorithms used inside the module.

Only the FWDMDL module contains more than one main module description.
Sections 2.2.18, 2.2.19, 2.2.20, 2.2.21, 2.2.22 and 2.2.23 contain a detailed description of the external main modules (i.e. shared modules not included in the high level flow diagram and called by more than one main module).

2.2.1 INPUT_PT

```
INPUT_PT
    |-----R_OBSERV_PT
    |       |-----SKIP_PT +
    |       |-----BLIND_PT *
    |       |-----SKIP_PT +
    |       |-----BLIND_PT *
    |       |-----SKIP_PT +
    |       |-----BLIND_PT *
    |       |-----SKIP_PT +
    |       |(----BLIND_PT *
    |       |(----SKIP_PT +
    |-----R_SETTINGS_PT
    |       |-----SKIP_PT +
    |-----R_MWOCCMAT_PT
    |       |-----SKIP_PT +
    |       |(----BLIND_PT *
    |       |((---SKIP_PT +
    |-----R_INALT_PT
    |       |-----SKIP_PT +
    |       |-----BLIND_PT *
    |       |-----SKIP_PT +
    |-----R_INPRES_PT
    |       |-----SKIP_PT +
    |-----R_INTEMP_PT
    |       |-----SKIP_PT +
    |-----R_INCONT_PT
    |       |-----SKIP_PT +
    |-----UPLIMIT_PT +
    |-----R_SPECT_PT
    |       |(----SKIP_PT +
    |-----WMOL_PT +
    |-----INIGAS_PT ]
    |       |(((--BLIND_PT *
    |       |((((-BLIND_PT *
    |-----R_INVMR_PT
    |       |(----SKIP_PT +
    |-----R_APOD_PT
            |-----SKIP_PT +
            |-----BLIND_PT *
```

Description: This is the subroutine which reads the input files used by p,T retrieval module. This subroutine makes also congruity checks on data and used parameters, builds memory structures that will be used later in the program and performs some initialisations. It makes available to the other routines all the variables read from input files.
The structure of this module is straightforward and it is not worth to describe in detail the operations therein performed. The used FORTRAN code, plenty of comments and self explanatory is reported in AD7. For completeness, we just list in the following the various small sub-modules used by the ‘input_pt’ subroutine.

2.2.1.1 R_OBSERV_PT

Description: subroutine used to read file of observations.
See the FORTRAN source code in [AD7]

2.2.1.2 R_SETTINGS_PT

Description: subroutine used to read file of settings
See the FORTRAN source code in [AD7]

2.2.1.3 R_MWOCCMAT_PT

Description: subroutine used to read file of microwindows and occupation matrix
See the FORTRAN source code in [AD7].

2.2.1.4 R_INALT_PT

Description: subroutine used to read file of the altitudes of initial profiles
See the FORTRAN source code in [AD7]

2.2.1.5 R_INPRES_PT

Description: subroutine used to read file of initial P profile
See the FORTRAN source code in [AD7].

2.2.1.6 R_INTEMP_PT

Description: subroutine used to read file of initial T profile
See the FORTRAN source code in [AD7]

2.2.1.7 R_INCONT_PT

Description: subroutine used to read the file of initial continuum profiles
See the FORTRAN source code in [AD7]

2.2.1.8 R_SPECT_PT

Description: subroutine used to read file of spectroscopic data
See the FORTRAN source code in [AD7]

2.2.1.9 WMOL_PT

Description: This subroutine is used in order to initialise of the molecular weights of the isotopes
See FORTRAN source code in [AD7].

2.2.1.10 INIGAS_PT

Description: Initialisation of the variables *'igas, igashi, igasmw, igasnr'* that define the two internal gas codes.

Variables exchanged with external modules:

Name:	Description:
nselmw	total number of selected microwindows
iline	number of lines in each microwindow
icode	HITRAN molecular code for each line of each MW
imaingas	HITRAN code of the main gas of the retrieval (=2 in the case of p-T retrieval)
igas	number of different gases for actual retrieval
igashi	HITRAN code number for each global gas number
igasmw	number of gases to be considered for each mw
igasnr	global gas number for the local gas number of each Mw

Module structure:

Begin loop 1 over all microwindows
Begin loop 2 over all lines of the actual Mw
1. Calculation of the output variables
end loop 2
end loop 1

Detailed description:

loop 1 over all microwindows
jmw=1→*nselmw*

loop 2 over all lines of the actual Mw
kline=1→*iline(jmw)*

1. Calculation of the output variables

Three different types of gas codes are distinguished inside the program:

1. the HITRAN code which is represented by the variable *icode* that attaches to each line of each microwindow the HITRAN code number of the gas.
2. the global gas code of the retrieval which is going from 1 to *igas*, the number of different gases that have to be considered for the retrieval. In this code the number 1 always belongs to the main gas of the retrieval, i.e. in the case of p-T retrieval 1 refers to CO_2. This numbering is connected to the HITRAN gas code by the vector *igashi*, which gives to each global gas number the HITRAN code number, i.e.:
   HITRAN gas number = *igashi(* global gas number *)*
3. the local gas code of each microwindow (*jmw*) which is going from 1 to *igasmw(jmw)*, the number of different gases that have to be considered for each microwindow. As in the global gas code, the number 1 belongs to the main gas of the retrieval (i.e. CO_2 for p-T retrieval). This numbering is connected to the global gas code by the matrix *igasnr* which gives to each local gas number of each microwindow the global number of the gas, i.e.:
   global gas number = *igasnr(*local gas number, *jmw)*

While *icode* is initialised during reading the line data base, the variables *igas*, *igashi*, *igasmw*, and *igasnr* are set up in this module by using the information of *icode*.
First, the variables *igashi* and *igasnr* are initialised so that the main gas of the retrieval is number 1 in the local and the global code:

*igashi(*1*)=imaingas* (=2 for p-T retrieval)
*igasnr(*1,*jmw)*=1

Then, for each line *kline* of each microwindow *jmw* it is checked, if the related gas (given by *icode(kline,jmw)*) is already included in the global and local gas codes. If this is not the case *igas, igashi, igasmw,* and *igasnr* are enhanced by 1.
During this procedure it is checked that in each microwindow there is at least one line of the main gas. It is also tested that *igas* becomes not larger than *imxgas* and *igasmw(jmw)* ≤ *imxgmw* for each microwindow *jmw*. If one of these conditions is not fulfilled, the program is stopped.

Example:

The inputs are:

Two microwindows are considered for the retrieval: *nselmw*=2.
3 lines in the 1st Mw and 4 lines in the 2nd: $iline = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$.
In the 1st Mw there are CO_2 (HITRAN code =2) and H_2O (HITRAN code =1) lines and in the 2nd Mw there is CH_4 (HITRAN code =6), CO_2, and H_2O: $icode = \begin{bmatrix} 2 & 6 \\ 1 & 2 \\ 2 & 1 \\ & 1 \end{bmatrix}$.
The main gas of the retrieval is CO_2: $imaingas = 2$.

The results of **inigas** are:

Total number of different gases: $igas = 3$.
HITRAN code for each global gas number: $igashi = \begin{bmatrix} 2 \\ 1 \\ 6 \end{bmatrix}$.
Number of different gases per microwindow: $igasmw = \begin{bmatrix} 2 \\ 3 \end{bmatrix}$.
Global gas number for the local gas number of each Mw: $igasnr = \begin{bmatrix} 1 & 1 \\ 2 & 3 \\ & 2 \end{bmatrix}$.

2.2.1.11 R_INVMR_PT

Description: subroutine used to read file of initial VMR profiles
See the FORTRAN source code in [AD7]

2.2.1.12 UPLIMIT_PT

Description: It controls whether the variable 'rulatm', that represents the upper limit of the atmosphere, is both greater than the highest simulated geometry and less than the highest point of the initial guess profiles.

Variables exchanged with external modules:

Name:	Description:
rzprof	vector of altitudes Z to which rtprof, rpprof and rvmrprof are referred [km]
rztang	vector containing the engineering values of tangent altitudes [km]
rbase	greater base of trapezium of Field of View function [km]
rulatm	upper limit of the atmosphere [km]

Detailed description:
See the FORTRAN source code in [AD7]

2.2.1.13 R_APOD_PT

Description: subroutine used to read the file of apodisation function in the interferogram domain.
See the FORTRAN source code in [AD7]

2.2.1.14 SKIP_PT

Description: subroutine to skip comment lines on read files convention is that: at least one comment line appears before a read statement last comment line starts with a character # in column 1
See the FORTRAN source code in [AD7]

2.2.2 SINVCAL_PT

SINVCAL_PT
|----FOUR1_PT*

Description:
The two modules **sinvcal_pt.f** and **sinvcal_mw_pt.f** are strictly linked; they are used to calculate the inverse matrix of the Variance Covariance Matrix of the observations, $\mathbf{VCM}^{-1}$.
The ORM program needs $\mathbf{VCM}^{-1}$, but generally the calculation of $\mathbf{VCM}^{-1}$ is performed knowing **VCM**.
Assuming that all the microwindows are uncorrelated, the Variance Covariance Matrix of the observation **VCM** is a matrix composed of as many quadratic blocks on the diagonal as many microwindows, times the number of limb views considered for each microwindow, are selected for each retrieval. In the retrieval program, we need the inverse of **VCM**, $\mathbf{VCM}^{-1}$.
Due to the particular structure of **VCM**, $\mathbf{VCM}^{-1}$ has the same structure as **VCM**: in particular, each block of $\mathbf{VCM}^{-1}$ is given by the inverse of the corresponding block of the **VCM**.
The calculation of $\mathbf{VCM}^{-1}$ can be performed in two different ways:

- the optimised method, used by module **sinvcal_pt**, calculates directly a block of $\mathbf{VCM}^{-1}$, skipping the step of calculating first VCM (but some corrections have to be applied to each block).
- the non-optimised method, which consists in calculating first the blocks of **VCM** and then inverting them.

In the first case a base block is calculated once for all microwindows, and it is then corrected to properly take in account the correlations on the borders of each particular microwindow. Version 2 of the ORM, unlike Version 1, performs these corrections on each block .
The optimised method can be performed only if all the points of the apodisation function are different from 0 or the microwindow in consideration contains a number of sampling points greater than the number of points considered for apodisation function.
Hence, the calculation of the matrix $\mathbf{VCM}^{-1}$ is performed in two steps:
_first the routine **sinvcal_pt.f** calculates the base block of that matrix in an optimised way; besides, from the input apodisation function in the interferogram domain, it calculates the apodisation function in the spectral domain, necessary for the following step.
_then the module **sinvcal_mw_pt.f** customises the block calculated in **sinvcal_pt.f** routine (or it calculates it if the optimised method is not feasible) for each selected microwindow in the actual retrieval.

The two steps are performed separately, since the first step could be performed, in the final version of the ORM code, in a pre-processor program, because it is independent on which retrieval is being performed.

Variables exchanged with external modules:

Name:	Description:
rapod	real*4 rapod(imxapo) = apodisation function represented in the OPD domain
napod	integer*4: no. of points used to represent apodisation function in OPD domain (rapod)
rzerof	real*8: zero filling expressed as the ratio between measured and transformed interferogram
rapod_sig	real*4 rapod_sigma(imxilc): apodisation function in the spectral domain

ma	
nailsdp	integer*4: no. of points used to represent the apodisation in the spectral domain
rvcmobinvopt	real*4 rvcmobinv(imxi,imxi) = inverse of the VCM of the fitted data points, referred to the largest considered microwindow, calculated in an optimised way. The noise is not included in this matrix.

Algorithm Description

Refer to Sect. 4.5 of [AD6] for the description of the algorithm which drives the computation of the VCM of the fitted spectral data points.

Module structure:

1. Calculation of the measured maximum path difference and other variables.
2. Calculation of the vector *rapodinv=1/(rapod**2)* or redefinition of apodisation function *rapod*.
3. Calculation of *rvcmobinvopt*, only if *rzerof* = 1.
4. Calculation of the apodisation function in the spectral domain *rapod_sigma*.

Detailed description:

1. Calculation of the measured maximum path difference and other variables.

Using *rzerof*, the ratio between the measured maximum path difference *mmpd* and the maximum path difference used for transforming the interferogram (equal to 20 cm), *mmpd* is computed.

$$mmpd = rzerof * 20.$$

Besides the number of points used for performing the subsequent FFTs:

$$nn = (napod-1) * 2$$

is calculated.

Note that *napod* is equal to *(2**n+1)* , with *n* integer (the control is made in **finput_pt.f** routine), and the *napod* points of *rapod* are assumed to represent the points of the apodisation function in OPD domain from $x = 0$ to $x = 20$.

2. Calculation of the vector *rapodinv=1/(rapod**2)* or redefinition of apodisation function *rapod.*

This step is meant to prepare some preliminary variables to be used in the subsequent calculations:

1. if the measured MPD *mmpd* is less than 20 (i.e. if *rzerof* < 1), the apodisation function in the interferogram domain is changed to take into account that a zero-filling to 20 cm path difference has been implicitly performed on the interferogram: in this case all the points corresponding to a path difference greater than *mmpd* are set to 0.
2. If this is not the case, the vector *rapodinv = 1 / rapod**2* is set up.

3. Calculation of *rvcmobinvopt*

This step is performed only if *rzerof* ≥ 0.99, because in the other cases the calculation of the VCM of the observations can be performed only with the not-optimised method (see **sinvcal_mw_pt** module).

- The previously calculated vector *rapodinv* is stored in a proper way into a complex vector. This allows to use the module **four1** for performing the FFT:

  complex*8 *rapodft(imxapo*2)*

rapodft(1) = cmplx*(rapodinv(1),0.)*
begin loop *i=2, ..., (napod-1)*
rapodft(i) = cmplx*(rapodinv(i),0.)*
rapodft(nn-i+2) = cmplx*(rapodinv(i),0.)*
end loop *i.*
rapodft(napod) = cmplx*(rapodinv(napod),0.)*

- The FFT of *rapodft* is then performed using **four1**:
four1(*rapodft,nn,1*)

[Refer to W.H.Press and others: 'Numerical Recipes in FORTRAN', Second Edition, Cambridge University Press, pag.501 for the description of the module **four1**. Note that the FFT of *rapodft* is overwritten in *rapodft.*]

- The vector *realapodft* defined as *realapodft(i)* = REAL*(rapodft(i)/nn)* for *i=1, ..., nn,* is computed.

- the array *rvcmobinvopt* is finally evaluated taking into account that:
_ the matrix is simmetric;
_ the first row is equal to the vector *realapodft*, while each of the other rows is obtained by shifting of one position forward the row just above it.

begin loop I: i=1, ..., imxi
begin loop II: j=i, ..., imxi
rvcmobinvopt(i,j) = realapodft(j-i+1)
rvcmobinvopt(j,i) = rvcmobinvopt(i,j)
end loop I i
end loop II j

4. Calculation of the apodisation function in the spectral domain.

- The vector *rapod* is stored in a proper way into a complex vector. This allows to use the module **four1** for performing the FFT:
complex*8 *rapod_xft(imxapo*2)*
rapod_xft(1) = cmplx*(rapod(1),0.)*
begin loop *i=2, ..., (napod-1)*
rapod_xft(i) = cmplx*(rapod(i),0.)*
rapod_xft(nn-i+2) = cmplx*(rapod(i),0.)*
end loop *i.*
rapod_xft(napod) = cmplx*(rapod(napod),0.)*
- The FFT of *rapod_xft* is then performed using **four1**:
four1(*rapod_xft,nn,1*)
- The vector *realapod_sigma* defined as *realapod_sigma(i)* = REAL*(rapod_xft(i)/nn)* for *i=1, ..., nn,* is computed.
- The vector *rapod_sigma,* containing the apodisation function in the spectral domain, is finally built as required by the routine **sinvcal_mw_pt.f**: it consists of *nailsdp* points corresponding to both the negative and positive frequencies, centred around the 0-frequency point, of the apodisation function.

2.2.3 SINVCAL_MW_PT

SINVCAL_MW_PT
|??---VCMEX_PT]

Description: see description of module **sinvcal_pt**.

Variables exchanged with external modules:

Name:	Description:
rapod_sigma	real*4 rapod_sigma(imxilc): apodisation function in the spectral domain
nailsdp	integer*4: no. of points used to represent the apodisation in the spectral domain
rvcmobinvopt	real*4 rvcmobinv(imxi,imxi) = inverse of the VCM of the fitted data points, referred to the largest considered microwindow, calculated in an optimised way. The noise is not included in this matrix.
nselmw	integer*4: total number of selected microwindows for the actual retrieval
nsam	integer*4 nsam(imxmw): total number of spectral sampling points for the selected microwindows
rzerof	real*8: zero filling expressed as the ratio between measured and transformed interferogram
rvcmobinv	real*4 rvcmobinv(imxi,imxi,imxmw): inverse of the VCM of the observations for each selected microwindow

Algorithm Description
Refer to Sect. 4.5 of [AD6] for the description of the algorithm which drives the computation of the VCM of the fitted spectral data points.

Module structure:
1. Initialisation of some variables
2. Determination of the microwindow with the maximum number of sampling points.
Begin condition 1: the maximum number of sampling points is greater than nailsdp
    3. Calculation of the exact block of the inverse of the VCM of the observations for the widest microwindow.
    4. Building of the 'matrix of the corrections'.
End condition 1
Begin loop 1 over the microwindows
    Begin condition 2: the considered microwindow is not the widest one
        Begin condition 3: the block of the VCM connected with the given microwindow has to be computed in the exact manner
            5. Calculation of the block of the VCM related to the given microwindow with the not-optimised method.
        Else condition 3
            6. Calculation of the block of the VCM related to the given microwindow with the optimised method.
        End condition 3
    End condition 2
End loop 1

Detailed description:

1. Initialisation of some variables
The logical variable *lex* is set to true (in this case all the blocks are calculated with the non-optimised method: in the final version of the program *lex* shall be set to false).
The matrix *rvcmobinv* is set to 0.

2. Determination of the microwindow with the maximum number of sampling point.
Among all the selected microwindows *nselmw* of the actual retrieval, the index *imwmax* of the microwindow with the maximum number of spectral points (the vector *nsam(imxmw)* contains the total number of spectral points in each microwindow), is determined.

Condition 1: the maximum number of sampling points is greater than nailsdp
Steps 3. and 4. are performed only if $nsam(imwmax) > nailsdp$

3. For the widest microwindow the exact VCM^{-1} is built
The calculation of this block is performed by the routine **vcmex_pt** :
vcmex_pt (nsam(imwmax), rapod_sigma, nailsdp, rcinv).

The matrix *rcinv* is stored in the matrix *rvcmobinv.*

4. Building of the 'matrix of the corrections',
The first [*(nailsdp - 1) / 2 +1*] rows of the matrix *rcinv* previously calculated are stored into the matrix *rcor*.
This matrix will be used for correcting the optimised VCM block *rvcmobinvopt* and generating the microwindow customised blocks.

Loop 1 over the microwindows, index imw
$imw = 1 \rightarrow nselmw$

Condition 2: imw is not the widest microwindow
$imw \neq imwmax$
The block of the VCM relative to this microwindow has just been calculated in step 3.

Begin condition 3: the block of the VCM^{-1} connected with the given microwindow has to be computed with the not-optimised method
If at least one of the following conditions is verified:
- $nsam(imw) \leq nailsdp$
- $rzerof < 0.99$
- $lex = true$

the calculation of the block relative to the considered microwindow is performed with the not-optimised method.

5. Calculation of the block of the VCM^{-1} connected with the given microwindow with the not-optimised method.
The calculation of this block is performed by the routine **vcmex_pt** :
vcmex_pt (nsam(imwmax), rapod_sigma, nailsdp, rcinv).

The matrix *rcinv* is stored in the matrix *rvcmobinv.*

6. Calculation of the block of the VCM^{-1} connected with the given microwindow with the optimised method.
The matrix *rvcmobinvopt* is stored in the matrix *rvcmobinv*.
rvcmobinv (i, j, imw) = rvcmobinvopt (i, j), i=1,....,nsam (imw); j=1,....,nsam (imw)

Then the first and the last rows and columns of this block are taken over by the ones contained in the matrix *rcor*.

Begin loop on *i: i =1, ..., (nailsdp-1)/2+1*
Begin loop on *j: j= i,....., nsam(imw)*
rvcmobinv(i, j, imw) = rcor(i,j)
rvcmobinv(j, I, imw) = rvcmobinv(j, i, imw)
rvcmobinv(nsam(imw) - i + 1, nsam (imw)-j+1, imw)= rvcmobinv(i, j, imw)
rvcmobinv(nsam(imw) - j + 1, nsam (imw)-i+1, imw)= rvcmobinv(i, j, imw)
End loop on *j*
End loop on *i*

2.2.3.1 *VCMEX_PT*

VCMEX_PT]
|((((+VINVCAL_PT *

Description: this module calculates the block, related to a particular microwindow, of the inverse of the VCM of the observations, VCM^{-1} , calculating first the VCM of the observations and then inverting that matrix.

Variables exchanged with external modules:

Name:	Description:
nsam1	integer*4 number of spectral points of the considered microwindow
rapod_sigma	real*4 rapod_sigma(imxilc): apodisation function in the spectral domain
nailsdp	integer*4: no. of points used to represent the apodisation in the spectral domain
rcinv	real*8 rcinv(imxi,imxi): block of the inverse of the VCM of the observation
rvcmobinvopt	real*4 rvcmobinv(imxi,imxi) = VCM^{-1} of the fitted data points, referred to the largest considered microwindow, calculated in an optimised way. The noise is not included in this matrix.

Module structure:
1. Calculation of the block of the VCM
2. Calculation of the block of VCM^{-1}

Detailed description:

1. Calculation of the block of the VCM
The VCM of the observations *rvcm* is calculated performing the matrix product $rj \cdot rj^T$, where *rj* is the matrix associated to the linear operation of apodisation, i.e. the convolution with the apodisation function.

- The matrix *rj*, whose dimensions are *nsam1·(nailsdp+nsam1-1)*, is built as follows:

_ the array *rj(imxi, imxj)* is set to 0.
_ the first *nailsdp* positions of the first row are filled with the apodisation function *rapod_sigma*
_ each of the other rows is obtained by shifting ahead of one position the row just above it.

Begin loop on i: i=1, ..., nsam1
Begin loop on j: j=1, ..., nailsdp
rj(i, j+i-1) = rapod_sigma (j)
End loop on j
End loop on i

- *rvcm* is finally obtained by performing the matrix product $rj \cdot rj^T$, where rj^T represents the transpose matrix of *rj*.

2. Calculation of the block of VCM^{-1}

The inverse of the matrix *rvcm*, *rcinv*, is computed by the module **vinvcal_pt.f**:
vinvcal_pt(*rvcm, nsam1, rcinv*)

2.2.4 VINVCAL_PT

VINVCAL_PT
|-----JACOBI1_PT]

Description: This subroutine calculates VCM^{-1} using the singular value decomposition method as explained in AD6. It uses the Numerical Recipes subroutine 'jacobi' in order to compute eigenvectors and eigenvalues of the matrix A.
It is exactly the same as routine **ainvcal_pt**, the only difference is that all the matrices inside the routine are dimensioned in a different way (*imxi · imxi* instead of *imxtop · imxtop*). This avoids to work with over-dimensioned matrices.

Variables exchanged with external modules:

Name:	Description:
rvcm	Block of the VCM of the observations, related to the considered microwindow
nsam1	Dimension of the considered block of the matrix rvcm (total number of spectral points of the given microwindow)
rcinv	Inverse of rvcm (or 'generalised' inverse)

2.2.5 OCCUSIM_PT

OCCUSIM_PT
|-----BLIND_PT *

Description: This module prepares some quantities useful in the subsequent calculations; in particular:

- builds the altitude grid (*rzsi*) corresponding to the tangent altitudes of the spectra that will be simulated,
- calculates a logical vector (*lfitgeo*) that indicates which of the altitudes *rzsi* correspond to a point where the temperature profile is fitted,
- sets-up the integer matrix *iocsim*, that specifies which of the simulated spectra correspond to a limb measurement,
- calculates *ilimbmw*, *irowmw*, and *iobs.*

Variables exchanged with external modules:

Name:	Description:
rztang	rztang(imxgeo) = vector containing the engineering values of tangent altitudes.
ilimb	ilimb = number of measured geometries
imaingas	imaingas = Hitran code of the main gas of the retrieval
lokku	lokku(imxgeo,imxmw) = occupation matrix used for the selection of operational MW's
nselmw	nselmw = total number of selected microwindows for the retrieval
lfit	lfit(imxlmb) = logical vector that identify the tangent altitudes which correspond to a fitted point in the profiles
rbase	rbase = greater base of trapezium of Field of View function
rzsi	rzsi(imxgeo) = tangent altitudes of the geometries to be simulated
nsam	nsam(imxmw) = n. of sampling points in each MW (coarse grid)
igeo	igeo = number of simulated geometries
lfitgeo	lfitgeo(imxgeo) = logical vector which identifies the simulations which correspond to a fitted point in the T profile, among all the simulations to be performed.
ipar	ipar = number of parameter-levels (i.e. N. of elements of rzpar vector)
iocsim	iocsim(imxgeo,imxmw) = occupation matrix for the simulations to performed: = 0 no simulation required, = 1 simulation required without FOV = 2 simulation required with FOV
ilimbmw	ilimbmw(imxmw) = number of valid geometries per microwindow (number of 2 in each column of iocsim)
irowmw	irowmw(imxmw) = the row of the Jacobian matrix where the actual microwindow starts
iobs	iobs = total number of data points to be fitted
rzpar	rzpar(imxlmb) = vector of the altitudes where the temperature profile is fitted

Module structure

The computations proceed along the following steps:

1. Establishes which is the number of simulations to be performed (*igeo*).
2. Setup of *rzsi*, *lfitgeo* and *rzpar*, calculation of *ipar*,
3. building of *iocsim*,
4. calculation of *ilimbmw*,
5. calculation of *irowmw* and *iobs.*

Detailed description

1. The total number of limb-observations to be simulated *igeo* is set equal to *igeo = ilimb + 4* in the case of H2O retrieval (*imaingas = 1*) in the other cases *igeo* is set equal to *igeo = ilimb+2*. The extra simulations, in addition to the measured sweeps, are performed in order to take into account the finite FOV effect.

2. Setup of *rzsi*, *lfitgeo* and *rzpar*, calculation of *ipar:*

rzsi, lfitgeo:

m = 0
lfitgeo(i) is initialised to .FALSE. for *i=1, ..., imxgeo*

m = m + 1
rzsi(m) = rztang(ilimb) - (rbase/2) (below the lowest sweep)
for *i=1, ..., ilimb*
m = m + 1
rzsi(m) = rztang(i) (in correspondence of the sweeps)
if *lfit(i)*=.TRUE. then set *lfitgeo(*)*=.TRUE.
end loop
m = m + 1
rzsi(m) = rztang(1) + (rbase/2) (above the highest sweep)
for *k=1, ...,igeo-ilimb-2*
m = m + 1
rzsi(m)=(rztang(ilimb-k+1)+rztang(ilimb-k))/2 (within the sweeps)
End loop.

⇒ The altitudes contained in *rzsi* are then re-sorted so that small values of the index correspond to lower altitudes. Note that *rzsi* is the only vector sorted starting from low altitudes.

⇒ The elements of *lfitgeo* are then re-sorted in inverse order with respect to the elements of *rzsi.* There is one-to-one correspondence between the vectors *lfitgeo* and *rzsi* but the correspondence is between elements which have reversed indexes; the first element of *lfitgeo* (*lfitgeo(1)*) refers to the last element of *rzsi* (*rzsi(igeo)*) and so on.

⇒ Setup of *rzpar. rzpar* contains the altitudes that correspond to the points where the VMR profile is fitted, i.e. the values of *rztang* for which *lfit* is TRUE. This vector is sorted starting from high altitudes.

⇒ Setup of *ipar: ipar* is the number of TRUE elements of the vector *lfit*, i.e. is the total number of points where the VMR profile is fitted.

3. Calculation of *iocsim*.
Let's consider an integer matrix *iocsim* having *igeo* rows and *nselmw* columns; all the elements are initialised to zero.
Each row of this matrix corresponds to one altitude where a simulation will be computed; i.e. each row corresponds to one element of the vector *lfitgeo* (direct index correspondence) and to one element of *rzsi* (reverse index correspondence).

Each column of this matrix corresponds to a selected microwindow.
Let's consider the rows of *iocsim* that correspond to altitudes where a sweep has been measured (altitudes which correspond to the elements of the vector *rztang*). Now, there is one-to-one correspondence between these rows and the rows of the occupation matrix *lokku*, so we review the elements of these rows of *iocsim* and we set equal to 2 the elements which correspond to a TRUE element in *lokku*. In practice we set equal to 2 the elements of *iocsim* whenever we want to perform simulations including FOV effect.
Afterwards, since for the simulation of spectra including FOV effect we interpolate the spectra in the altitude domain (see description of FOV subroutine), extra spectra have to be simulated in addition to the ones which correspond to measurements. Which are the required extra spectra is established using a series of conditions that is explained in the description of FOV module. The same conditions are reviewed in OCCUSIM module and, whenever (altitude, microwindow) a simulation is required, the corresponding element in *iocsim* matrix is set equal to 1.

4. Calculation of *ilimbmw:*
ilimbmw(j) is defined as the number of TRUE elements in the *j-th* column of *lokku*, i.e. *ilimbmw(j)* indicates the total number of sweeps in which the microwindow *j* is fitted. Of course, the vector *ilimbmw* has *nselmw* elements.

5. Calculation of *irowmw* and *iobs*:
irowmw is defined as:

$$irowmw(j) = 1 + \sum_{k=1}^{j-1} nsam(k) \cdot ilimbmw(k) \qquad \text{for } j=1,..., nselmw$$

where *nsam(k)* is the number of sampling data points of microwindow *j* (in the 0.025 cm^{-1} frequency grid). In practice, *irowmw* indicates the row of the Jacobian matrix *rjacob*, where the data related to the actual microwindow start.
iobs is the total number of observed spectral data points that are fitted in the retrieval and is therefore computed as:

*iobs = [irowmw(nselmw) - 1] + [nsam(nselmw) * ilimbmw(nselmw)]*.

2.2.6 TCGEO

Description:
Determination of the vectors *igeotder* and *igeocder*, that relate to each simulated geometry the parameter levels which have to be considered for the derivatives (i.e. the parameter levels where a change of the parameter influences the spectrum).

Variables exchanged with external modules:

Name:	Description:
lfitgeo	logical vector that is true if a simulated geometry is also a parameter-level
ipar	number of parameter levels
igeo	number of simulated geometries
lfit	logical vector that is true if an observational level is also a parameter level
ilimb	number of measured geometries
nucl	nucl+1 = upper parameter level for continuum fit
igeotder	for each simulated geometry *j* the highest (*igeotder*(*j*,1)) and lowest (*igeotder*(*j*,2)) parameter level which has to be considered for the temperature-derivatives
igeocder	for each simulated geometry *j* the highest (*igeocder*(*j*,1)) and the lowest (*igeocder*(*j*,2)) parameter level which has to be considered for the continuum-derivatives

Module structure:
1.Calculation of *igeotder*
2.Calculation of *igeocder*

Detailed description:

1.Calculation of *igeotder:*
The highest parameter level influences all simulated geometries:
For $1 \le jgeo \le igeo$:
$igeotder(jgeo,1) = 1$

The lowest parameter level that influences the simulated geometry is the one of the geometry itself, if the geometry is also a parameter level. If the geometry is no parameter level, the parameter level below is used (if it exists).

For $1 \le jgeo \le igeo$:

Count the parameter levels up to *jgeo*:
mpar=0
For $1 \le kgeo \le jgeo$:
if [*lfitgeo(kgeo)*]: *mpar*=*mpar*+1

If the geometry *jgeo* is a parameter level:
if [*lfitgeo(jgeo)*]: *igeotder(jgeo,2)=mpar*
if *jgeo* is no parameter level:
else:
if *mpar* is not equal to the total number of parameter levels:

if [*mpar* ≠ *ipar*]: *igeotder(jgeo,2)=mpar*+1
if *mpar* is equal to the total number of parameter levels:
if [*mpar = ipar*]: *igeotder(jgeo,2)=mpar*

2.Calculation of *igeocder:*

2.1 Determination of the highest derivative to be calculated for continuum
kcl=0
Begin loop I on limb observations *jcl=1, ..., ilimb*
Begin condition I: if *lfit(jcl)* = TRUE then
kcl=kcl+1
Begin condition II if *jcl > nucl* then
Begin loop II on simulated geometries *kgeo=1, ..., igeo*
igeocder(kgeo,1) = kcl
End loop II on simulated geometries
go to (**)
End if condition II
End if condition I
End loop I on limb observations
(**) continue

2.2 Determination of the lowest derivative to be calculated for continuum
Begin loop I on the simulated geometries: *jgeo = 1, ..., igeo-1*
mpar = 0
Begin loop II on the simulated geometries: *kgeo = 1, ..., jgeo+1*
if *lfitgeo(kgeo)* = TRUE *mpar=mpar+1*
End loop II on the simulated geometries
igeocder(jgeo,2) = mpar
End loop I on the simulated geometries

2.2.7 CHBASE_PT

CHBASE_PT]
|((((-LININT_PT *
|((((+ESPINT_PT *
|((((+GRAVITY *

Description: This routine moves the atmospheric profiles into the altitude representation defined by the retrieval. This altitude grid coincides with the tangent altitudes of the measurements in the range covered by the measurements, has an extra point below the lowest measurement and coincide with the grid in which the initial guess profiles are provided in the altitude range between the highest measurement and the upper limit of the atmosphere.

Variables exchanged with external modules:

Name:	Description:
rzprof	rzprof(imxpro) = vector of altitudes Z to which rtprof, rpprof and rvmrprof are referred
rtprof	rtprof(imxpro) = vector of temperature as a function of altitude Z.
rpprof	rpprof(imxpro) = vector of pressure profile as a function of altitude Z.
rvmrprof	rvmrprof(imxpro,imxgas) = matrix of VMR profiles
rcprof	rcprof(imxpro,imxmw) = array containing continuum cross section as a function of altitude and microwindow
ipro	ipro = N. of elements of rzprof, rtprof, rpprof, rcprof, rvmrprof
igas	igas = total number of different gases
nselmw	nselmw = N. of selected microwindows
rztang	rztang(imxgeo) = tangent altitudes of the considered sweeps
ilimb	ilimb = N. of considered sweeps
rzpar	rzpar is not used in this module, is a spare interface.
ipar	ipar is not used in this module, is a spare interface.
rlat	rlat = latitude of the actual limb-scan (deg.)
lfit	lfit(imxlmb) = logical vector which identifies the altitudes where the T profile is fitted, among the altitudes rztang.
rzbase	rzbase(imxpro) = altitude of the base-levels
rtbase	rtbase(imxpro) = temperature of the base levels
rpbase	rpbase(imxpro) = pressure on the base-levels
rvmrbase	rvmrbase(imxpro,imxgas) = volume mixing ratio of the gases on the base levels
rcbase	rcbase(imxpro,imxmw) = continuum on the base-levels for each MW
ibase	ibase = number of base-levels
lparbase	lparbase(imxpro) = logical vector which identifies the altitudes where the T profile is fitted, among the altitudes rzbase.

Algorithm Description:
The goal of this module is to change the representation of the initial profiles that are contained in the so called '*prof*' variables. After this routine is completed, the atmospheric profiles represented in the base of the retrieval are available in the '*base*' variables.

Module structure
The module proceeds in the following steps:
1. Calculation of the number (*ibase*) of elements belonging to *rzbase,*
2. building of the altitude grid *rzbase,*

3. computation of the '*base*' profiles,
4. setup of *lparbase.*

Detailed description

1. Calculation of *ibase,* the number of elements of the *rzbase* vector.

Let's call *ihigh* the index of the element of the vector *rzprof* which verifies:

$$rzprof(ihigh+1) \le rztang(1) < rzprof(ihigh)$$

then *ibase* is computed as:

$$ibase = ihigh + ilimb + 1$$

2. Building of the altitude grid *rzbase*:

- *rzbase(i), i = 1, ..., ihigh* are equal to the elements of *rzprof* which are located above the highest tangent altitude of the scan sequence (i.e. above *rztang(1)*)
- *rzbase(i), i = ihigh+1, ..., ihigh+ilimb* is set equal to the tangent altitudes of the limb-measurements that are contained in *rztang*.
- set *istart = 1*
- *rzbase(ibase) = rzprof(ipro - istart)*

The elements of *rzbase* (as in the case of *rzprof*) are sorted starting from high altitudes (i.e. small value of the index = high altitude).

3. Calculation of the *'base'* profiles.

The '*base'* profiles are the '*prof*' profiles interpolated to the altitudes of '*rzbase'*:

- *rtbase* is obtained by linearly interpolating in the altitude domain *rtprof* (referred to the altitude grid *rzprof*), to the altitudes *rzbase* (**LININT_PT** subroutine is used for this purpose).
- *rvmrbase* is obtained by linearly interpolating in the altitude domain *rvmrprof* (referred to the altitude grid *rzprof*), to the altitudes *rzbase*. This operation is performed for all the considered - gases (*igas*). (**LININT_PT** subroutine is used).

Note: please note that at those altitudes of *rzbase* which coincide with an altitude of *rpbase,* the linear interpolation is not strictly required: in this case the '*base*' profiles can directly be set equal to the corresponding values of the '*prof*' profiles.

- *rpbase* is computed using hydrostatic equilibrium law.

  At the lowest altitude it is assumed that:

  $$rpbase(ibase) = rpprof(ipro - istart)$$

  then for the subsequent altitude steps the used formula is:

  $$rpbase(i-1) = rpbase(i) * \exp\left[- \frac{rmovr * gravity(\bar{z}_i, rlat) * \Delta z_i}{\bar{t}_i} \right]$$

  where:

  $$\bar{z}_i = [rzbase(i-1) + rzbase(i)] / 2$$
  $$\Delta z_i = rzbase(i-1) - rzbase(i)$$
  $$\bar{t}_i = \left[rtbase(i-1) + rtbase(i) \right] / 2$$

  rmovr is a parameter, (see description of parameters.inc)

  $gravity(\bar{z}_i, rlat)$ is computed by '**GRAVITY**' function,

  i ranges from *ibase,* ..to..., 2.

Note that all the '*base*' profiles are referred to the altitude grid defined by *rzbase.*

- Continuum base profiles *rcbase* are obtained as follows:
  - Above the measurements:
    Begin loop I *jmw=1, ..., nselmw*
    Begin loop II *l=1, ...ihigh*
    rcbase(l,jmw)=rcprof(l,jmw)
    End loop II
    End loop I
  - Within the measurements linear interpolation in pressure domain is used.
    Begin loop I *l=ihigh+1, ihigh+ilimb*
    Begin loop II *jmw=1,nselmw*
    call linp_pt*(rpprof,rcprof(1,jmw),ipro,rpbase(l),rcbase(l,jmw))*
    end loop II
    end loop I
  - Below the lowest measurement:
    Begin loop I *jmw=1, ...,nselmw*
    rcbase(ibase,jmw)=rcprof(ipro-istart,jmw)
    End loop II

4. Setup of *lparbase*.
lparbase is a logical vector which identifies the altitudes where the temperature profile is fitted, among the altitudes contained in *rzbase*.
All the *ibase* elements of *lparbase* are first initialised to .FALSE.
The elements of this vector are then reviewed:

for *i*=1 to *ibase*
if an index *j* exists so that: *rzbase(i) = rztang(j)* and *lfit(j)* = .TRUE. then
set *lparbase(i)* = .TRUE.
end if
end loop.

2.2.8 FAILS_PT

FAILS_PT]
|-----BLIND_PT *

Description: This subroutine performs the interpolation of AILS function (*rails*), that is provided at the coarse grid points (step *dstep*), at the points of the frequency fine-grid (step *delta*). The results are stored in *rils* and are used in module CONV as convolving function to pass from the atmospheric spectrum calculated on the fine grid,
to the simulation of the observed spectrum calculated on the coarse grid. The subroutine also computes the number of fine-grid points to add on each side of the spectral intervals to be simulated (in order to reduce truncation effects within the CONV module) and two other quantities that are needed within the CONV module, they are: *a*)the number of sampling points resulting from the

interpolation process, *b*) the ratio between the fine-grid step and the summation over the values of the resulting function.

Variables exchanged with external modules:

Name:	Description:
nselmw	number of selected microwindows
nailsdp	number of AILS data points
nrd	ratio between the frequency steps of the coarse and the fine grid
rails	rails(imxapo,imxmw):apodised instrument line shape for all selected MWs
delta	distance between fine-wavenumber grid points [cm-1]
dstep	distance between coarse-wavenumber grid points [cm-1]
rils	rils(imxils,imxmw: instrument-line-shape function in the frequency fine-grid
iadd	number of fine-wavenumber grid points to be added on both sides of each microwindow (due to the ils-convolution)
nils	number of elements of rils
rintils	ratio between the frequency step approximating infinitesimal spectral resolution and the summation of the interpolated-AILS-function values

Algorithm Description

The interpolation process is realized by making a convolution with a *sinc* function that is computed inside the subroutine. For the purpose the *sinc* function needs to:

- be sampled on the fine grid,
- have a frequency extension twice that of the input AILS,
- have the unity value at the central frequency,
- have the zero values separated by the coarse grid step.

Module structure

The module proceeds in the following steps:

1. Calculation of the *sinc* function,
2. Convolution between the input AILS and the *sinc* function providing the result in the fine frequency grid.
3. Calculation of the ratio between the fine-grid step and the integral of the function resulting from convolution.

Detailed Description

1. Calculation of the *sinc* function

- the number of sampling points of the AILS represented in the fine grid is computed as:

$$nils=((nailsdp-1)*dstep)/delta+1$$

- the number of points to add on each side of the spectral intervals to be simulated

$$iadd=(nils-1)/2$$

- the maximum frequency extension of the *sinc* function is computed as:

$$dels=(nils-1)*delta$$

- the *sinc* function is computed from $\sigma = -dels$ to $\sigma = +dels$ at frequency steps = *delta* as:

$$sinc = \sin(arg)/arg$$

where:

$$arg = \sigma * \pi / dstep.$$

The singularity *sinc* = 1 applies for *arg* = 0.

2. The convolution integral between the input AILS and the *sinc* function is computed, at the i^{th} frequency, as:

$$rils_i = \sum_{j=1}^{nailsdp} rails_j * sin\ c_k$$

where index *k* starts from *nils-i+1* and is incremented by steps of *nrd* in correspondence of each increment to *j*.
The computation of $rils_i$ is repeated for values of *i* going from 1 to *nils*.

3. Along the loop over *i* at step 2 the suitable summation over the computed values $rils_i$ is built up with the purpose of calculating (at the end) the ratio between the fine-grid step *delta* and this summation.

2.2.9 GRID_PT

GRID_PT]
|(----BLIND_PT *

Description:

- Calculation of the general fine wavenumber grid for all microwindows of the retrieval.
- Check, that all lines inside the Mws are handled as lines (explicit calculation of the line profile) and not as nearby continuum

Variables exchanged with external modules:

Name:	Description:
nselmw	total number of selected microwindows
nsam	number of sampling points in each mw (general coarse grid)
nrd	Ratio dstep/delta between coarse and fine grid step
dstep	distance between general coarse-wavenumber grid points [cm-1] NOTE: the sampling point at frequency=0 has index=1
ifspmw	index of the first sampling point of each MW
iadd	number of fine-wavenumber grid points to be added on both sides of each microwindow (due to the ils-convolution)
delta	distance between general fine-wavenumber grid points [cm-1]
iline	number of lines in each microwindow
dsilin	central line wavenumber
ioutin	flag for each line =1: line-shape has to be calculated at each wavenumber inside the microwindow =2: line is considered as nearby continuum
isigma	number of wavenumber grid points for each Mw

dsigma	general wavenumber fine grid for each Mw

Module structure:

Begin loop 1 over all microwindows
1. Calculation of the number of grid points
2. Calculation of the general fine grid
3. Check on *ioutin*

end loop 1

Detailed description:

loop 1 over all microwindows
jmw=1→*nselmw*

1. Calculation of the number of grid points:
Since the microwindow is enhanced on both sides by *iadd* grid points, the formula for the number of grid points is:

$$isigma(jmw) = \left(nsam(jmw) - 1\right) \cdot nrd + 1 + 2 \cdot iadd$$

Here it has to be checked that *isigma(jmw)* ≤ *imxsig.*

2. Calculation of the general fine grid:
The wavenumber belonging to each general fine grid point is:
($1 \le k \le isigma(jmw)$)

$$dsigma(k, jmw) = \left(ifspmw(jmw) - 1\right) \cdot dstep - iadd \cdot delta + (k - 1) \cdot delta$$

3. Check on *ioutin*:
For each line *kline* ($1 \le kline \le iline(jmw)$) it is checked, that, if the line centre is inside the enhanced microwindow ± *dext* (*dsigma(1,jmw)* - *dext* → *dsigma(isigma(jmw),jmw)* + *dext*) (*dext* is a parameter), *ioutin(kline,jmw)* is equal to 1. If this is not the case *ioutin(kline,jmw)* is set to 1.

2.2.10 GUESSPAR_PT

GUESSPAR_PT]
|(----PTFROMZ_PT *
|(----MWCONT_PT *
|(----FICARRA_PT *
|((---BLIND_PT *

Description: This module builds the initial guess of the vector *rxpar* which contains the parameters that are going to be fitted in p,T retrieval.

Variables exchanged with external modules:

Name:	Description:
rzbase	rzbase(imxpro) = altitude of the base-levels
rtbase	rtbase(imxpro) = temperature at the base levels
rpbase	rpbase(imxpro) = pressure at the base-levels
rcbase	rcbase(imxpro,imxmw) = continuum at the base-levels for each MW
ibase	ibase = number of base-levels
nselmw	nselmw = total number of selected microwindows for the retrieval
rlat	rlat = actual latitude of the measurements (deg.)
rzpar	rzpar(imxlmb) = vector of the altitudes where the temperature profile is fitted
ipar	ipar = number of parameter-levels (i.e. N. of elements of rzpar vector)
rztang	rztang(imxgeo) = vector containing the engineering values of tangent altitudes.
ilimb	ilimb = number of measured geometries
lfit	lfit(imxlmb) = These are logical vectors that identify the levels
lokku	lokku(imxgeo,imxmw) = occupation matrix used for the selection of operational MW's for each observation geometry
lparbase	lparbase(imxpro) = logical vector which identifies the altitudes where the T profile is fitted, among the altitudes rzbase.
rperc	rperc = maximum relative (with respect to rconint) distance between central frequencies of two microwindows which are defined as close-close ones for the definition of continuum emission
rconint	rconint(imxlmb,imxmw) = frequency range around each MW, for each sweep, tangent altitude, in which the continuum can be considered as varying linearly.
rxpar	
itop	itop = total number of parameters to be fitted
icontpar	icontpar = total number of continuum parameters to be fitted
rjaccon	rjaccon(imxpro*imxmw,imxcop) = jacobian matrix for the derivative of the continuum base-level values with respect to the continuum parameters
isaved	isaved(imxsav) = vector containing all the necessary quantities for the reconstruction of continuum profiles performed by *ficarra* subroutine
dstep	dstep = distance between coarse-wavenumber grid points [cm-1]
nsam	nsam(imxmw) = number of sampling points in each MW (general coarse grid)
ifspmw	ifspmw(imxmw) = index of the first sampling point of each MW * NOTE: the sampling point at frequency=0 has index=1
nucl	nucl = number of limb geometries to be skipped before starting continuum fit; numbering starts from top.
lcfit	lcfit(imxgeo,imxmw) = continuum occupation matrix
lccmat	lccmat(imxgeo,imxmw) = logical matrix which identifies altitudes & MWs where the continuum is set equal to the continuum of a nearby MW (close-close MWs).
ifco	ifco = switch for enabling / disabling the fitting of atmospheric continuum and instrumental offset

Algorithm Description

Starting from the initial guess of p,T and continuum profiles, the initial guess of the vector *rxpar* of the fitted parameters is evaluated. The total number of fitted parameters (*itop*) and the total number of continuum fitted parameters (*icontpar*) are evaluated as well.

Detailed description

The module proceeds in the following steps:

- For *j=1,..., ilimb, rxpar(j)* is set equal to the pressures at the tangent altitudes of the sweeps which correspond to fitted points in the temperature profile. These sweeps are identified, among all the fitted sweeps, by a TRUE element in the vector *lfit.* (Sorting is always from top)
- For *j=ilimb+1,...,ilimb+ ipar, rxpar(j)* is set equal to the temperatures at the tangent altitudes of the sweeps which correspond to fitted points in the temperature profile. These sweeps are identified, among all the fitted sweeps, by a TRUE element in the vector *lfit.* (Sorting is always from top)

For the calculation of pressure and temperature at the tangent levels that correspond to fitted points, **PTFROMZ** module is used:

Begin loop I *j=1,ipar*
call ptfromz_pt(*rzbase, rpbase, rtbase, ibase, rlat, rzpar(j), rxpar(j), rxpar(j+ipar)*)
End loop I

- subroutine **MWCONT_PT** is called: this subroutine computes the vector *rcpar* of the continuum parameters to be fitted and *icontpar* that is the total number of continuum fitted parameters. The integer vector *isaved* is also computed.
- subroutine **FICARRA_PT** is called: it rebuilds continuum profiles starting from the vector *rcpar* and the integer vector *isaved,* it computes also the jacobian matrix *rjaccon* which contains the derivatives of the different points in the continuum 'base' profiles with respect to the continuum fitted parameters.
- For *j=ilimb+ipar+1, ..., ilimb+ipar+icontpar, rxpar(j)* is set equal to *rcpar(j-ilimb-ipar)*
- For *j=ilimb+ipar+icontpar+1, ..., ilimb+ipar+icontpar+nselmw, rxpar(j)* is set equal to zero. These elements of *rxpar* refer to the instrumental offsets whose initial guess is supposed to be equal to zero.
- *itop* is then computed accordingly to:
  if *ifco* = 2 ----> *itop = ilimb + ipar + icontpar + nselmw*
  if *ifco* = 1 ----> *itop = ilimb + ipar + icontpar*
  if *ifco* = 0 ----> *itop = ilimb + ipar*
- *itop* is then checked and if *itop > imxtop* (*imxtop* is a parameter contained in 'parameters_pt.inc') a fatal error is produced and the program is stopped (the occurrency of this error is limited to the cases in which the program variables are under-dimensioned).

2.2.11 FWDMDL_PT

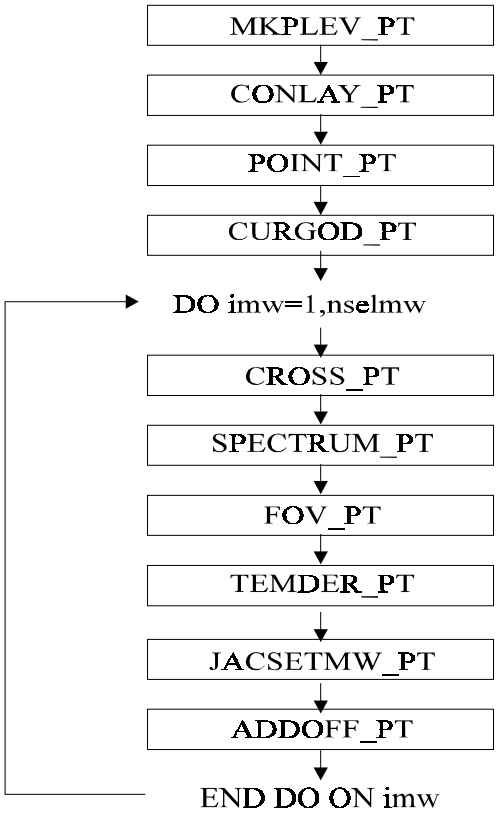


Fig.4: Flow diagram of the forward model used in p,T retrieval.

Description: Using the base-vectors of the atmospheric profiles it calculates the simulated spectra for all observations and the jacobian matrix of these observations with respect to the fitted parameters. The source code of the module 'fwdmdl_pt.f' contains only the flow of the calls reported in Fig.4. The source code of this module has been delivered also to the industry which takes care of MIPAS Level 2 prototype development.

Variables exchanged with external modules:
See section 2.3 for the description of the variables. We report here only the list of the variables at the interface of this module.

subroutine
fwdmdl_pt*(rzsi,igeo,rzbase,rtbase,rpbase,rvmrbase,rcbase,ibase,rulatm,rwmolref,dsigm0, rhw0ref,rmaxtv1,rmaxtv2,rzt12,rhwvar,igas,rexphref,rincz,redfact,rlat,lfitgeo,rdt,lparbase,nselmw, iept,rearad,deps,isigma,dsigma,delta,iocsim,igasmw,ruplin,rlolin,iline,icode,rint0,relow,rhw0, dsilin,ioutin,igasnr,rexph,rwmol,igashi,iiso,ninterpol,nsam,nils,rils,rintils,nrd,iadd,ilimbmw, lokku,nucl,ilimb,igeocder,igeotder,rjaccon,roffs,rbase,rsl,icontpar,ipar,irowmw,ilookupmw,lmgas, smw,nll,npl,rp1l,rdpl,ntl,rt1l,rdtl,ru,rkl,tab,rjacob,rspfov,iigrid,cint,lirrgridmw,igridc,nused1, rsan,ilim)*

2.2.11.1 MKPLEV_PT

MKPLEV_PT
|-----BLIND_PT *
|(----CHECK_PT]
|(?---BLIND_PT *
|((?--CHECK_PT]
|((---BLIND_PT *
|((?--LININT_PT *
|((((-LININT_PT *
|((((+ESPINT_PT *
|((((+GRAVITY *
|((((+BLIND_PT *
|((((+GRAVITY *

Description: builds the layering of the atmosphere for the subsequent calculation of the radiative transfer integral. Calculates also the perturbed temperature profiles which will be used for the computation of the derivatives of the simulated spectra with respect to temperature.

Variables exchanged with external modules:

Name:	**Description:**
rzsi	rzsi(imxgeo) = tangent altitudes of the geometries to be simulated
igeo	igeo = number of simulated geometries
rzbase	rzbase(imxpro) = altitude of the base-levels
rtbase	rtbase(imxpro) = temperature of the base levels
rpbase	rpbase(imxpro) = pressure on the base-levels
rvmrbase	rvmrbase(imxpro,imxgas) = volume mixing ratio of the gases on the base levels
ibase	ibase = number of base-levels
rulatm	rulatm = upper limit of the atmosphere
rwmolref	rwmolref = molecular weigth of the gas that has been selected as a reference for building the levels.
dsigm0	dsigm0 = Centre frequency of the line selected as a reference for building the levels.
rhw0ref	rhw0ref = half-width of the line selected as a reference for building the levels.
rmaxtv1	rmaxtv1 = max. allowed temperature variation (K) between two neighbouring levels, when the lower level is located below rzt12.
rmaxtv2	rmaxtv2 = max. allowed temperature variation (K) between two neighbouring levels, when the lower level is located above rzt12.
rzt12	rzt12 = altitude (km) where the temperature thresholds rmaxtv1 and rmaxtv2 are exchanged.
rhwvar	rhwvar = max. allowed half-width variation of the selected reference line between two neighbouring levels.
igas	igas = total number of different gases
rexphref	rexphref = exponent for the calculation of Lorentz h-w for the line

	selected as a reference for building the levels.
rincz	rincz = guess altitude increment (km) used for building the levels above the highest simulated geometry.
redfact	redfact = reduction factor applied to 'rincz' when it produces not acceptable P levels above the highest simulated geometry.
rlat	rlat = actual latitude (degrees)
lfitgeo	lfitgeo(imxgeo) = logical vector which identifies the simulations which correspond to a fitted point in the T profile, among all the simulations to be performed.
rdt	rdt = temperature increment (K) used to perturb temperature profiles.
rzmod	rzmod(imxlev) = heights of model levels used for the radiat. transf. calc.
rpmod	rpmod(imxlev) = pressure on model levels used for the radiat. transf. calc.
rtmod	rtmod(imxlev) = temperature on model levels used for the radiat. transf. calc.
rmrmod	rmrmod(imxlev,imxgas) = volume mixing ratio for each gas considered in actual retrieval on model levels used for rad. tra. calc.
ilev	ilev = number of model levels (for rad. trans. calculation)
itglev	itglev(imxgeo) = number of the tangent-level for each geometry
rzmodpert	rzmodpert(imxlev,imxlmb) = perturbed altitude grids after the perturbation of temp. profiles.
rtmodpert	rtmodpert(imxlev,imxlmb) = perturbed temperature profiles, used for the calculation of T derivatives
iderlay	iderlay(imxlmb,3) highest (x,1), lowest (x,3) and middle (x,2) (the one directly above the 'derivated' layer) which is affected by each derivative (imxlmb refers to the parameter-levels)
ipar	ipar = number of altitudes where the temperature profile is fitted.

Module structure

The module proceeds along the following steps:

1. Building of the levels located between the lowest and the highest simulated geometries
2. building of the levels located above the highest simulated geometry,
3. interpolation of temperature and VMR profiles to the altitude levels generated in steps 1. and 2., determination of pressure at the generated levels,
4. calculation of *itglev,*
5. generation of perturbed temperature profiles,
6. calculation of the perturbed altitude grids *rzmodpert,*
7. calculation of *iderlay.*

Detailed description

Step 1: Building of the altitude levels located between the lowest and the highest simulated geometries.

First of all the altitudes *rzsi* (tangent altitudes of the simulated geometries) are taken as levels (main-levels). Then, starting from low altitudes (from *rzsi*(1)) the couples of neighbouring main-levels are processed by **CHECK** subroutine which establishes, by checking pre-defined criteria, whether the two considered levels can be accepted. If the two levels are accepted, then the next couple of levels is checked, otherwise sub-levels are generated using the following procedure:

Let's call *rz1* and *rz2* the altitudes of the two considered main-levels that cannot be accepted,

isublev = 1 (sub-level index)

(**) *logic* = TRUE

start loop on sub-layers lying between the two considered main-levels: *j=1, ...,isublev+1*

rz1n = rz1 + (j-1)[(rz1-rz2)/(isublev+1)]*

rz2n = rz1 + j[(rz1-rz2)/(isublev+1)]* (Generated sub-level)

CHECK the couple of levels *rz1n, rz2n*: the result of **CHECK** module is stored in the logical variable *lcheck*

logic = logic .and. lcheck

end loop

if *logic* = TRUE then accept the generated sub-levels and proceed to consider the next couple of main-levels.

if *logic* = FALSE then set *isublev = isublev + 1* and proceed to (**).

After this step, each tangent altitude of the simulated geometries has associated its own level; furthermore, evenly-spaced levels can exist between the tangent altitudes. The generated sub-levels allow to properly model the atmosphere also in the regions where the atmospheric properties have a large variation in the scale of the distance between the tangent altitudes of two neighbouring simulated geometries.

Step 2: building of the levels located above the highest simulated geometry.

Let's start from the tangent altitude of the highest simulated geometry *rzsi(igeo)*. New levels located above *rzsi(igeo)*are generated using the following algorithm:

ifails = 0

rz1 = rzsi(igeo)

(+) *rz2 = rz1 + rincz*

rz1 and *rz2* are then processed by **CHECK**:

if the check is successful then:

rz2 is accepted as a new level,

rz1 = rz2,

rincz is set equal to its initial value,

ifails = 0,

proceed to (+).

else:

ifails = ifails +1

*rincz = rincz / (redfact * ifails)*

proceed to (+)

end if

The above procedure is stopped when the new generated level is higher than the upper limit of the atmosphere increased of the value of *rincz* (*rulatm+rincz)*; the level higher than (*rulatm+rincz)* is not included in the generated set of levels, but the check with (*rulatm+rincz)* instead of *rulatm* assures that the layering is built on the overall range of the atmosphere to be considered.

All the obtained levels are then sorted starting from high altitudes and recorded in the vector *rzmod(imxlev)*; the total number of generated levels is recorded in the variable *ilev.*

General remark: during steps 1 and 2, whenever a new level is built, the total number of generated levels is checked and if this number is greater than the parameter *imxlev* then, the thresholds *rmaxtv1, rmaxtv2* and *rhwvar* are multiplied by a factor 1.1 and the procedure is restarted from step 1. Before exiting, the procedure restores the initial values of the thresholds, so that next time the module is called, the right value of the thresholds is used. At the last call of the forward model, if different thresholds, with respect to the user-defined ones are used, a warning is produced by the

main program. This feature avoids the production of a large number of levels during the iterations of the retrieval, when the temperature profile can be really distorted with respectet to its real shape, and no very accurate simulations are required.

Step 3: interpolation of temperature and VMR profiles to the altitude levels generated in steps 1. and 2. Computation of pressure at the altitudes *rzmod*.
The following operations are performed at this step:

- For each altitude of the vector *rzmod*, the corresponding temperature *rtmod* is obtained by linear interpolation (in the altitude domain, using **LININT**) between the elements of temperature profile *rtbase(imxpro)* which are referred to the altitudes *rzbase(imxpro)*.
- For each altitude of the vector *rzmod*, the corresponding VMR of all the considered gases (1, ..*igas*), *rvmrmod(imxlev,imxgas)* is obtained by linear interpolation (in the altitude domain, using **LININT**) between the elements of VMR profiles *rvmrbase(imxpro,imxgas)* which are referred to the altitudes *rzbase(imxpro)*.
- Rebuilding of pressure profile i.e. calculation of pressure at the *rzmod* altitudes.
  The pressure corresponding to the lowest level *rzmod(ilev)* is computed using exponential interpolation **(ESPINT)** from the pressure profile *rpbase(imxpro)* which is referred to the altitudes *rzbase(imxpro)*.
  The subsequent elements of *rpmod* are then computed using hydrostatic equilibrium law, as follows:

$$rpmod(i-1) = rpmod(i) * \exp\left[- \frac{rmovr * gravity(\bar{z}_i, rlat) * \Delta z_i}{\bar{t}_i} \right]$$

where:

$\bar{z}_i = [rzmod(i-1) + rzmod(i)] / 2$
$\Delta z_i = rzmod(i-1) - rzmod(i)$
$\bar{t}_i = \left[rtmod(i-1) + rtmod(i) \right] / 2$
rmovr is a parameter, (see description of parameters.inc)
$gravity(\bar{z}_i, rlat)$ is computed by '**GRAVITY**' function,
i ranges from *ilev*, ..to..., 2.

Step 4: calculation of *itglev*.
itglev is an integer vector which indicates, for each simulated geometry, the index of the tangent level. *itglev(i) = j* means that the tangent level of the *i-th* simulated geometry is the level number *j*; remember that the numbering of the levels starts from high altitudes.

Step 5: Generation of the perturbed temperature profiles *rtmodpert(imxlev,imxlmb)*,
The matrix *rtmodpert(imxlev,imxlmb)* is expected to contain the perturbed temperature profiles. The *k-th* column of *rtmodpert* contains the *k-th* perturbed temperature profile.
A logical vector called *lparmod* of *ilev* elements is set up. *lparmod* identifies among the altitudes *rzmod*, the altitudes where the temperature profile is fitted. The elements of *lparmod* are first initialised to .FALSE.; then, only the elements that correspond to the tangent altitudes of the simulated spectra for which *lfitgeo* is TRUE are set to TRUE.
For each TRUE element of *lparmod* a perturbed temperature profile is produced; in total, *ipar* perturbed profiles are produced. Let's call *j* the index which numbers the TRUE elements of *lparmod* and *I(j)* the integer function which connects the index of *lparmod* to the index of the TRUE

elements of *lparmod;* in practice if the *j-th* TRUE element of *lparmod* (the numbering always starts from the top) has index *k*, it results *I(j) = k.*
For a fixed *j=2, ..., ipar-1* we describe here below how the profile *rtmodpert(1...ilev,j)* is generated:

- First the vector *rtmod(1...ilev)* is copied into the vector *rtmodpert(1....ilev,j)*
- the element *I(j)* of *rtmodpert* is perturbed:
  rtmodpert(I(j),j) = rtmod(I(j)) + rdt
- the elements of *rtmodpert(l,j)* with *I(j-1) < l < I(j)* are obtained using linear interpolation in altitude, between *rtmod(I(j-1))* and *rtmodpert(I(j),j),*
- the elements of *rtmodpert(l,j)* with *I(j) < l < I(j+1)* are obtained using linear interpolation in altitude, between *rtmodpert(I(j),j)* and *rtmod(I(j+1)).*

In the last two bullets the profiles *rtmodpert* are considered as referred to the altitude grid *rzmod.*

The first and the last perturbed temperature profiles are generated in a different way.
Generation of the first perturbed temperature profile:
we set: *rtmodpert(I(1),1) = rtmod(I(1)) + rdt*
then for *k=1, ...,I(1)-1,* i.e. in the region above *rzmod(I(1))* the perturbed temperature profile is given by:

*rtmodpert(k,1) = rtmod(k) * [rtmodpert(I(1),1)/rtmod(I(1))]*

the elements of *rtmodpert(k,1)* with *I(1) < k < I(2)* are obtained using linear interpolation in altitude, between *rtmod(I(2))* and *rtmodpert(I(1),1).*

Generation of the last perturbed temperature profile:
we set: *rtmodpert(I(ipar),ipar) = rtmod(I(ipar)) + rdt*
then for *k= I(ipar)+1, ...,ipar,* i.e. in the region below *rzmod(I(ipar))* the perturbed temperature profile is given by:

*rtmodpert(k,ipar) = rtmod(k) * [rtmodpert(I(ipar),ipar)/rtmod(I(ipar))].*

the elements of *rtmodpert(k,ipar)* with *I(ipar-1) < k < I(ipar)* are obtained using linear interpolation in altitude, between *rtmod(I(ipar-1))* and *rtmodpert(I(ipar),ipar).*

Step 6: calculation of the perturbed altitude grids.
The lowest altitude of the profiles is considered as unperturbed, the other altitudes are re-computed using hydrostatic equilibrium law. In practice the following operations are performed:
If we indicate with *j* the perturbation index, then for *j = 1,...,ipar*

```
Begin of a loop on the perturbations j:
    rzmodpert(ilev,j) = rzmod(ilev)
        begin of a loop on the levels (jlev = 2, ..., ilev):
            jj=ilev-jlev+1
            rconst=-gravity((rzmod(jj)+rzmod(jj+1))/2.,rlat)*rmovr
            rzmodpert(jj,j)=rzmodpert(jj+1,j)+
                ((rtmodpert(jj,j)+rtmodpert(jj+1,j))/(2.*rconst)*
                 log(rpmod(jj)/rpmod(jj+1)))
        end loop on levels
end loop on perturbations.
```

Where *rmovr* is a parameter defined in the file “parameters_pt.inc’.
From now on the perturbed temperature profiles are referred to the altitude grids *rzmodpert.*

Step 7: calculation of *iderlay*.

For each perturbed profile '*j*' we calculate first *iderlev* which is defined as:
iderlev(j,1) is the first (highest) level where the temperature profile *rtmodpert(1...ilev,j)* is changed with respect to the reference profile *rtmod* (due to the *j-th* perturbation).
iderlev(j,3) is the last (lowest) level where the temperature profile *rtmodpert(1...ilev,j)* is changed with respect to the reference profile *rtmod* (due to the *j-th* perturbation).
iderlev(j,2) is the central level where the temperature in changed because of the *j-th* perturbation; i.e. *iderlev(j,2) = itglev(I(j)).*
Afterwards *iderlay* is computed as described herewith:

Begin loop on perturbations *j=1,ipar*
iderlay(j,2)=iderlev(j,2)-1
End loop on perturbations

iderlay(1,1) = 1
iderlay(1,3) = iderlay(2,2)

Begin loop on perturbations *j=2,ipar-1*
iderlay(j,1) = iderlay(j-1,2)+1
iderlay(j,3) = iderlay(j+1,2)
end loop on perturbations

iderlay(ipar,1) = iderlay(ipar-1,2) + 1
iderlay(ipar,3) = ilev - 1

2.2.11.2 CHECK_PT

CHECK_PT]
|?----PTFROMZ_PT*

Description: This module is used by **MKPLEV_PT** in order to check whether two neighbouring levels can be accepted. This is done by evaluating the temperature and the Voigt line-width variation for a selected reference line, going from one level to the other.

Variables exchanged with external modules:

Name:	Description:
rz1	rz1 = altitude (km) of the first considered level
rz2	rz2 = altitude (km) of the second considered level
rtprof	rtprof(imxpro) = actual temperature profile
rpprof	rpprof(imxpro) = actual pressure profile
rzprof	rzprof(imxpro) = altitudes to which rtprof and rpprof are referred.
ipro	number of elements in the profiles rpprof, rtprof, rzprof
rwmolref	rwmolref = molecular weigth of the gas that has been selected as a reference for building the levels.
dsigm0	dsigm0 = Centre frequency of the line selected as a reference for building the levels.
rhw0ref	rhw0ref = half-width of the line selected as a reference for building the levels.
rmaxtv1	rmaxtv1 = max. allowed temperature variation (K) between two neighbouring levels, when the lower level is located below rzt12.
rmaxtv2	rmaxtv2 = max. allowed temperature variation (K) between two neighbouring levels, when the lower level is located above rzt12.
rzt12	rzt12 = altitude (km) where the temperature thresholds rmaxtv1 and rmaxtv2 are exchanged.

rhwvar	rhwvar = max. allowed half-width variation of the selected reference line between two neighbouring levels.
lcheck	lcheck = logical variable output of the module. If lcheck is returned TRUE the checks have been successful
rexphref	rexphref = exponent for the calculation of Lorentz h-w for the line selected as a reference for building the levels.
rlat	rlat = actual latitude (degrees)
lfirstcall	lfirstcall = logical variable that indicates whether this is the first time that CHECK module is called in the current run of MKPLEV_PT.

Detailed description

The checks proceed along the following steps:

- The internal variable *rmaxtv* is set equal to *rmaxtv1* if *rz1* < *rtz12* otherwise *rmaxtv* = *rmaxtv2*.
- The variables *lcheck, lcheck1* and *lcheck2* are initialised to TRUE.
- Temperature and pressure corresponding to the altitudes *rz1* and *rz2* are evaluated using **PTFROMZ** module. The temperatures of the two levels *rz1* and *rz2* are stored respectively in the variables *rtemp1* and *rtemp2*, while the pressures are stored respectively in the variables *rpres1* and *rpres2*.
- The temperature variation between the two levels is checked:
  if $|rtemp1 - rtemp2| < rmaxtv$ then set *lcheck1* = FALSE
- Doppler and Lorentz half-widths *rhwd, rhwl* of the seleced reference line are then evaluated at the two levels:
  *rhwd1=dsigm0*3.581047d-7**sqrt*(rtemp1/rwmolref)*
  *rhwd2=dsigm0*3.581047d-7**sqrt*(rtemp2/rwmolref)*
  rhwl1=rhw0ref(rpres1/rp0h)*(rt0h/rtemp1)**rexphref*
  rhwl2=rhw0ref(rpres2/rp0h)*(rt0h/rtemp2)**rexphref*
- The Voigt half-widths *rhwv* are then given by:
  *rhwv1 = 0.5 * rhwl1 * 1.0686215708754 +* sqrt*(rhwl1*rhwl1*0.216866444 + rhwd1*rhwd1)*
  *rhwv2 = 0.5 * rhwl2 * 1.0686215708754 +* sqrt*(rhwl2*rhwl2*0.216866444 + rhwd2*rhwd2)*
- The ratio *rhwrat* between the two half widths is then:
  rhwrat = abs(rhwv2 / rhwv1)
- The check on the half-widths is then performed:
  if *rhwrat* < 1 then set *rhwrat*=1./*rhwrat*
  if *rhwrat* > *rhwvar* then *lcheck1* = FALSE
- The result of the checks is then stored in *lcheck*:
  lcheck = *lcheck1* .and. *lcheck2*.

2.2.11.3 CONLAY_PT

Description:
Calculation of the continuum layer values and their partial derivatives with respect to the parameter levels.

Variables exchanged with external modules:

Name:	Description:
rpbase	pressure on the base-levels
rcbase	continuum on the base-levels for each Mw
ibase	number of base-levels
rpmod	pressure on the model-levels
ilev	number of model-levels
iderlay	highest (x,1) and lowest (x,3) and middle(x,2) model layer which is affected by each derivative (imxlmb refers to the parameter-levels)
lparbase	logical occupation vector in the base-grid which determines which base-level is also a parameter-level
nselmw	number of microwindows of the retrieval
rclay	model-layer values of the continuum
rpartcder	partial derivatives of the continuum layer values with respect to the parameter-level values

Module structure:

loop 1 over all microwindows
1.Calculation of *rclay*
2.Calculation of *rpartcder*
end loop 1

Detailed description:

loop 1 over all microwindows:
$jmw=1 \rightarrow nselmw$

1.Calculation of *rclay:*
In a loop over all layers the continuum is interpolated to the middle pressure of each layer:
For $1 \leq klay \leq ilev\text{-}1$:

Determine the mid pressure of the layer: $r1 = \frac{rpmod(klay) + rpmod(klay+1)}{2}$

Determine the base levels in between *r1* lies. These levels are *mbas* and *mbas*+1. Then the interpolation is done:

$$rclay(klay, jmw) = rcbase(mbas, jmw) + \frac{r1 - rpbase(mbas)}{rpbase(mbas+1) - rpbase(mbas)} \cdot \left(rcbase(mbas+1, jmw) - rcbase(mbas, jmw)\right)$$

2.Calculation of *rpartcder:*

This will be described after the implementation of the continuum retrieval.

2.2.11.4 POINT_PT

POINT_PT]
|((((+BLIND_PT *

Description
Determination of the matrix of the IAPT (Implemented Atmospheric Pressure and Temperature) numbers.
(For the concept of 'IAPTs see also: Technical note on: High level algorithm definition and physical and mathematical optimisations, TN-IROE-RSA9601)

Variables exchanged with external modules:

Name:	Description:
igeo	number of simulated geometries
itglev	number of the tangent-level for each geometry
iept	number of additional IAPTs for each geometry above the lowest geometry
ipoint	matrix, which attaches to each pair of layer/geometry the IAPT number
ipath	number of different IAPT numbers in ipoint

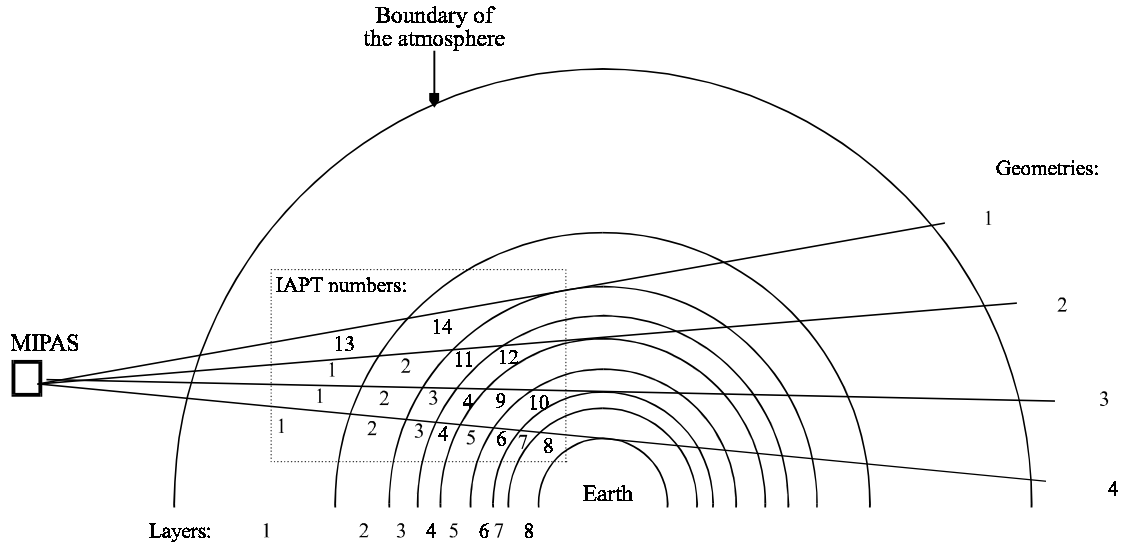
Module structure

1. Calculation of *ipoint* and *ipath*

Detailed description

1. Calculation of *ipoint* and *ipath*
In the following we will shortly explain the concept of IAPT numbers.
This numbering results from the approximation that we will not calculate the equivalent temperatures, pressures and the related cross-sections for each possible pair of layer/geometry, but only for selected pairs. These selected pairs are indicated by their different IAPT numbers and the

set up of this selection is controlled by the input parameter *iept*, which is the number of new IAPT numbers in each geometry above the lowest one. (Therefore, the matrix *ipoint* is a transformation from the two dimensions layer/geometry to one dimension, the IAPT number, and allows to reduce the dimensions of the vectors *rpeq*, *rteq* and *rcross*.)
The following picture shows an example of 4 geometries and 8 layers together with the IAPT numbers belonging to each layer/geometry. In this case *iept* is 2, i.e. 2 new IAPT numbers for each geometry 1,2 and 3.



The related matrix *ipoint* is:

$$\begin{bmatrix} 1 & 1 & 1 & 13 \\ 2 & 2 & 2 & 14 \\ 3 & 3 & 11 & 0 \\ 4 & 4 & 12 & 0 \\ 5 & 9 & 0 & 0 \\ 6 & 10 & 0 & 0 \\ 7 & 0 & 0 & 0 \\ 8 & 0 & 0 & 0 \end{bmatrix}$$

and *ipath* is 14.

For different values of *iept* this matrix is:

$$ipoint^{iept=0} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 0 \\ 4 & 4 & 4 & 0 \\ 5 & 5 & 0 & 0 \\ 6 & 6 & 0 & 0 \\ 7 & 0 & 0 & 0 \\ 8 & 0 & 0 & 0 \end{bmatrix} \quad ipoint^{iept=1} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 11 \\ 3 & 3 & 3 & 0 \\ 4 & 4 & 10 & 0 \\ 5 & 5 & 0 & 0 \\ 6 & 9 & 0 & 0 \\ 7 & 0 & 0 & 0 \\ 8 & 0 & 0 & 0 \end{bmatrix} \quad ipoint^{iept=3} = \begin{bmatrix} 1 & 1 & 1 & 15 \\ 2 & 2 & 12 & 16 \\ 3 & 3 & 13 & 0 \\ 4 & 9 & 14 & 0 \\ 5 & 10 & 0 & 0 \\ 6 & 11 & 0 & 0 \\ 7 & 0 & 0 & 0 \\ 8 & 0 & 0 & 0 \end{bmatrix}$$

For negative values of *iept*, all pairs of layer/geometry are distinguished:

$$ipoint^{iept<0} = \begin{bmatrix} 1 & 9 & 15 & 19 \\ 2 & 10 & 16 & 20 \\ 3 & 11 & 17 & 0 \\ 4 & 12 & 18 & 0 \\ 5 & 13 & 0 & 0 \\ 6 & 14 & 0 & 0 \\ 7 & 0 & 0 & 0 \\ 8 & 0 & 0 & 0 \end{bmatrix}$$

The build-up of this matrix in the module is realised by first setting for each geometry *jgeo* all matrix elements to their layer number (up to the tangent layer *itglev(jgeo)-1*) and all others to 0. In a following loop over jgeo, beginning from the 2nd lowest geometry *igeo-1* all layers between *itglev(jgeo)-iept* and *itglev(jgeo)-1* get a new number, their IAPT-number, which is counted up. The highest IAPT-number is automatically also the output *ipath*.

2.2.11.5 CURGOD_PT

```
CURGOD_PT
   |(----BLIND_PT *
   |(----DREFIND_PT +
   |((---SQRT *
   |(((--QSIMP5_PT >
   |(((--DREFIND_PT +
   |((((-SQRT *
   |((((+QSIMP5_PT >
```

Description: This subroutine performs the ray-tracing for the different geometries and calculates,

1. for all the pairs geometry-layer:
- the column of all the gases (*rcol*) that have to be taken in account in the actual retrieval
- the perturbed column (*rcolpert*) of the main gas of the actual retrieval for the perturbed temperature profiles
- the air column (*raircol*)
- the length of the optical path (*ropath*) in the layer

2. and, only for a sub-set of the possible 'paths', the IAPT-numbers (see subroutine point):
- the equivalent pressure (in Curtis-Godson meaning) (*rpeq*) for all the gases (IAP)
- the perturbed equivalent pressure (*rpeqpert*) of the main gas for the perturbed temperature profiles
- the equivalent temperature (*rteq*) for all the gases (IAT)
- the perturbed equivalent temperature (*rteqpert*) of the main gas for the perturbed temperature profiles.

For some explanations of the reasons of the choices implemented in this module, refer to *T.N. on 'High Level algorithm definition and physical and mathematical optimisations' (TN-IROE-RSA9601)*, sect. 6.1 and 6.2.

Variables exchanged with external modules:

Name:	Description:
ipath	Total number of different IAPTs number
igeo	Total number of simulated geometries
ipar	Total number of altitudes where the temperature profil is fitted
ilev	Total number of atmospheric levels
itglev	Vector that associates to each geometry, the corresponding number of the tangent level.
iderlay	iderlay(imxlmb,3): highest (x,1), lowest (x,3) and middle (x,2) (the one directly above the 'derivated' layer) layer which is affected by each derivative (imxlmb refers to the parameter levels)
ipoint	Matrix of IAPT-number
igas	Total number of gases in the selected MW
rpmod	rpmod(imxlev) = pressure on levels used for the radiat. transf. calc.
rtmod	rtmod(imxlev) = temperature on levels used for the radiat. transf. calc.
rtmodpert	rtmodpert(imxlev,imxlmb) = perturbed temperature profiles, used for the calculation of T derivatives
rzmod	rzmod(imxlev) = heights of levels used for the radiat. tranf. calc.
rzmodpert	rzmodpert(imxlev,imxlmb) = perturbed altitude grids after the perturbation of temp. profiles.
rmrmod	rmrmod(imxlev,imxgas) = volume mixing ratio for each gas considered in actual retrieval on levels used for rad. transf. calc.
rlat	latitude of the actual limb-scan (deg.)
rearad	earth radius
deps	degree of accuracy required for the calculation of Curtis-Godson integrals
rpeq	rpeq(imxpat,imxgas) = implemented atmospheric (equivalent) pressures (IAPs)
rpeqpert	rpeqpert(imxpat,2) = equivalent pressures of the main gas for the perturbed temperature profiles
rteq	rteq(imxpat,imxgas) = implemented atmospheric (equivalent) temperatures (IATs)
rteqpert	rteqpert(imxpat,2) = equivalent temperatures of the main gas for the perturbed temperature profiles
rcol	rcol(imxlay,imxgeo,imxgas) = columns for each layer, each geometry and each gas
rcolpert	rcolpert(imxlay,imxgeo,2) = columns of the main gas for the perturbed temperature profiles
raircol	raircol(imxlay,imxgeo) = air-column for each layer and each geometry
ropath	ropath(imxlay,imxgeo) = optical path lenght for each layer, each geometry
rtmain	rtmain(imxpat) = Curtis-Godson equivalent temperature (IAT) of the main gas

Module structure

1. Initialisation of some variables.

Begin loop 1 over all the simulated geometries

2. Calculation of the number of layers corresponding to the considered geometry, tangent altitude and Snell's law constant.

Begin loop 2 over layers of the actual geometry

3. Check if the actual combination geometry-layer ('path') corresponds to a new IAPT number.

4. Preparation of the inputs for subsequent calculation of the integrals

Begin loop 3 over all the gases in selected microwindows

5. Definition of VMR of the actual gas on the boundaries of the layer

6. Calculation of the following integrals:

- gas column for all the 'paths' (combination of layer and geometry);

- air column and path length for all the paths (this calculation is independent on the gas, so it has to be done only once in the do-loop on gases);
- IAPTs.

7. Calculation and storage of required quantities and definition of the main equivalent temperatures

end loop 3

Begin loop 4 over retrieval parameters

Begin condition 1: if the actual path needs a calculation of the equivalent values for the temperature profile perturbed at the considered parameter:

8. Preparation of the inputs for subsequent integration
9. Calculation of the following integrals (only for the main gas):
   - perturbed column for each path
   - perturbed IAPTs.
10. Calculation and storage of required quantities.

end condition1

End loop 4

End loop 2

End loop 1

Detailed description:

1. Initialisation of some variables

The variable that counts the IAPT *iponew* is set equal to 0 (see 3.).
Each element of the matrix of the perturbed equivalent temperature is set to 100 (this allows to avoid some 'if-conditions', without finding singularities, in the module that calculates the cross-sections).

loop 1 over all the simulated geometries

kgeo=igeo→1

We start from the lowest geometry (igeo) in order to read matrix ipoint in the appropriate way.

2. Calculation of some variables.

_Determination of the number of layers for the actual geometry (*ilayge=itglev(kgeo)-1*).
_Determination of the tangent altitude referred to the surface of the earth r_T (*dtanal=rzmod(ilayge+1)*) and to the centre of the earth R_T (*dtan_0=rearad+dtanal*).
(In the following R will indicate a particular altitude referred to the centre of the earth, r the same altitude referred to the surface of the earth.)
_Calculation of constant in Snell's law:

$$dsnellc = R_T \cdot n(r_T).$$

The refractive index at altitude *r* is a function of pressure and temperature at that altitude.
It is calculated by the function **drefind(T,p)**.

loop 2 over all the layers to be considered for each geometry

lay=1→*itglev(kgeo)-1*

3. Check if the actual combination geometry-layer ('path') corresponds to a new IAPT-number
For the actual combination geometry-layer, a check is performed in order to establish whether equivalent pressure and temperature have to be calculated or not.
Only if the IAPT-number *ipoint (lay,kgeo)* is greater than *iponew*, the pointer *iponew* is updated to the value of *ipoint (lay,kgeo)* and the logical variable *lflag* passes from false to true.

4. Preparation of the inputs for subsequent integration
This section prepares the inputs to module **qsimp5** that will compute the integrals.
The logical variable *lnopert* is set equal to true: this means that we are not calculating perturbated quantities, so the air column and the path length will be calculated.
dalay=rzmod(lay+1), dta=rtmod(lay+1), dpa=rpmod(lay+1) and *dblay=rzmod(lay), dtb=rtmod(lay), dpb=rpmod(lay)* are respectively the heights (referred to the surface of the earth), the temperatures and the pressures on the lower and higher boundary of the layer.
The integration variable *x* used for subsequent integrals is: $x = \sqrt{R^2 - R_T^2}$.
So, the heights referred to the centre of the earth of the boundaries of the layer (*da_0* and *db_0*) are used for calculating the limits of integration for the actual layer:
$dxa = \sqrt{da\_0^2 - dtan\_0^2}$ and $dxb = \sqrt{db\_0^2 - dtan\_0^2}$.

loop 3 over the all the gases in the selected microwindows
jgas=1→*igas*

5. Definition of VMR of the gas on the boundaries of the layer
dmra=rmrmod(lay+1,jgas) and *dmrb=rmrmod(lay,jgas)* are the VMRs of the actual gas on the boundaries of the layer.
6. Calculation of the equivalent values by means of integration along the line of sight.
The module qsimp5 (*dalay, dxa, dblay, dxb, dta, dtb, dpa, dpb, dmra, dmrb, dsnellcpert, dtan_0pert, rearad, rat, deps, dcoll, daircoll, dopathl, dtl, dpl, 1, lflag, lpert*)
performs the calculation of all the required integrals.

7. Calculation and storage of required quantities and definition of main equivalent temperatures
The column (in number of molecules per square centemeter) of the actual path is finally calculated and stored:

$$rcol(lay, kgeo, jgas) = dcoll \cdot rk \cdot 10^{-6},$$

where *rk* is a parameter contained in the file 'parameters.inc' and the factor 10^{-6} is due to the fact that the VMRs are read from input in parts per million (ppm).
If logical variable lflag is true, the equivalent pressure (in mbar) and temperature (°K) are normalised and stored:

$$rpeq(iponew, jgas) = \frac{dpl}{dcoll},$$

$$rteq(iponew, jgas) = \frac{dtl}{dcoll}$$

Since the air column and the path length are indipendent on the gas, inside the loop on the gases they have to be calculated only the first time (*jgas=1*).
Besides, since CO_2 has local code equal 1 in p-T retrieval, if *jgas=1*, the air column, the path lenght and the temperature of the main gas are calculated:

$$raircol = daircoll \cdot rk\ ,$$
$$ropath(lay, kgeo) = dopathl\ ,$$
$$rtmain(iponew) = rteq(iponew, jgas)$$

loop 4 over the retrieved tangent altitudes
jpar=1→ipar

Condition 1
When the value of one tangent temperature is perturbed, it is assumed that not all the columns and IAPTs are perturbed, but only the IAPTs and the columns corresponding to layers between the two tangent levels contiguous to the perturbed one.

The variable $iderlay\left(jpar, \begin{Bmatrix} 1 \\ 2 \\ 3 \end{Bmatrix} \right)$ represents respectively the $\begin{Bmatrix} hightest\ layer \\ middle\ layer\ (the\ one\ directly\ above\ the\ pert.level) \\ lowest\ layer \end{Bmatrix}$ affected by perturbed level *jpar*.

For this reason, two different perturbed values of IAPTs and columns have to be calculated respectively for each IAPT-number and each path, one is obtained when the tangent temperature is perturbed on a tangent level above the layer (*rpeqpert(jpath,2), rteqpert(jpath,2), rcolpert(jpath,2)*), the other one when the tangent temperature is perturbed on the tangent level below it (*rpeqpert(jpath,1), rteqpert(jpath,1), rcolpert(jpath,1)*).
The only exceptions occur for the hightest and the lowest parameter.
Therefore, inside the loop on the parameters (*jpar=1→ipar*), if the condition:

$$iderlay(jpar,1) \le lay \le iderlay(jpar,3)\ ,$$

a procedure analoguous to that performed from 2 to 7 is repeated, but some differences have to be considered.

8. Preparation of the inputs for the integration
The logical variable *lnopert* is set to false, in order not to calculate the considered air column and the path in this case.
The following quantities are calculated:

- perturbed tangent altitude, referred to the surface (*dtanalpert=rzmodpert(itglev(kgeo),jpar)*), and to the centre of the earth (*dtan_0pert=dtanalpert+rearad*)
- Snell's law constant for perturbed tangent altitude ($dsnellcpert = drefind(rtmodpert(ilayge+1), rpmod(ilayge+1)) \cdot dtan\_0pert$)
- the limits of integration *dxa* and *dxb*
- the values of pressure *dpa* and *dpb*, perturbed temperature *dta* and *dtb* and VMR of the main gas *dmra* and *dmrb* on the boundaries of the layer.

The loop on the gases is not performed in this case, because it is assumed that the main contribution to the derivatives arises from the main gas.

9. Calculation of perturbed equivalent quantities
The module

qsimp5(*dalay, dxa, dblay, dxb, dta, dtb, dpa, dpb, dmra, dmrb, dsnellcpert, dtan_0pert, rearad, rlat, deps, dcoll, daircoll, dopathl, dtl, dpl, 1, lflag, lpert***),** together with its sub-module **trapz5,** performs the calculation of all the required integrals.

10. Calculation and storage of required quantities

If the layer is above the perturbed level ($lay \le iderlay(jpar,2)$)

the perturbed column (in number of molecules per square centimeters) of the actual path is calculated and stored:

$$rcolpert(lay,kgeo,1) = dcollpert \cdot rk \cdot 10^{-6}$$

If logical variable *lflag* is true

equivalent pressure (in mbar) and temperature (in °K) are stored in the first row:

$$rpeqpert(iponew,1) = \frac{dpl}{dcoll} \ ,$$

$$rteqpert(iponew,1) = \frac{dtl}{dcoll}$$

end if

else (the layer is below to the perturbed level)

$$rcolpert(lay,kgeo,2) = dcollpert \cdot rk \cdot 10^{-6}$$

If logical variable *lflag* is true

$$rpeqpert(iponew,2) = \frac{dpl}{dcoll} \ ,$$

$$rteqpert(iponew,2) = \frac{dtl}{dcoll}$$

end if

end if

The factor 10^{-6} has to be used because the VMRs are given in ppm, *rk* is a parameter.

2.2.11.6 *QSIMP5_PT & TRAPZ5_PT*

```
QSIMP5_PT
    |-----DREFIND_PT +
    |-----DFUNC1_PT >
    |-----TRAPZ5_PT
    |       |(----SQRT *
    |       |(----PTNMRFROMZ_PT
    |       |       |-----LOG *
    |       |       |-----LOG *
    |       |       |-----EXP *
    |       |       |-----DREFIND_PT +
    |       |(----DFUNC1_PT >
```

Description:

Starting from:

- the limits of integration *dxa* and *dxb*,

- the value of temperature, pressure (and consequently of refractive index) and VMR of the actual gas on the boundaries of the layer,
- the interpolation law in altitude of all these quantities inside the layer,

these two modules can calculate five different numerical integrals: *dcoll*, *dpl*, *dtl*, *daircoll* and *dopathl*. According to the value of the logical variables *lflag* and *lnopert* some of them are not calculated.

Variables exchanged with external modules:

Name:	Description:
dalay	altitude of the lower boundary of the layer
dxa	lower limit of integration
dblay	altitude of the higher boundary of the layer
dxb	higher limit of integration
dta	temperature corresponding to the lower boundary of the layer
dtb	temperature corresponding to the higher boundary of the layer
dpa	pressure corresponding to the lower boundary of the layer
dpb	pressure corresponding to the higher boundary of the layer
dmra	VMR corresponding to the lower boundary of the layer
dmrb	VMR corresponding to the higher boundary of the layer
dsnellc	Snell's law constant
dtan_0	tangent altitude referred to centre of the earth
rearad	earth radius
rlat	latitude
deps	required accuracy for the integrals calculation
dcoll	returned column of this path (to be moved to the choisen measurement units)
daircoll	returned air density (to be multiplied by parameter rk)
dopathl	returned path lenght (in km)
dtl	returned equivalent temperature (to be normalised)
dpl	returned equivalent pressure (to be normalised)
jgas	actual gas number (local code)
lflag	logical variable: only when it is true, the equivalent pressure and temperature have to be calculated
lnopert	logical variable: it is true when the not-perturbed profils are considered. Only when it is true the air column and the path lenght have to be calculated.

Module structure:
See 'Numerical Recipes in FORTRAN', pag. 130-133.

Detailed description:
The structure of this module is exactly the same of the one reported on 'Numerical Recipes in FORTRAN', pag. 130-133, with the exception that more than one integral (a maximum of five integrals) are calculated at the same time.
In particular, the integrals are computed in numerical way using Simpson rule: in the implemented method, the trapezoidal rule is refined until a specified degree of accuracy *deps* has been achieved.

The integrals calculated by **qsimp5** and **trapz5** modules are the following:

1. $$dcoll = \int_{dxa}^{dxb} X_{gas}\left(r(x)\right) \cdot \frac{p\left(r(x)\right)}{T\left(r(x)\right)} \cdot \frac{ds}{dx} \cdot dx$$

2. $$dpl = \int_{dxa}^{dxb} X_{gas}(r(x)) \cdot \frac{p^2(r(x))}{T(r(x))} \cdot \frac{ds}{dx} \cdot dx$$

3. $$dtl = \int_{dxa}^{dxb} X_{gas}(r(x)) \cdot p(r(x)) \cdot \frac{ds}{dx} \cdot dx$$

4. $$daircoll = \int_{dxa}^{dxb} \frac{p(r(x))}{T(r(x))} \cdot \frac{ds}{dx} \cdot dx$$

5. $$dopathl = \int_{dxa}^{dxb} \frac{ds}{dx} \cdot dx;$$

In these integrals the integration variable *x*, is given by:

$$x = \sqrt{R^2 - R_T^2},$$

$\frac{ds}{dx} \cdot dx$ represents the increment along the line of sight *s*, refractive index dependent, $X_{gas}(r(x))$, $p(r(x))$, $T(r(x))$ represent respectively the gas VMR, pressure and temperature behaviour as a function the integration variable *x*.

The values of pressure, temperature, VMR, refractive index at a particular height *r* corresponding to the particular *x* is computed by the module
ptnmrfromz (*dz1, dalay, dta, dpa, dmra, dblay, dtb, dpb, dmrb,* <u>*dt1*</u>*,* <u>*dp1*</u>*,* <u>*dmr1*</u>*,* <u>*drefind1*</u>**)**.

The function

$$\textbf{dfunc1(}dx, drefr, \begin{Bmatrix} dsnellc \\ dsnellcpert \end{Bmatrix}, \begin{Bmatrix} dtan\_0 \\ dtan\_0pert \end{Bmatrix}, dalay, dta, dpa, dblay, dtb, rlat\textbf{)}$$

calculates the value of $\frac{ds}{dx}$ at the altitude r.

The calculation of some of the integrals is bound to the value of the logical variables *lflag* and *lnopert*.
In particular, integral no.1 is always calculated, integrals no. 2 and 3 are calculated only if logical variable *lflag* is true, integrals no. 4 and 5 are calculated only if *lnopert* is true and *jgas* is equal to 1, so the relative outputs of this module have meaning only when these conditions are verified.
Actually, in order to reduce the number of 'if-conditions', the calculation are performed for all the integrals, while the accuracy criteria are checked only for the required integrals.

2.2.11.7 DFUNC1_PT

DFUNC1_PT
|?----DLIM_PT]
| |-----GRAVITY *

Description:
This function calculates the derivative of the line of sight *s* with respect to the integration variable *x*, $\frac{ds}{dx}$.
Since this function has a resolvable singularity in correspondence of the tangent height, the expression of the limit is used in this case.

Variables exchanged with external modules:

Name:	Description:
dx	actual value of the integration variable *x*
drefr	refractive index at *r(x)*
dsnellc	Snell's law constant of the actual configuration
dtan_0	tangent altitude referred to the centre of the earth
dalay	height of the lower boundary of the layer
dta	temperature corresponding to the lower boundary of the layer
dpa	pressure corresponding to the lower boundary of the layer
dblay	height of the higher boundary of the layer
dtb	temperature corresponding to the higher boundary of the layer
rlat	latitude (it is used for calculating the numerical limit)

Module structure:
Begin condition 1: singular point
1. The result is obtained using the function **dlim** (*dalay, dta, dpa, dblay, dtb, dtan_0, rlat***)**
else
2. The result is obtained using function dfunc1
end condition 1

Detailed description:
Condition 1:
if the input variable *dx* is smaller than 10^{-8} km, then we are at the tangent altitude: since $\frac{ds}{dx}$ has a resolvable singularity at this point, ***dfunc1*** is calculated using the function **dlim(dalay,dta,dpa,dblay,dtb,dtan_0,rlat)**
1.
dfunc1=dlim(dalay,dta,dpa,dblay,dtb,dtan_0,rlat)
2.
dfunc1 is calculated using the following formula:

$$dfunc1 = \frac{1}{\sqrt{1 - \frac{dsnellc^2}{drefr^2 \cdot dx^2} + \frac{dtan\_0^2}{dx^2}}}$$

2.2.11.8 DLIM_PT

Description

This function calculates the limit of the function $\frac{ds}{dx}$ at the tangent altitude.

Variables exchanged with external modules:

Name	Description
dalay	height of the lower boundary of the layer
dta	temperature corresponding to the lower boundary of the layer
dpa	pressure corresponding to the lower boundary of the layer
dblay	height of the higher boundary of the layer
dtb	temperature corresponding to the higher boundary of the layer
dtan_0	tangent altitude referred to the centre of the earth
rlat	latitude (it is used for calculating the numerical limit)

Detailed description:
After the calculation of the altitude of the mean altitude of the layer *drmed* and the gradient of the temperature inside the layer *dgradt*, the returned value is calculated using the following expression:

$$dlim = \frac{1}{\sqrt{1 - \frac{dtan\_0}{1 + rcn \cdot \frac{dpa}{dta}} \cdot \frac{rcn \cdot dpa}{dta^2} \cdot \left(dta \cdot rmvr \cdot gravity(drmed, rlat) \cdot \frac{2}{(dta + dtb)} + dgradt \right)}}$$

gravity*(r*,*rlat*)** is the function that calculates gravity acceleration, known the altitude and the latitude, *rmvr* and *rcn* are parameters contained in the file 'parameters.inc'.

2.2.11.9 DREFIND_PT

Description:
This function calculates refraction index *n* for given temperature and pressure.

Variables exchanged with external modules:

Name:	Description:
dt	temperature
dp	pressure

Detailed description:

The used formula is:

$$n = 1 + refind \cdot \frac{dp}{dt}.$$

refind is a parameter contained in the file 'parameters.inc'

2.2.11.10 PTNMRFROMZ_PT

Description:
Starting from the value of pressure, temperature, refractive index, VMR on the boundaries of a given layer, this module calculates the value of pressure, temperature, refractive index, VMR for a given altitude *dz1* inside the layer.

Variables exchanged with external modules:

Name:	Description:
dz1	altitude, referred to the surface of the earth, where the values of pressure, temperature, refractve index and VMR are required.
dalay	altitude of the lower boundary of the layer
dta	temperature in correspondence of *dalay*
dpa	pressure in correspondence of *dalay*
dmra	VMR in correspondence of *dalay*
dblay	altitude of the higher boundary of the layer
dtb	temperature in correspondence of *dblay*
dpb	pressure in correspondence of *dblay*
dmrb	VMR in correspondence of *dblay*
dt1	returned temperature at *dz1*
dp1	returned pressure at *dz1*
dmr1	returned VMR at *dz1*
drefind1	returned refractive index at *dz1*

Module structure:
1. Calculation of the temperature *dt1* at the altitude *dz1* by using a linear interpolation.
2. Calculation of pressure *dp1* at the altitude *dz1* by using exponential interpolation.
3. Calculation of VMR *dmr1* at the altitude *dz1* by using a linear interpolation.
4. Calculation of refractive index *drefind1* at the altitude *dz1*.

Detailed Description:
1. Inside the layer, the temperature *dt1* is interpolated linearly in altitude, known the values of temperature (*dta* and *dtb*) and the heights of the levels that delimit the boundaries of the layer (*dalay* and *dblay*).
2. The pressure *dp1* at altitude *dz1* is obtained performing an exponential interpolation of the values of pressure on the boundaries of the layer *dpa* and *dpb*.
3. The value of VMR at *dz1* is obtained by performing a linear interpolation of the values of VMR on the boundaries of the layer (*dmra* and *dmrb*).
4. The refractive index *drefind1* at the altitude *dz1* is computed by the function **drefind(dt1,dp1)**.

2.2.11.11 CROSS_PT

```
CROSS_PT
   |-----BLIND_PT *
   |-----LOFICO_PT ]
   |-----FLINT_PT *
   |         |-----FPARTS_PT *
   |-----DECOMPR_PT *
   |((((+HUMLI_PT ]
   |((((+FLINT_PT *
   |((((+FCO2CHI_PT *
   |((((+POLCOE2ND_PT +
   |((((+POLCOE2ND_PT +
```

Description:

- Using the spectroscopic line-data this routine determines the absorption cross-sections for each general wavenumber fine grid point, each IAPT number and each gas which has to be considered in the actual microwindow using the equivalent pressures and temperatures calculated in '**curgod_pt**' for the unperturbed atmospheric profile.
- Additionally it calculates the absorption cross-sections for each general wavenumber fine grid point, for each IAPT number, but only the main gas, using the equivalent temperatures which were calculated in '**curgod_pt**' for the temperature-perturbed atmospheric profiles.
- In the case cross-section look-up tables are available, this module returns the value of cross-sections obtained decompressing the compressed look-up tables. ORM is able to handle also cases in which the look-up tables are available only for a sub-set of the operational microwindows and for a sub-set of the gases contributing to the emission in each microwindow.
- In the case irregular grid is available, this module returns the value of the cross-section on the so-called 'compressed' grid, that is the grid containing only the '1' grid points of the irregular grid.

Variables exchanged with external modules:

Name:	Description:
imw	number of the actual Mw
rpeq	equivalent pressures
rteq	equivalent temperatures
rteqpert	equivalent temperatures of the main gas for the perturbed temperature profiles
rpeqpert	equivalent pressure of the main gas for the perturbed temperature profiles (not used in this version of cross_pt)
itglev	number of the tangent-level for each geometry
isigma	number of wavenumber grid points for each Mw
dsigma	general wavenumber fine grid
delta	general fine grid interval [cm-1]

igeo	number of simulated geometries
iocsim	occupation matrix for the simulations to be performed
igasmw	number of gases to be considered in each Mw
ruplin	upper limit where the line has to be considered [km]
rlolin	lower limit where the line has to be considered [km]
iline	number of lines in each microwindow
icode	HITRAN code for each line of each Mw
rint0	line intensity for each line of each Mw
relow	lower state energy for each line of each Mw
rhw0	foreign broadened half width for each line of each Mw
dsilin	central wavenumber for each line of each Mw
ioutin	flag for each line of each Mw =1: line-shape has to be calculated at each wavenumber inside the Mw =2: line is considered as nearby continuum (calculated at three points inside the Mw
igasnr	global gas number for the local gas number of each Mw
rexph	exponent for T dependence of half width for each line of each Mw
rwmol	molecular weight for each HITRAN molecular code and isotope number
igashi	HITRAN code number for each global gas number
iiso	isotope number for each line of each Mw
ipoint	IAPT-number for each layer and each geometry
ninterpol	switch for the decision of interpolation of the absorption cross-sections for the geometries above the lowest geometry (only if the IAPT number of the path is increasing, which was decided during the calculation of ipoint) =-1: no interpolation, all cross-sections recalculated =0: all cross-sections above the lowest geometry are interpolated =1: new calculation only of the tangent-layer, all other layers interpolated =2: new calculation of the tangent-layer and the layer above, all others interpolated =3: ...
raircol	air-column for each layer and each geometry
rcol	column amounts for each layer, each geometry and each gas
rzmod	heights of levels used for the radiat. tranf. calculation
lmgas	logical array lmgas(imxgmw,imxmw) lmgas(mgas,imw)=.true. : calculation of cross-sections without look-up tables lmgas(mgas,imw)=.false. :calculation of cross-sections by means of look-up tables
ilookupmw	integer*4 ilookupmw(imxmw) ilookupmw(imw)=0 no look-up tables for mw imw ilookupmw(imw)=1 look-up tables for all the absorbers of the mw

	ilookupmw(imw)=2 look-up tables for not all the absorbers of the mw
nll	I*4 : nll(imxgmw,imxmw): number of basis vector
npl	I*4 : npl(imxgmw,imxmw): number of -log(pressure) tabulation points
rp1l	R*4: rp1l(imxgmw,imxmw): lowest -log(pressure) value
rdpl	R*4: rdpl(imxgmw,imxmw): spacing of -log(pressure) tabulation
ntl	I*4: ntl(imxgmw,imxmw): number of temperature tabulation points
rt1l	R*4: rt1l(imxgmw,imxmw): lowest tabulated temperature
rdtl	R*4: rdtl(imxgmw,imxmw): spacing of temperature tabulation
ru	R*4: ru(imxsi2,imxbv,imxgmw,imxmw): U-matrix
rkl	R*4: rkl(imxbv,imxnx,imxgmw,imxmw): K-matrix
tab	character*3: tab(imxgmw,imxmw): tabulation code of cross-section look-up tables
rcross	R*4: rcross(imxsi2,imxpat,imxgmw): absorption cross sections for each general wavenumber fine grid point (1st index), each IAPT number (2nd index) and each gas (3rd index) for the actual Mw
rcrosspert	R*4: rcrosspert(imxsi2,imxpat,2): absorption cross-sections for the main gas: for each general wavenumber fine grid point (1st index), for each IAPT number (2nd index), and for the two equivalent temperatures profiles (3rd index). So, rcrosspert(i,j,1) are the cross sections calculated using the temperatures rteqpert(j,1) and rcrosspert(i,j,2) by using rteqpert(j,2).
lirrgridmw	logical: *lirrgridmw(imxmw):* logical vector that, for each selected microwindow in the actual retrieval, indicates whether the irregular grid is available.
iigrid	integer*4: *iigrid(imxsig,imxgeo,imxmw).* *iigrid (1→ isigma(imw),imw):* irregular grid in the '0' and '1' representation for all the fine grid points of the extended microwindow *imw*.
nused1	integer*4: nused1(imxmw): total number of points of the compressed grid for each microwindow

Module structure:

1. Initialisation of variables

Begin loop 1 over the geometries valid for the actual microwindow

Begin loop 2 over the layers of the actual geometry for which a new cross- section must be determined

Begin condition 1 on the use of look-up tables

2. Calculation of cross-sections by means of available look-up tables

End condition 1

Begin condition 2 : the cross-sections are calculated without look-up tables
Begin condition 3 the cross sections are interpolated
Begin loop 3 over the gases of the actual Mw
3. Interpolation of the cross sections
end loop 3
else condition 3 the cross sections are calculated
4. Definition of local fine and coarse wavenumber grid
Begin loop 4 over all lines of the actual Mw that must be considered for the actual altitude
5. Initialisation of variables for line-calculation
Begin condition 4 the lines are calculated at each point
6. Calculation of the line in the local coarse grid
7. Calculation of the line in the local fine grid
8. Calculation of the line for the temperature perturbed cross sections in the coarse and fine grid
else condition 4 the lines are handled as near continuum
9. Calculation of the line at 3 points inside the Mw
end condition 4
end loop 4
10. Interpolation of the T perturbed cross sections to the general wavenumber fine grid
Begin loop 5 over the gases of the actual Mw
11. Interpolation of the cross sections from the local coarse and fine grid to the general fine grid
12. Interpolation of the nearby continuum to the general fine grid
end loop 5
end condition 3
end condition 2
end loop 2
end loop 1

Detailed description:

loop 1 over the geometries valid for the actual microwindow
kgeo=igeo→1
if [*iocsim(kgeo,imw)*≠0]
Starting from the lowest geometry (*igeo*) this loop (i.e. the commands inside the loop) is only executed if this observation geometry has to be simulated for the actual Mw.

loop 2 over the layers of the actual geometry for which a new cross-section must be determined
llay=1→itglev(kgeo) -1
This loop begins from the outer layer and goes down to the tangent layer (*itglev(kgeo)-1*). It is only executed if new cross-sections must be calculated, i.e. if the IAPT-number *ipoint(llay,kgeo)* is increasing. For the cases that the IAPT number is not increasing, the cross-sections have already been calculated during an earlier execution.

Condition 1 on the use of look-up tables
if ilookupmw(imw) ≠ 0 , at least for some of the absorbers contained in the mw look-up tables are available.

condition 2: the cross-sections are calculated without look-up tables
if *ilookupmw(imw) =0 .OR.. ilookupmw(imw) = 2 ,*
the line-by-line cross-section calculation is performed.

condition 3 the cross sections are interpolated or calculated
The cross sections are interpolated (using the cross sections which have already been calculated for the lowest geometry) if we are not in the lowest geometry and if we are in a layer that has to be interpolated (indicated by *ninterpol*):

if [*kgeo*<*ilowgeo* ∧ *llay* < *itglev(kgeo)-ninterpol* ∧ *ninterpol*≠-1]

Where *ilowgeo* is the lowest geometry that must be calculated for the actual Mw.
If this conditions are not fulfilled the cross sections are calculated explicitly using the line data.

loop 3 over the gases of the actual Mw
mgas=1→igasmw(imw)
The calculation inside this loop are performed only if one of the following conditions are verified:
ilookupmw(imw) = 0 .or. lmgas(mgas,imw) = .true.

loop 4 over all lines of the actual Mw that must be considered for the actual altitude and corresponding to gases of which cross-section was not previously calculated with the use of look-up tables
mline=1,iline(imw)
if [*ruplin(mline,imw)>rzmod(llay)>rlolin(mline,imw)*]
if [*ilookupmw(imw) = 0 .or. lmgas(igasact (icode (mline, imw), imw) = .true.*]

condition 4 the lines are calculated at each point or handled as continuum
if [*ioutin(mline,imw)=1*]: the lines are explicitly calculated at each point of the local coarse and fine grid.
if [*ioutin(mline,imw)=2*]: the lines are handled as near continuum and calculated only at three points inside the Mw.

loop 5 over the gases of the actual Mw
mgas=1→igasmw(imw)
The calculation inside this loop are performed only if one of the following conditions are verified:
ilookupmw(imw) = 0 .or. lmgas(mgas,imw) = .true.

1. Initialisation of variables

- Calculation of vector *igasact(imxhit)* that gives for each hitran gas number the local Mw gas number: *igasact(igashi(igasnr(j,imw))) = j* for *1 ≤ j ≤ igasmw(imw)*
- Determination of the line with the largest intensity of the main gas: line number: *imaxlin*

- The total number of points *nsigma* of the grid to be used for the calculation of cross-section is determined. If an irregular grid is available, the compressed grid is used and *nsigma= nused1(imw),* if the irregular grid is not available, *nsigma= isigma(imw).*

2. Calculation of cross-sections by means of available look-up tables

For each absorbers contained in the considered mw: *mgas = 1 → igasmw(imw),* a control is done in order to see if the look-up table relative to this absorber is available (*lmgas(mgas,imw)=* .false.).
If this is the case, cross-section calculation is performed as follows:
firstly, the preliminary calculations are performed:
the hitran code of the gas:
ihit=igashi(igasnr(mgas,imw));
and the equivalent -log(pressure) and temperature relative to the path ipo:
rp= -alog(rpeq(ipoint(llay,kgeo),igasnr(mgas,imw));
rt= rteq(ipoint(llay,kgeo),igasnr(mgas,imw).

The calculation of cross-section is performed by module **decompr_pt:**
decompr_pt(*rp, rt, mgas, imw, ru, rkl, nll, npl, rp1l, rdpl, ntl, rt1l, rdtl, nsigma, tab, rcross1***)**
The vector rcross1 is then stored in the array *rcross*:
for each *msig,* from 1 to *nsigma:*
rcross(msig,ipo,mgas)=rcross1(msig)

If the gas *mgas* is the main gas of the mw (*mgas = 1*), also the calculation of the perturbed cross-sections is performed by means of look-up tables.

For *m1*=1 and *m1*=2 the following calculations are performed:
if (*rpeqpert(ipoint(llay,kgeo),m1) > 1.e=-8))* then
rppert=- alog(rpeqpert(ipoint(llay,kgeo),m1))
else
do msig=1,nsigma
rcrosspert(msig, ipo,m1)=0.
end do
go to {end on loop over gases}
end if
rtpert=rteqpert(ipoint(llay,kgeo),m1)

The calculation of the cross-section is done by the module **decompr_pt**:
decompr_pt(*rppert, rtpert, mgas, imw, ru, rkl, nll, npl, rp1l, rdpl, ntl, rt1l, rdtl, nsigma, tab, rcross1*)
The vector *rcross1* is then stored in the array *rcrosspert*:
for each *msig,* from 1 to *nsigma:*
rcrosspert(msig,ipo,m1)=rcross1(msig).

3. Interpolation of the cross sections

The cross sections for the geometries above the lowest geometry are calculated (for each general fine grid point) by linear interpolation using the cross sections already calculated for the lowest geometry. This linear interpolation is performed with respect to the equivalent pressures, i.e. it is first decided between which equivalent pressures of the lowest geometry the actual equivalent pressure lies and than the cross sections are interpolated to the actual equivalent pressure. This is

done for the cross sections of all gases (*rcross*) and for the temperature perturbed cross sections (rcrosspert) for the main gas.
E.g. for *rcross* the formula for all wavenumbers on the general fine wavenumber grid (*msig*) is:

$$rcross(msig, ipoint(llay, kgeo), mgas) = r1 + (r2 - r1) \cdot \frac{p - p1}{p2 - p1}$$

$$r1 = rcross(msig, ipoint(llay1, ilowgeo), mgas)$$

$$r2 = rcross(msig, ipoint(llay2, ilowgeo), mgas)$$

with: $p = rpeq(ipoint(llay, kgeo), igasnr(mgas, imw))$

$$p1 = rpeq(ipoint(llay1, ilowgeo), igasnr(mgas, imw))$$

$$p2 = rpeq(ipoint(llay2, ilowgeo), igasnr(mgas, imw))$$

where *llay1* and *llay2* determine the pressures *p1* and *p2* of the lowest geometry between which the actual pressure *p* lies.

4. Definition of local fine and coarse wavenumber grid

The local (for the actual geometry and layer) coarse and fine wavenumber grid is defined by calling the module **lofico**:

$$\text{lofico}\begin{bmatrix} dsilin, ipoint(llay, kgeo), imw, rwmol, icode, iiso, rhw0, imaxli, rpeq, rteq, \\ delta, isigma, dsigma, igasmw, rexph, iqlfgf, dsiglf, dsiglc, isiglf, isiglc, \\ deltalf, deltalc, rcrolf, rcrolc, rcrolfpert, rcrolcpert \end{bmatrix}$$

Note that even if an irregular grid is available, and as a consequence the final cross-sections will be stored on the compressed grid, the local fine and coarse grids are built starting not from the compressed grid, but from the regular fine grid.
Therefore, lofico routine is not affected by the use of the compressed grid.

5. Initialisation of variables for line-calculation

- Calculation of the Doppler half width:

$$rdhalf = dsilin(mline, imw) \cdot dcdop \cdot \sqrt{\frac{rteq(ipo, ign)}{rwmol(icode(mline, imw), iiso(mline, imw))}}$$

with: $ipo = ipoint(llay, kgeo)$

and: $ign = igasnr(igasact(icode(mline, imw)), imw)$, the global gas number for the hitran gas number of the actual line.
dcdop is a parameter.

- Calculation of the Lorentz half width:

$$rlhalf = rhw0(mline,imw)\cdot\frac{rpeq(ipo,ign)}{rp0h}\cdot\left[\frac{rt0h}{rteq(ipo,ign)}\right]^{rexph(mline,imw)}$$

With the parameters *rp0h, rt0h.*

- Calculation of the line intensity
  The line intensity *rlint* is calculated by a call to the module **flint**:

$$rlint = \text{flint}\begin{bmatrix} rint0(mline,imw), relow(mline,imw), rteq(ipo,ign), \\ dsilin(mline,imw), icode(mline,imw), iiso(mline,imw) \end{bmatrix}$$

- Calculation of the line intensity for the temperature perturbed profiles

$$rlint1 \text{ or } rlint2 = \text{flint}\begin{bmatrix} rint0(mline,imw), relow(mline,imw), rteqpert(ipo,1 \text{ or } 2), \\ dsilin(mline,imw), icode(mline,imw), iiso(mline,imw) \end{bmatrix}$$

6. Calculation of the line in the local coarse grid

The cross sections on the local coarse grid *rcrolc (*dimension *(imxsig,imxgmw)*) are calculated from the boundaries of the microwindow up to a distance of $(rdhalf + rlhalf)\cdot rvmult$ wavenumbers from the line centre by using the Lorentz function (*rvmult* is a parameter). In the region around the line centre the cross sections on the fine grid are constant. This constant is determined as the mean value of the last Lorentz calculated cross sections on the left and on the right of the line.
The boundary indices for the Lorentz calculation on the local coarse grid are:

$$i1c = 1$$

$$i2c = \text{nint}\left[\frac{dsilin(mline,imw) - (rdhalf + rlhalf)\cdot rvmult - dsiglc(1)}{deltalc}\right] + 1$$

$$i3c = \text{nint}\left[\frac{dsilin(mline,imw) + (rdhalf + rlhalf)\cdot rvmult - dsiglc(1)}{deltalc}\right] + 1$$

$$i4c = isiglc$$

Where *isiglc, dsiglc, deltalc* have been determined in 3.
(One has to take care that for a line very near to the boundary of the microwindow (where i2c could become less than *i1c* ...) these coeffiecents are set to the boundary values!)

Calculation of Lorentz function for $i1c \le i \le i2c\text{-}1$ and $i3c+1 \le i \le i4c$:

$$rlinfctlc(i) = \frac{1}{\pi}\frac{rlhalf}{rlhalf^2 + (dsiglc(i) - dsilin(mline,imw))^2}$$

Calculation of the cross sections and adding to the cross sections from the previous lines:

$rcrolc(i,ig) = rlint \cdot rlinfctlc(i) + rcrolc(i,ig)$
with: $ig = igasact\left(icode\left(mline,imw\right)\right)$, the local gas number for the actual line.

The value for the 'plateau' region, i.e. in the vicinity of the line centre is:

$$rplatfctn = \frac{rlinfctlc(i2c-1) + rlinfctlc(i3c+1)}{2}$$

So, for $i2c \le i \le i3c$:

$$rcrolc(i,ig) = rlint \cdot rplatfctn + rcrolc(i,ig)$$

7. Calculation of the line in the local fine grid

On the local fine grid the lines are only calculated in the vicinity of the line, where the cross sections on the local coarse grid are constant (see 5.), i.e. for distances less than $\left(rdhalf + rlhalf\right) \cdot rvmult$ wavenumbers from the line centre. In this region the line profile is partly calculated by the Lorentz and partly by the Voigt function. The Voigt function is used inside an intervall of $\pm rdhalf \cdot rdmult$ wavenumbers from the line centre (*rdmult* is a parameter).
The boundary indices on the local fine grid are:

$$i1f = (i2c-2) \cdot iqlclf + 2$$

$$i2f = \text{nint}\left[\frac{dsilin\left(mline,imw\right) - rdhalf \cdot rdmult - dsiglf\left(i1f\right)}{deltalf}\right] + i1f$$

$$i3f = \text{nint}\left[\frac{dsilin\left(mline,imw\right) + rdhalf \cdot rdmult - dsiglf\left(i1f\right)}{deltalf}\right] + i1f$$

$$i4f = i3c \cdot iqlclf$$

With the parameter *iqlclf*, the quotient between the local coarse and fine grid.

Calculation of Lorentz function for $i1f \le i \le i2f\text{-}1$ and $i3f\text{+}1 \le i \le i4f$:

$$rlinfctlf(i) = \frac{1}{\pi}\frac{rlhalf}{rlhalf^2 + \left(dsiglc(i) - dsilin(mline,imw)\right)^2}$$

Calculation of the cross sections and adding to the cross sections from all the previous lines:

$$rcrolf(i,ig) = rlint \cdot rlinfctlf(i) - rplatcro + rcrolf(i,ig)$$

with: $ig = igasact\left(icode\left(mline,imw\right)\right)$, the local gas number for the actual line, and *rplatcro* the coarse grid 'plateau' value which was determined in 5.

For $i2f \le i \le i3f$ the line function is determined by the Voigt lineshape:

$$rlinfctlf(i) = \sqrt{\frac{\ln 2}{\pi}} \frac{rre}{rdhalf}$$

where *rre* is the result from a call to the routine **humli**(*rx,ry,rre*), with:

$$rx = \sqrt{\ln 2} \frac{|dsiglf(i) - dsilin(mline, imw)|}{rdhalf}$$

$$ry = \sqrt{\ln 2} \frac{rlhalf}{rdhalf}$$

The cross sections are calculated from *rlinfctlf* like in the case of the Lorentz calculation (see above).

8. Calculation of the line for the temperature perturbed cross sections in the coarse and fine grid

The calculation of the temperature perturbed cross sections is approximated by calculating the perturbed line intensities (*rlint1*,*rlint2*), and recalculatin the line profile only for the fine grid part, where the Voigt line shape is used:

The temperature perturbed cross sections on the coarse grid are:

For $i1c \le i \le i2c\text{-}1$ and $i3c+1 \le i \le i4c$:
$rcrolcpert(i,1 \text{ or } 2) = rlint1 \text{ (or } rlint2) \cdot rlinfctlc(i) + rcrolcpert(i,1 \text{ or } 2)$
For $i2c \le i \le i3c$:

$$rcrolcpert(i,1 \text{ or } 2) = rlint1 \text{ (or } rlint2) \cdot rplatfct + rcrolcpert(i,1 \text{ or } 2)$$

The temperature perturbed cross sections on the fine grid are:

For $i1f \le i \le i2f\text{-}1$ and $i3f+1 \le i \le i4f$:
$rcrolfpert(i,1 \text{ or } 2) = rlint1 \text{ (or } rlint2) \cdot \left(rlinfctlf(i) - rplatfct\right) + rcrolfpert(i,1 \text{ or } 2)$

For $i2c \le i \le i3c$:
$rcrolfpert(i,1 \text{ or } 2) = rlint1 \text{ (or } rlint2) \cdot \left(r1(\text{or } r2) - rplatfct\right) + rcrolfpert(i,1 \text{ or } 2)$

with $r1 = \sqrt{\frac{\ln 2}{\pi}} \frac{rre\_1}{rdhalf\_1}$ and $r2 = \sqrt{\frac{\ln 2}{\pi}} \frac{rre\_2}{rdhalf\_2}$

where *rre_1* and *rre_2* are the results from a call to the routine **humli**(*rx_1,ry_1,rre_1*) and **humli**(*rx_2,ry_2,rre_2*) with:

$$rx\_1(\text{or } rx\_2) = \sqrt{\ln 2} \frac{|dsiglf(i) - dsilin(mline, imw)|}{rdhalf\_1(\text{or } rdhalf\_2)}$$

$$ry\_1(\text{or } ry\_2) = \sqrt{\ln 2} \frac{rlhalf\_1(\text{or } rlhalf\_2)}{rdhalf\_1(\text{or } rdhalf\_2)}$$

and the perturbed Lorentz and Doppler half widths:

$$rlhalf\_1 = rhw0(mline,imw)\cdot\frac{rpeqpert(ipo,1)}{rp0h}\cdot\left[\frac{rt0h}{rteqpert(ipo,1)}\right]^{rexph(mline,imw)}$$

$$rlhalf\_2 = rhw0(mline,imw)\cdot\frac{rpeqpert(ipo,1)}{rp0h}\cdot\left[\frac{rt0h}{rteqpert(ipo,1)}\right]^{rexph(mline,imw)}$$

$$rdhalf\_1 = dsilin(mline,imw)\cdot dcdop\cdot\sqrt{\frac{rteqpert(ipo,1)}{rwmol(icode(mline,imw),iiso(mline,imw))}}$$

$$rdhalf\_2 = dsilin(mline,imw)\cdot dcdop\cdot\sqrt{\frac{rteqpert(ipo,2)}{rwmol(icode(mline,imw),iiso(mline,imw))}}$$

9. Calculation of the line at 3 points inside the Mw

For lines outside the microwindow which are taken into account as near continuum, the cross sections (and perturbed cross sections for the main gas CO_2) are calculated at the first point, at the middle point and at the last point of the microwindow. Later, in 11, they will be interpolated to the general fine grid.

The procedure is:

- calculating the line profile using the Lorentz line shape (see above) at the three wavenumbers inside the microwindow.
- if the line is a CO_2 line (if [icode(mline,imw)=2]) the profile is multiplied with the CO_2 chi factor which is calculated by a call to module **fco2chi**:

  $$\text{fco2chi}\left[rteq(ipo,ign),dconsi-dsilin(mline,imw),1\right]$$

  ipo and *ign* have been defined in 4. This application of the χ-factor has also to be done for the perturbed near profiles (since they are only calculated for CO_2, the main gas of p-T retrieval). Therefore, in the call of **fco2chi** *rteqpert(ipo*,1 or 2*)* is used instead of *rteq*.

- The absorption cross sections at the 3 points inside the Mw are now calculated like in 7 or 8 by multiplication of the profile with *rlint* (in the case of CO_2 also with *rlint1* and *rlint2*) and added to the near continuum cross sections (and perturbed cross sections) from the previous line calculation.

10. Interpolation of the T perturbed cross sections to the general wavenumber fine grid

In this part the temperature perturbed cross sections *rcrolcpert* and *rcrolfpert* are interpolated linearly in wavenumber to the general fine grid and added in the output vector *rcrosspert(i,ipoint(llay,kgeo),1* or *2)* (*i* is the index on the general fine grid). If an irregular grid is available (*lirrgridmw(imw)*= true) only the perturbed cross-section values corresponding to the points of the 'compressed' grid have to be stored in

$\mathrm{rcrosspert}\left(\mathrm{ksig}=1\rightarrow nsigma, ipo, \frac{1}{2}\right)$.

11. Interpolation of the cross sections from the local coarse and fine grid to the general fine grid

As in 10, but now for each gas of the microwindow, the output cross section vector *rcross(i,ipo,mgas)* of the general fine grid is filled by linear interpolation in wavenumber using the vectors *rcrolf(j,mgas)* and *rcrolc(k,mgas)*, where *j* is the index on the local fine grid and *k* on the local coarse grid.

If an irregular grid is available (*lirrgridmw(imw)*= true) only the cross-section values corresponding to the points of the 'compressed' grid have to be stored in

$\mathrm{rcross}(\mathrm{ksig}=1\rightarrow nsigma, ipo, mgas)$.

12. Interpolation of the nearby continuum to the general fine grid

For each gas of the Mw the nearby continuum values (original and perturbed) which were calculated in 8. for three points inside the microwindow are interpolated (2nd order) to the general wavenumber fine grid and added to the cross section output vectors *rcross* and (if the gas is CO_2) *rcrosspert*.

Again, if an irregular grid is available (*lirrgridmw(imw)*= true) these operations have to be done only for the points of the 'compressed' grid.

The coefficients for the parabolic interpolation are calculated using module **polcoe2nd**.

2.2.11.12 FCO2CHI

Description:

This function calculates the chi-factor for the correction of the CO_2-lineshape. The chi-factor is calculated for the N_2- and the O_2-broadening of CO_2-lines using the parametrizations from:

- C. Cousin, R. Le Doucen, C. Boulet, and A. Henry, 'Temperature dependence of the absorption in the region beyond the 4.3-μm band head of CO_2. 2: N_2 and O_2 broadening', Appl. Opt.,24, 3899-3907, (1985).
- V. Menoux, R. Le Doucen, J. Boissoles, and C. Boulet, 'Line shape in the low frequency wing of self- and N_2 broadened ν_3 CO_2 lines: temperature dependence of the asymmetry', Appl. Opt., 30, 281-286, (1991).

Variables exchanged with external modules:

Name:	Description:
fco2chi	CO_2 χ-factor (the function value)
rt	equivalent temperature
dsidif	distance to the line centre
nswco2	switch for the calculation of the chi-factor in the case of CO_2-lines =0: no chi-factor, =1: due to N_2/O_2 broadening, =3: only due to N_2 -broadening
isohit	HITRAN isotope number of the actual line

Module structure:

1. Calculation of the CO_2 χ-factor

Detailed description:

1. Calculation of the CO_2 χ-factor

The chi-factors are linearly interpolated in the ranges 193 - 238 K and 238 - 296 K and linearly extrapolated to lower (higher) temperatures from these ranges. This subroutine is only valid for calculations up to 130 cm^{-1} from the lines center, since beyond this wavenumber the asymmetry of the chi-factor is only known for 296 K of N_2 and O_2 and for 193 K for O_2. (the only asymmetry included here is for 193 K for N_2 in the range 50-130 cm^{-1}). The chi-factor is then calculated by weighting of the N_2 and O_2 factors according to their atmospheric relative abundance.

```
real*8 function fco2chi(rt,dsidif,nswco2)

      dsi=abs(dsidif)
*
*  if Temperature < 238K
*
      if (rt.lt.238.) then
        if (dsi.le.5.) then
          rchin2=1.
          rchio2=1.
        else if (dsi.le.9.) then
          r1=1.
          r2=1.968*exp(-0.1354*dsi)
          rtq=(rt-193.)/45.
          rchin2=(r2-r1)*rtq + r1
          rchio2=rchin2
        else if (dsi.le.11.) then
          r1=3.908*exp(-0.1514*dsi)
          r2=1.968*exp(-0.1354*dsi)
          r3=1.
          r4=r2
          rtq=(rt-193.)/45.
          rchin2=(r2-r1)*rtq + r1
          rchio2=(r4-r3)*rtq + r3
        else if (dsi.le.22.) then
          r1=3.908*exp(-0.1514*dsi)
          r2=1.968*exp(-0.1354*dsi)
          r3=7.908*exp(-0.1880*dsi)
          r4=r2
          rtq=(rt-193.)/45.
          rchin2=(r2-r1)*rtq + r1
          rchio2=(r4-r3)*rtq + r3
        else if (dsi.le.23.) then
          r1=3.908*exp(-0.1514*dsi)
          r2=0.160*exp(-0.0214*dsi)
          r3=7.908*exp(-0.1880*dsi)
          r4=r2
          rtq=(rt-193.)/45.
          rchin2=(r2-r1)*rtq + r1
          rchio2=(r4-r3)*rtq + r3
        else if (dsi.le.28.) then
          r1=0.207 - 3.778e-3 * dsi
          r2=0.160*exp(-0.0214*dsi)
          r3=0.122 - 7.539e-4 * dsi
          r4=r2
          rtq=(rt-193.)/45.
          rchin2=(r2-r1)*rtq + r1
          rchio2=(r4-r3)*rtq + r3
        else if (dsi.le.35.) then
          r1=0.219*exp(-0.0276*dsi)
```

```
        r2=0.160*exp(-0.0214*dsi)
        r3=0.122 - 7.539e-4 * dsi
        r4=r2
        rtq=(rt-193.)/45.
        rchin2=(r2-r1)*rtq + r1
        rchio2=(r4-r3)*rtq + r3
      else if (dsi.le.50.) then
        r1=0.219*exp(-0.0276*dsi)
        r2=0.160*exp(-0.0214*dsi)
        r3=0.349*exp(-0.0369*dsi)
        r4=r2
        rtq=(rt-193.)/45.
        rchin2=(r2-r1)*rtq + r1
        rchio2=(r4-r3)*rtq + r3
      else if (dsi.le.130.) then
        if (dsidif.lt.0) then
          r1=0.20894*exp(-0.026694*dsi)
        else
          r1=0.146*exp(-0.0196*dsi)
        end if
        r2=0.162*exp(-0.0216*dsi)
        r3=0.129*exp(-0.0170*dsi)
        r4=r2
        rtq=(rt-193.)/45.
        rchin2=(r2-r1)*rtq + r1
        rchio2=(r4-r3)*rtq + r3
      else if(dsi.le.135.) then
        if(dsidif.lt.0) then
          r1=2.824997*exp(-0.0467266*dsi)
        else
          r1=0.146*exp(-0.0196*dsi)
        endif
        r2=0.162*exp(-0.0216*dsi)
        r3=0.129*exp(-0.0170*dsi)
        r4=r2
        rtq=(rt-193.)/45.
        rchin2=(r2-r1)*rtq + r1
        rchio2=(r4-r3)*rtq + r3
      else if(dsi.le.160.) then
        if(dsidif.lt.0) then
          r1=2.824997*exp(-0.0467266*dsi)
        else
          r1=1.164*exp(-0.035*dsi)
        endif
        r2=0.162*exp(-0.0216*dsi)
        r3=0.1455*exp(-0.0350*dsi)
        r4=r2
        rtq=(rt-193.)/45.
        rchin2=(r2-r1)*rtq + r1
        rchio2=(r4-r3)*rtq + r3
      else
        if(dsidif.lt.0) then
          r1=1.192053*exp(-0.0413334*dsi)
        else
         r1=1.164*exp(-0.035*dsi)
        endif
        r2=0.162*exp(-0.0216*dsi)
        r3=0.1455*exp(-0.0350*dsi)
        r4=r2
        rtq=(rt-193.)/45.
        rchin2=(r2-r1)*rtq + r1
        rchio2=(r4-r3)*rtq + r3
      end if
*
*  if Temperature >= 238K
*
    else
      if (dsi.le.0.5) then
        rchin2=1.
        rchio2=1.
      else if (dsi.le.3.) then
        r1=1.
        r2=1.064*exp(-0.1235*dsi)
```

```
  rtq=(rt-238.)/58.
  rchin2=(r2-r1)*rtq + r1
  rchio2=1.
 else if (dsi.le.5.) then
  r1=1.
  r2=1.064*exp(-0.1235*dsi)
  r3=1.
  r4=3.341*exp(-0.4021*dsi)
  rtq=(rt-238.)/58.
  rchin2=(r2-r1)*rtq + r1
  rchio2=(r4-r3)*rtq + r3
 else if (dsi.le.8.) then
  r1=1.968*exp(-0.1354*dsi)
  r2=1.064*exp(-0.1235*dsi)
  r3=r1
  r4=3.341*exp(-0.4021*dsi)
  rtq=(rt-238.)/58.
  rchin2=(r2-r1)*rtq + r1
  rchio2=(r4-r3)*rtq + r3
 else if (dsi.le.20.) then
  r1=1.968*exp(-0.1354*dsi)
  r2=1.064*exp(-0.1235*dsi)
  r3=r1
  r4=0.155*exp(-0.0179*dsi)
  rtq=(rt-238.)/58.
  rchin2=(r2-r1)*rtq + r1
  rchio2=(r4-r3)*rtq + r3
 else if (dsi.le.22.) then
  r1=1.968*exp(-0.1354*dsi)
  r2=0.125*exp(-0.0164*dsi)
  r3=r1
  r4=0.155*exp(-0.0179*dsi)
  rtq=(rt-238.)/58.
  rchin2=(r2-r1)*rtq + r1
  rchio2=(r4-r3)*rtq + r3
 else if (dsi.le.50.) then
  r1=0.160*exp(-0.0214*dsi)
  r2=0.125*exp(-0.0164*dsi)
  r3=r1
  r4=0.155*exp(-0.0179*dsi)
  rtq=(rt-238.)/58.
  rchin2=(r2-r1)*rtq + r1
  rchio2=(r4-r3)*rtq + r3
 else if (dsi.le.70.) then
  r1=0.162*exp(-0.0216*dsi)
  r2=0.146*exp(-0.0196*dsi)
  r3=r1
  r4=0.238*exp(-0.0266*dsi)
  rtq=(rt-238.)/58.
  rchin2=(r2-r1)*rtq + r1
  rchio2=(r4-r3)*rtq + r3
 else if (dsi.le.140.) then
  r1=0.162*exp(-0.0216*dsi)
  r2=0.146*exp(-0.0196*dsi)
  r3=r1
  r4=0.146*exp(-0.0196*dsi)
  rtq=(rt-238.)/58.
  rchin2=(r2-r1)*rtq + r1
  rchio2=(r4-r3)*rtq + r3
 else
  r1=0.162*exp(-0.0216*dsi)
  r3=r1
  if(dsidif.lt.0) then
    r2=1.8593*exp(-0.03776*dsi)
    r4=r2
  else
    r2=0.146*exp(-0.0196*dsi)
    r4=r2
  endif
  rtq=(rt-238.)/58.
  rchin2=(r2-r1)*rtq + r1
  rchio2=(r4-r3)*rtq + r3
 end if
```

```
      end if
*
*  calculation of the chi-factor by weighting of the chi-factors
*  for n2 and o2 according to the relative abundance in the atmosphere
*  (if this is greater than 1 the chi-factor is set to 1.)
*
      if (nswco2.eq.1) then
        fco2chi=0.789*rchin2+0.211*rchio2
      else if (nswco2.eq.2) then
        fco2chi=rchin2
      else
        write(*,*) 'In fco2chi: wrong input of switch -nswco2-:'
        write(*,*) 'STOP program'
        stop
      end if
      if (fco2chi.gt.1.) fco2chi=1.
```

2.2.11.13 FLINT_PT

|-----FLINT_PT *
|-----FPARTS_PT *

Description
Calculation of the line intensity.

Variables exchanged with external modules:

Name:	Description:
flint	line intensity (the function value)
ri0	line intensity of the actual line
rel	lower state energy of the actual line
rt	equivalent temperature
dsil	central wavenumber of the actual line
ighit	HITRAN molecular code of the actual line
isohit	HITRAN isotope number of the actual line

Module structure:
1. Calculation of the line intensity

Detailed description:
1. Calculation of the line intensity
The line intensity is temperature dependent and is calculated by the following formula:

$$flint = fparts \cdot ri0 \cdot \exp\left[-rhck \cdot rel \cdot \frac{rt0int - rt}{rt0int \cdot rt}\right] \cdot \frac{1 - \exp\left[-\frac{rhck \cdot dsil}{rt}\right]}{1 - \exp\left[-\frac{rhck \cdot dsil}{rt0int}\right]}$$

rhck and *rt0int* are parameters, *fparts* is calculated by a call to the module (function) **fparts**:

$\text{fparts}[ighit, isohit, rt]$

2.2.11.14 FPARTS_PT

Description
Calculation of the temperature dependence of the Total Internal Partition Sum (TIPS) Q(T) for each molecule/isotope using Gamache's procedure QTIPS.
For all the species other than HNO_3 the TIPS functions from the '96 HITRAN data are used, while for HNO_3 the TIPS function taken from '92 HITRAN data is used.

Only the temperature range 70-405K is implemented here!

-R. R. Gamache, R. L. Hawkins, L. S. Rothman, 'Total internal partition sums for atmospheric molecules in the temperature range 70-2005K: Atmospheric linear molecules', J.Mol.Spec. 142, 205-209,1990.
-Routine from HITRAN '96 database.

Variables exchanged with external modules:

Name:	Description:
fparts	Quotient of the partition sums
ighit	HITRAN molecular code of the actual line
isohit	HITRAN isotope number of the actual line
rt	equivalent temperature

Module structure

1. Calculation of the quotient of partition sums

Detailed description:

1. Calculation of the quotient of partition sums
The partition sum are calculated for most of the HITRAN molecules using a parametrisation. This subroutine is given together with its source code.

```
*****************************************************************************
*  SUBROUTINE              : fparts
*  CREATED BY              : michael hoepfner
*  DATE OF CREATION        : 10.1.96
*  DATE OF LAST MODIFICATION  : 11.7.96
*  LAST MODIFICATION BY       : michael hoepfner
*  MODIFICATION            : HITRAN96
*  DATE OF LAST MODIFICATION  :  9.10.96
*  MODIFICATION            : hno3 calculation like GENLN2:
*                            not (t0/t)**1.5, but
*                            parametrisation: change of rqcoef(43,j) and
*                            rq296(43)
*  LAST MODIFICATION BY       : michael hoepfner
*
*  DESCRIPTION    : Calculation of the quotient of the partition sums
*                   using Gamache's procedure
*                   Sources:
*                  -R. R. Gamache, R. L. Hawkins, L. S. Rothman,
```

```
*               'Total internal partition sums for atmospheric
*                molecules in the temperature range 70-2005K:
*                Atmospheric linear molecules',J.Mol.Spec. 142,205-209
*                1990.
*              -Routine QTIPS HITRAN96
*              Only the temperature range 70-500K is implemented here!
*
*  INPUTS:
*   ighit              HITRAN molecular code of the actual line
*   isohit             HITRAN isotope number of the actual line
*   rt                 equivalent temperature
*  OUTPUTS:
*   fparts             quotient of the partition sums
*  CALLED BY: flint
*************************************************************************
**
      real*8 function fparts_pt(ighit,isohit,rt)
      implicit none
      include 'parameters_pt.inc'
      real*8 rt,rqcoef(imxiso,imxcof),rq296(imxiso)
      integer*4 ighit,isohit,j,isovec(imxhit),i1
*
*  starting positions-1 of each gas in the isotope array
*
       data isovec/
     +  0, 4,12,17,22,28,31,34,37,39,40,42,43,46,
     + 47,49,51,52,54,58,61,63,64,67,69,70,72,73,
     + 74,75,76,79,80,81,82,84/
c
c...Total internal partition sums for T>=70 to <=500 K range:
c...   H2O  --   161
      DATA (rqcoef( 1,j),j=1,4)/-.44405E+01, .27678E+00,
     +             .12536E-02,-.48938E-06/
c...   H2O  --   181
      DATA (rqcoef( 2,j),j=1,4)/-.43624E+01, .27647E+00,
     +             .12802E-02,-.52046E-06/
c...   H2O  --   171
      DATA (rqcoef( 3,j),j=1,4)/-.25767E+02, .16458E+01,
     +             .76905E-02,-.31668E-05/
c...   H2O  --   162
      DATA (rqcoef( 4,j),j=1,4)/-.23916E+02, .13793E+01,
     +             .61246E-02,-.21530E-05/
c...   CO2  --   626
      DATA (rqcoef( 5,j),j=1,4)/-.13617E+01, .94899E+00,
     +            -.69259E-03, .25974E-05/
c...   CO2  --   636
      DATA (rqcoef( 6,j),j=1,4)/-.20631E+01, .18873E+01,
     +            -.13669E-02, .54032E-05/
c...   CO2  --   628
      DATA (rqcoef( 7,j),j=1,4)/-.29175E+01, .20114E+01,
     +            -.14786E-02, .55941E-05/
c...   CO2  --   627
      DATA (rqcoef( 8,j),j=1,4)/-.16558E+02, .11733E+02,
     +            -.85844E-02, .32379E-04/
c...   CO2  --   638
      DATA (rqcoef( 9,j),j=1,4)/-.44685E+01, .40330E+01,
     +            -.29590E-02, .11770E-04/
c...   CO2  --   637
      DATA (rqcoef(10,j),j=1,4)/-.26263E+02, .23350E+02,
     +            -.17032E-01, .67532E-04/
c...   CO2  --   828
      DATA (rqcoef(11,j),j=1,4)/-.14811E+01, .10667E+01,
     +            -.78758E-03, .30133E-05/
c...   CO2  --   728
      DATA (rqcoef(12,j),j=1,4)/-.17600E+02, .12445E+02,
     +            -.91837E-02, .34915E-04/
c...    O3  --   666
      DATA (rqcoef(13,j),j=1,4)/-.16443E+03, .69047E+01,
     +             .10396E-01, .26669E-04/
c...    O3  --   668
      DATA (rqcoef(14,j),j=1,4)/-.35222E+03, .14796E+02,
     +             .21475E-01, .59891E-04/
c...    O3  --   686
```

```
      DATA (rqcoef(15,j),j=1,4)/-.17466E+03, .72912E+01,
     +            .10093E-01, .29991E-04/
c...   O3  --  667
      DATA (rqcoef(16,j),j=1,4)/-.20540E+04, .85998E+02,
     +            .12667E+00, .33026E-03/
c...   O3  --  676
      DATA (rqcoef(17,j),j=1,4)/-.10148E+04, .42494E+02,
     +            .62586E-01, .16319E-03/
c...  N2O  --  446
      DATA (rqcoef(18,j),j=1,4)/ .24892E+02, .14979E+02,
     +           -.76213E-02, .46310E-04/
c...  N2O  --  456
      DATA (rqcoef(19,j),j=1,4)/ .36318E+02, .95497E+01,
     +           -.23943E-02, .26842E-04/
c...  N2O  --  546
      DATA (rqcoef(20,j),j=1,4)/ .24241E+02, .10179E+02,
     +           -.43002E-02, .30425E-04/
c...  N2O  --  448
      DATA (rqcoef(21,j),j=1,4)/ .67708E+02, .14878E+02,
     +           -.10730E-02, .34254E-04/
c...  N2O  --  447
      DATA (rqcoef(22,j),j=1,4)/ .50069E+03, .84526E+02,
     +            .83494E-02, .17154E-03/
c...   CO  --   26
      DATA (rqcoef(23,j),j=1,4)/ .27758E+00, .36290E+00,
     +           -.74669E-05, .14896E-07/
c...   CO  --   36
      DATA (rqcoef(24,j),j=1,4)/ .53142E+00, .75953E+00,
     +           -.17810E-04, .35160E-07/
c...   CO  --   28
      DATA (rqcoef(25,j),j=1,4)/ .26593E+00, .38126E+00,
     +           -.92083E-05, .18086E-07/
c...   CO  --   27
      DATA (rqcoef(26,j),j=1,4)/ .16376E+01, .22343E+01,
     +           -.49025E-04, .97389E-07/
c...   CO  --   38
      DATA (rqcoef(27,j),j=1,4)/ .51216E+00, .79978E+00,
     +           -.21784E-04, .42749E-07/
c...   CO  --   37
      DATA (rqcoef(28,j),j=1,4)/ .32731E+01, .46577E+01,
     +           -.69833E-04, .18853E-06/
c...  CH4  --  211
      DATA (rqcoef(29,j),j=1,4)/-.26479E+02, .11557E+01,
     +            .26831E-02, .15117E-05/
c...  CH4  --  311
      DATA (rqcoef(30,j),j=1,4)/-.52956E+02, .23113E+01,
     +            .53659E-02, .30232E-05/
c...  CH4  --  212
      DATA (rqcoef(31,j),j=1,4)/-.21577E+03, .93318E+01,
     +            .21779E-01, .12183E-04/
c...   O2  --   66
      DATA (rqcoef(32,j),j=1,4)/ .35923E+00, .73534E+00,
     +           -.64870E-04, .13073E-06/
c...   O2  --   68
      DATA (rqcoef(33,j),j=1,4)/-.40039E+01, .15595E+01,
     +           -.15357E-03, .30969E-06/
c...   O2  --   67
      DATA (rqcoef(34,j),j=1,4)/-.23325E+02, .90981E+01,
     +           -.84435E-03, .17062E-05/
c...   NO  --   46
      DATA (rqcoef(35,j),j=1,4)/-.75888E+02, .79048E+01,
     +            .17555E-01,-.15606E-04/
c...   NO  --   56
      DATA (rqcoef(36,j),j=1,4)/-.29980E+02, .36479E+01,
     +            .80522E-02,-.71296E-05/
c...   NO  --   48
      DATA (rqcoef(37,j),j=1,4)/-.80558E+02, .83447E+01,
     +            .18448E-01,-.16323E-04/
c...  SO2  --  626
      DATA (rqcoef(38,j),j=1,4)/-.24056E+03, .11101E+02,
     +            .22164E-01, .52334E-04/
c...  SO2  --  646
      DATA (rqcoef(39,j),j=1,4)/-.24167E+03, .11151E+02,
```

```
     +            .22270E-01, .52550E-04/
c...  NO2  --  646
      DATA (rqcoef(40,j),j=1,4)/-.53042E+03, .24216E+02,
     +            .66856E-01, .43823E-04/
c...  NH3  --  4111
      DATA (rqcoef(41,j),j=1,4)/-.42037E+02, .25976E+01,
     +            .13073E-01,-.62230E-05/
c...  NH3  --  5111
      DATA (rqcoef(42,j),j=1,4)/-.28609E+02, .17272E+01,
     +            .87529E-02,-.41714E-05/
*...hno3   lo temperature range --  146
      data (rqcoef(43,j),j=1,4)/-.7420795579E+04, .3498357216E+03,
     +            .8905132937E-01, .3935627923E-02/
c...   OH  --  61
      DATA (rqcoef(44,j),j=1,4)/ .17478E+02, .31954E+00,
     +            .76581E-03,-.71337E-06/
c...   OH  --  81
      DATA (rqcoef(45,j),j=1,4)/ .17354E+02, .32350E+00,
     +            .76446E-03,-.70932E-06/
c...   OH  --  62
      DATA (rqcoef(46,j),j=1,4)/ .30717E+02, .13135E+01,
     +            .31430E-02,-.28371E-05/
c...   HF  --  19
      DATA (rqcoef(47,j),j=1,4)/ .15486E+01, .13350E+00,
     +            .59154E-05,-.46889E-08/
c...  HCl  --  15
      DATA (rqcoef(48,j),j=1,4)/ .28627E+01, .53122E+00,
     +            .67464E-05,-.16730E-08/
c...  HCl  --  17
      DATA (rqcoef(49,j),j=1,4)/ .28617E+01, .53203E+00,
     +            .66553E-05,-.15168E-08/
c...  HBr  --  19
      DATA (rqcoef(50,j),j=1,4)/ .27963E+01, .66532E+00,
     +            .34255E-05, .52274E-08/
c...  HBr  --  11
      DATA (rqcoef(51,j),j=1,4)/ .27953E+01, .66554E+00,
     +            .32931E-05, .54823E-08/
c...   HI  --  17
      DATA (rqcoef(52,j),j=1,4)/ .40170E+01, .13003E+01,
     +           -.11409E-04, .40026E-07/
c...  ClO  --  56
      DATA (rqcoef(53,j),j=1,4)/ .36387E+03, .28367E+02,
     +            .46556E-01, .12058E-04/
c...  ClO  --  76
      DATA (rqcoef(54,j),j=1,4)/ .37039E+03, .28834E+02,
     +            .47392E-01, .12522E-04/
c...  OCS  --  622
      DATA (rqcoef(55,j),j=1,4)/-.93697E+00, .36090E+01,
     +           -.34552E-02, .17462E-04/
c...  OCS  --  624
      DATA (rqcoef(56,j),j=1,4)/-.11536E+01, .37028E+01,
     +           -.35582E-02, .17922E-04/
c...  OCS  --  632
      DATA (rqcoef(57,j),j=1,4)/-.61015E+00, .72200E+01,
     +           -.70044E-02, .36708E-04/
c...  OCS  --  822
      DATA (rqcoef(58,j),j=1,4)/-.21569E+00, .38332E+01,
     +           -.36783E-02, .19177E-04/
c...  H2CO  --  126
      DATA (rqcoef(59,j),j=1,4)/-.11760E+03, .46885E+01,
     +            .15088E-01, .35367E-05/
c...  H2CO  --  136
      DATA (rqcoef(60,j),j=1,4)/-.24126E+03, .96134E+01,
     +            .30938E-01, .72579E-05/
c...  H2CO  --  128
      DATA (rqcoef(61,j),j=1,4)/-.11999E+03, .52912E+01,
     +            .14686E-01, .43505E-05/
c...  HOCl  --  165
      DATA (rqcoef(62,j),j=1,4)/-.73640E+03, .34149E+02,
     +            .93554E-01, .67409E-04/
c...  HOCl  --  167
      DATA (rqcoef(63,j),j=1,4)/-.74923E+03, .34747E+02,
     +            .95251E-01, .68523E-04/
```

```
c...   N2  --   44
      DATA (rqcoef(64,j),j=1,4)/ .13684E+01, .15756E+01,
     +           -.18511E-04, .38960E-07/
c...  HCN  --  124
      DATA (rqcoef(65,j),j=1,4)/-.13992E+01, .29619E+01,
     +           -.17464E-02, .65937E-05/
c...  HCN  --  134
      DATA (rqcoef(66,j),j=1,4)/-.25869E+01, .60744E+01,
     +           -.35719E-02, .13654E-04/
c...  HCN  --  125
      DATA (rqcoef(67,j),j=1,4)/-.11408E+01, .20353E+01,
     +           -.12159E-02, .46375E-05/
c... CH3Cl  --  215
      DATA (rqcoef(68,j),j=1,4)/-.91416E+03, .34081E+02,
     +            .75461E-02, .17933E-03/
c... CH3Cl  --  217
      DATA (rqcoef(69,j),j=1,4)/-.92868E+03, .34621E+02,
     +            .76674E-02, .18217E-03/
c...  H2O2  -- 1661
      DATA (rqcoef(70,j),j=1,4)/-.36499E+03, .13712E+02,
     +            .38658E-01, .23052E-04/
c...  C2H2  -- 1221
      DATA (rqcoef(71,j),j=1,4)/-.83088E+01, .14484E+01,
     +           -.25946E-02, .84612E-05/
c...  C2H2  -- 1231
      DATA (rqcoef(72,j),j=1,4)/-.66736E+02, .11592E+02,
     +           -.20779E-01, .67719E-04/
c...  C2H6  -- 1221
      DATA (rqcoef(73,j),j=1,4)/-.10000E+01, .00000E+00,
     +            .00000E+00, .00000E+00/
c...   PH3  -- 1111
      DATA (rqcoef(74,j),j=1,4)/-.15068E+03, .64718E+01,
     +            .12588E-01, .14759E-04/
c...  COF2  --  269
      DATA (rqcoef(75,j),j=1,4)/-.54180E+04, .18868E+03,
     +           -.33139E+00, .18650E-02/
c...   SF6  --   29
      DATA (rqcoef(76,j),j=1,4)/-.10000E+01, .00000E+00,
     +            .00000E+00, .00000E+00/
c...   H2S  --  121
      DATA (rqcoef(77,j),j=1,4)/-.15521E+02, .83130E+00,
     +            .33656E-02,-.85691E-06/
c...   H2S  --  141
      DATA (rqcoef(78,j),j=1,4)/-.15561E+02, .83337E+00,
     +            .33744E-02,-.85937E-06/
c...   H2S  --  131
      DATA (rqcoef(79,j),j=1,4)/-.62170E+02, .33295E+01,
     +            .13480E-01,-.34323E-05/
c... HCOOH  --  126
      DATA (rqcoef(80,j),j=1,4)/-.29550E+04, .10349E+03,
     +           -.13146E+00, .87787E-03/
c...   HO2  --  166
      DATA (rqcoef(81,j),j=1,4)/-.15684E+03, .74450E+01,
     +            .26011E-01,-.92704E-06/
c...    O  --    6
      DATA (rqcoef(82,j),j=1,4)/-.10000E+01, .00000E+00,
     +            .00000E+00, .00000E+00/
c...ClONO2  -- 5646
      DATA (rqcoef(83,j),j=1,4)/-.10000E+01, .00000E+00,
     +            .00000E+00, .00000E+00/
c...ClONO2  -- 7646
      DATA (rqcoef(84,j),j=1,4)/-.10000E+01, .00000E+00,
     +            .00000E+00, .00000E+00/
c...  NO+  --   46
      DATA (rqcoef(85,j),j=1,4)/ .91798E+00, .10416E+01,
     +           -.11614E-04, .24499E-07/
*
*  total internal partition sums at reference temperature 296 K
*
c
c               H2O 161,        181,        171,
      DATA rq296/ .174626E+03, .176141E+03, .105306E+04,
c            162;    CO2 626,        636,        628,        627,
```

```
     + .865122E+03, .286219E+03, .576928E+03, .607978E+03, .354389E+04,
c             638,        637,        828,        728;      O3 666,
     + .123528E+04, .714432E+04, .323407E+03, .376700E+04, .348186E+04,
c             668,        686,        667,        676;     N2O 446,
     + .746207E+04, .364563E+04, .430647E+05, .212791E+05, .499183E+04,
c             456,        546,        448,        447;      CO 26,
     + .334938E+04, .344940E+04, .526595E+04, .307008E+05, .107428E+03,
c              36,         28,         27,         38,         37;
     + .224704E+03, .112781E+03, .661209E+03, .236447E+03, .138071E+04,
c           CH4 211,        311,        212;      O2 66,         68,
     + .589908E+03, .117974E+04, .477061E+04, .215726E+03, .452188E+03,
c              67;      NO  46,         56,         48;     SO2 626,
     + .263998E+04, .339730E+04, .157040E+04, .358252E+04, .634449E+04,
c             646;     NO2 646;    NH3 4111,        5111;    HNO3 146;
     + .637321E+04, .136318E+05, .171089E+04, .114134E+04, .206001E+06,
cc    + .637321E+04, .136318E+05, .171089E+04, .114134E+04,.213822E+06,
c           OH 61,         81,         62;       HF 19;      HCl 15,
     + .160659E+03, .161692E+03, .621323E+03, .414625E+02, .160650E+03,
c              17;      HBr 19,         11;       HI 17;      ClO 56,
     + .160887E+03, .200165E+03, .200227E+03, .388948E+03, .131524E+05,
c              76;     OCS 622,        624,        632,        822;
     + .133824E+05, .121746E+04, .124793E+04, .247482E+04, .130948E+04,
c         H2CO 126,        136,        128;    HOCl 165,        167;
     + .268388E+04, .550322E+04, .284573E+04, .193166E+05, .196584E+05,
c           N2 44;     HCN 124,        134,        125;   CH3Cl 215,
     + .467136E+03, .893323E+03, .183657E+04, .615046E+03, .144858E+05,
c              217;   H2O2 1661;   C2H2 1221,        1231,   C2H6 1221;
     + .147153E+05, .767871E+04, .412519E+03, .330014E+04, .546265E+05,
c         PH3 1111;    COF2 269;      SF6 29;     H2S 121,         141,
     + .325067E+04, .697632E+05, .162242E+07, .503204E+03, .504486E+03,
c              131;   HCOOH 126;     HO2 166;        O 6;  ClONO2 5646,
     + .201546E+04, .389257E+05, .430184E+04,-.100000E+01, .212829E+07,
c            7646;      NO+ 46;
     + .218246E+07, .308855E+03/
*
*  if the gas is ((hno3)),c2h6,sf6,o,clono2
*
cc      if (ighit.eq.12.or.ighit.eq.27.or.ighit.eq.30.or.ighit.eq.34
cc     &    .or.ighit.eq.35)
cc     &       then
      if (ighit.eq.27.or.ighit.eq.30.or.ighit.eq.34
     &    .or.ighit.eq.35)
     &       then
        fparts_pt=(rt0int/rt)**1.5
        if (ighit.eq.34) then
         print*,'In fparts_pt: No O (HITRAN code 34)'
     &          //' should be in the database.'
         print*,' Programm STOPPED.'
         stop
        end if
*
*  for all other gases (only for temperature range 70K < rt < 500K)
*
      else
        if (rt.lt.70.or.rt.gt.500) then
          print*,'In fparts_pt: Temperature out of range:',rt,'K'
          print*,'STOP program'
          stop
        end if
*
*  position in the array of isotopes
*
        i1=isovec(ighit)+isohit
*
*  calculate quotient of total internal partition sum
*
        fparts_pt=rq296(i1) /
     &       (rqcoef(i1,1)+rqcoef(i1,2)*rt+rqcoef(i1,3)*rt*rt
     &       +rqcoef(i1,4)*rt*rt*rt)
      end if
*
*  end of function fparts_pt
*
```

end

2.2.11.15 HUMLI_PT

Description

Calculation of the Voigt lineshape.

-J. Humlicek, 'Optimized computation of the Voigt and complex probability functions', J. Quant. Radiat. Transfer, 27, 437-444,1982.

Variables exchanged with external modules:

Name:	Description:
rx	x-coefficient= $\sqrt{\ln 2}\,\frac{\sigma-\sigma_0}{\alpha_D}$ (α_D : Doppler Half Width, σ: actual wavenumber, σ_0: central line wavenumber)
ry	y-coefficient= $\sqrt{\ln 2}\,\frac{\alpha_L}{\alpha_D}$ (α_L : Lorentz Half Width)
rre	$\text{Voigt - lineshape} \cdot \alpha_D \cdot \sqrt{\frac{\pi}{\ln 2}}$

Module structure

1. Calculation of the Voigt lineshape

Detailed description

1. Calculation of the Voigt lineshape

The Humlicek algorithm calculates the complex probability function. In our case we only use the real part which corresponds to the Voigt line profile. For the description we refer to the article by Humlicek (see above).

2.2.11.16 POLCOE2ND_PT

Description
Calculation of the coefficients for a 2nd order polynominal interpolation.

Variables exchanged with external modules:

Name:	Description:
dx	the three x-points where the y values are given
dy	the y values belonging to dx
dcof	the 3 coefficients of the polynom: $y = dcof(1) + dcof(2) \cdot x + dcof(3) \cdot x^2$

Module structure
1. Calculation of the coefficient.

Detailed description:

1. Calculation of the coefficients
The coefficients for the parabolic interpolation are calculated using the formulas:

$$dcof(1) = \frac{\left[\begin{array}{l} dx(1)^2 \cdot \left(dx(3) \cdot dy(2) - dx(2) \cdot dy(3)\right) + dx(2)^2 \cdot \left(dx(1) \cdot dy(3) - dx(3) \cdot dy(1)\right) \\ + dx(3)^2 \cdot \left(dx(2) \cdot dy(1) - dx(1) \cdot dy(2)\right) \end{array}\right]}{dx(1)^2 \cdot \left(dx(3) - dx(2)\right) + dx(2)^2 \cdot \left(dx(1) - dx(3)\right) + dx(3)^2 \cdot \left(dx(2) - dx(1)\right)}$$

$$dcof(2) = \frac{dx(1)^2 \cdot \left(dy(3) - dy(2)\right) + dx(2)^2 \cdot \left(dy(1) - dy(3)\right) + dx(3)^2 \cdot \left(dy(2) - dy(1)\right)}{dx(1)^2 \cdot \left(dx(3) - dx(2)\right) + dx(2)^2 \cdot \left(dx(1) - dx(3)\right) + dx(3)^2 \cdot \left(dx(2) - dx(1)\right)}$$

$$dcof(3) = \frac{dx(1) \cdot \left(dy(2) - dy(3)\right) + dx(2) \cdot \left(dy(3) - dy(1)\right) + dx(3) \cdot \left(dy(1) - dy(2)\right)}{dx(1)^2 \cdot \left(dx(3) - dx(2)\right) + dx(2)^2 \cdot \left(dx(1) - dx(3)\right) + dx(3)^2 \cdot \left(dx(2) - dx(1)\right)}$$

$$dcof(1) = \frac{\left[\begin{array}{l} dx(1)^2 \cdot \left(dx(3)\cdot dy(2) - dx(2)\cdot dy(3)\right) + dx(2)^2 \cdot \left(dx(1)\cdot dy(3) - dx(3)\cdot dy(1)\right) \\ +dx(3)^2 \cdot \left(dx(2)\cdot dy(1) - dx(1)\cdot dy(2)\right) \end{array}\right]}{dx(1)^2 \cdot \left(dx(3) - dx(2)\right) + dx(2)^2 \cdot \left(dx(1) - dx(3)\right) + dx(3)^2 \cdot \left(dx(2) - dx(1)\right)}$$

$$dcof(2) = \frac{dx(1)^2 \cdot \left(dy(3) - dy(2)\right) + dx(2)^2 \cdot \left(dy(1) - dy(3)\right) + dx(3)^2 \cdot \left(dy(2) - dy(1)\right)}{dx(1)^2 \cdot \left(dx(3) - dx(2)\right) + dx(2)^2 \cdot \left(dx(1) - dx(3)\right) + dx(3)^2 \cdot \left(dx(2) - dx(1)\right)}$$

$$dcof(3) = \frac{dx(1)\cdot \left(dy(2) - dy(3)\right) + dx(2)\cdot \left(dy(3) - dy(1)\right) + dx(3)\cdot \left(dy(1) - dy(2)\right)}{dx(1)^2 \cdot \left(dx(3) - dx(2)\right) + dx(2)^2 \cdot \left(dx(1) - dx(3)\right) + dx(3)^2 \cdot \left(dx(2) - dx(1)\right)}$$

2.2.11.17 LOFICO_PT

Description
Calculation of the local fine grid and the local coarse grid for the actual atmopheric path.

Variables exchanged with external modules:

Name:	Dimension	Description:
dsilin	imxlin imxmw	central wavenumber for each line of each Mw
ipo		actual path number
imw		actual microwindow number
rwmol	imxhit, imxism	molecular weight for each HITRAN molecular code and isotope number
icode	imxlin, imxmw	HITRAN code for each line of each Mw
iiso	imxlin, imxmw	isotope number for each line of each Mw
rhw0	imxlin, imxmw	foreign broadened half width for each line of each Mw
imaxli		number of the line of the main gas with largest intensity
rpeq	imxpat, imxgas	equivalent pressures
rteq	imxpat, imxgas	equivalent temperatures
delta		general fine grid interval [cm-1]
isigma	imxmw	number of wavenumber grid points for each Mw
dsigma	imxsig, imxmw	general wavenumber fine grid
igasmw	imxmw	number of gases to be considered in each Mw
rexph	imxlin, imxmw	exponent for T dependence of half width for each line of each Mw
iqlfgf		ratio between local fine and general fine grid
dsiglf	imxsig	local fine grid [cm^{-1}]
dsiglc	imxsig	local coarse grid [cm^{-1}]

isiglf		number of local fine grid points
isiglc		number of local coarse grid points
deltalf		distance between local fine grid points
deltalc		distance between local coarse grid points
rcrolf	imxsig, imxgmw	cross sections on local fine grid
rcrolc	imxsig, imxgmw	cross sections on local coarse grid
rcrolfpert	imxsig,2	cross section on local fine grid for temperature perturbed cross sections
rcrolcpert	imxsig,2	cross section on local coarse grid for temperature perturbed cross sections

Module structure:

1. Determination of the local fine and coarse grid.
2. Initialisation of the cross section vector on the fine and coarse grid.

Detailed description:

1. Determination of the local fine and coarse grid.

The doppler and lorentz half width for the most intense line of the main gas is calculated.

$$rdhalf = dsilin(imaxli,imw) \cdot dcdop \cdot \sqrt{\frac{rteq(ipo,1)}{rwmol(icode(imaxli,imw),iiso(imaxli,imw))}}$$

$$rlhalf = rhw0(imaxli,imw) \cdot \frac{rpeq(ipo,1)}{rp0h} \cdot \left[\frac{rt0h}{rteq(ipo,1)}\right]^{rexph(imaxli,imw)}$$

Calculation of the (approximate) Voigt half width:

$$rvhalf = rdhalt + rlhalf$$

Determine local fine grid:
For $rvlf \cdot rvhalf < delta$:

$$iqlfgf = 1$$
$$deltalf = delta$$

For $rvlf \cdot rvhalf > delta$:

$$iqlfgf = int\left(\frac{rvlf \cdot rvhalf}{delta}\right)$$
$$deltalf = iqlfgf \cdot delta$$

Define the local coarse grid:

$$deltalc = iqlclf \cdot deltalf$$

Number of local fine grid points:

$$isiglf = int\left(\frac{dsigma(isigma(imw),imw) - dsigma(1,imw)}{deltalf}\right) + 2$$

Fill local fine grid vector:

$$dsiglf\,(1) = dsigma(1,imw)$$
$$dsiglf\,(i) = dsiglf\,(i-1) + deltalf$$

Number of local coarse grid points:

$$isiglc = int\left(\frac{dsigma(isigma(imw),imw) - dsigma(1,imw)}{deltalc}\right) + 2$$

Fill local coarse grid vector:

$$dsiglc(1) = dsigma(1,imw)$$
$$dsiglc(i) = dsiglc(i-1) + deltalc$$

2. Initialisation of the cross section vector on the fine and coarse grid.

The vectors *rcrolf, rcrolc, rcrolfpert* and *rcrolcpert* are initialized to 0.

2.2.11.18 SPECTRUM_PT

SPECTRUM_PT]
|-----CONV_PT*

Description

- calculation of the original spectra, the temperature perturbed spectra, and the derivatives with respect to the continuum on the general wavenumber fine grid for all geometries of the actual microwindow
- convolution of these spectra and derivatives with the AILS function to the general coarse wavenumber grid
- if an irregular grid is available, the calculation of the high resolution spectrum, the temperature perturbed spectra, and the derivatives with respect to the continuum is made on the so-called ‘compressed grid’ (the one made with only the ‘1’ points of the irregular grid) , then a direct interpolation and convolution is performed.

Variables exchanged with external modules:

Name:	Description:
imw	number of the actual Mw
itglev	number of the tangent-level for each geometry
igasmw	number of gases to be considered in each Mw
igasnr	global gas number for the local gas number of each Mw
isigma	number of wavenumber grid points for each Mw
rcross	absorption cross sections for each wavenumber each IAPT and each gas for the actual MW
rcol	column amounts for each layer, each geometry and each gas
raircol	air-column for each layer and each geometry
ipoint	IAPT-number for each layer and each geometry
ipath	number of different IAPT-numbers of ipoint
rtmain	equivalent temperature of the main gas
dsigma	general wavenumber fine grid
igeo	number of simulated geometries
iocsim	occupation matrix for the simulations to be performed
nsam	n. of sampling points in each Mw (general coarse grid)
nils	number of elements of rils
rspct	spectrum for each geometry on the general coarse grid 1st index: general wavenumber coarse grid 2nd index: geometries to be simulated for the actual Mw
rils	instrument-line-shape function on the general fine grid
rintils	ratio between the frequency step approximating infinitesimal spectral resolution and the integral of the ILS function
nrd	Ratio between general coarse grid step and fine grid step
rclay	model-layer values of the continuum
iderlay	highest (x,1), lowest (x,3) and middle (x,2) (the one directly above the 'perturbed' layer) which is affected by each derivative
igeocder	for each geometry the highest (x,1) and lowest (x,2) continuum derivative (in the parameter-grid) which has to be calculated
rpartcder	partial derivatives of the continuum layer values with respect to the parameter-level values
rspctcder	continuum derivative spectra on the general coarse grid for each geometry and each parameter level 1st index: general wavenumber coarse grid 2nd index: geometries to be simulated for the actual Mw 3rd index: levels where the parameters are retrieved
igeotder	for each geometry the highest (x,1) and lowest (x,2) temperature derivative (in the parameter-grid) which has to be calculated
ipar	number of parameter-levels
rteqpert	equivalent temperatures of the main gas for the perturbed temperature profiles
rcolpert	columns of the main gas for the perturbed temperature profiles
rcrossper t	cross-sections for the perturbed temperature profiles
rspcttpert	temperature perturbed spectra on the general coarse grid for each geometry and each parameter level

	1st index: general wavenumber coarse grid 2nd index: geometries to be simulated for the actual Mw 3rd index: levels where the parameters are retrieved
cint	character*3: *cint(imxmw):* it indicates, for each microwindow, what kind of interpolation has to be performed between the spectral points of the irregular grid.
lirrgridmw	logical: *lirrgridmw(imxmw):* logical vector that, for each selected microwindow in the actual retrieval, indicates whether the irregular grid is available.
igridc	integer*4: igridc(imxsi2,imxmw): matrix that, to each microwindow and each point of the compressed grid, associates the corresponding value on the regular fine grid.
nused1	integer*4: nused1(imxmw): total number of points of the compressed grid for each microwindow
rsan	real*8: rsan(imxi,imxsi2,4,imxmw): variable used for making the direct interpolation/convolution. *rsan(jsam,i,n,imw)=* $\sum_{j=\max(((jsam-1)*nrd+1),igridc(i,imw))-(jsam-1)*nrd,\,k=\max(0,-igridc(i,imw)+((jsam-1)\cdot nrd+1))}^{j=\min(igridc(i+1,imw)-1,nils-1+(jsam-1)*nrd)-(jsam-1)*nrd,\,k=\min(igridc(i+1)-1-igridc(i),((jsam-1)\cdot nrd+nils-igridc(i))} rils(nils-j+1)\cdot k^{n-1}$
ilim	integer*4: ilim(2,imxi,imxmw): variable used for making the direct interpolation/convolution : *ilim(1,jsam,imw):* first point of the compressed grid to be considered for the computation of the low resolution spectral point at *jsam* for microwindow *imw*; *ilim(2,jsam,imw):* total number of points of the compressed grid to be considered for the computation of the low resolution spectral point at *jsam* for microwindow *imw*.

Module structure:

1. Initialisation of variables and Planck function for later interpolation

Begin loop 1 over geometries valid for the actual microwindow

2. Initialisation of variables

Begin loop 2 over general wavenumber fine grid

Begin condition 1: the value of the spectrum at this wavenumber has to be calculated and not interpolated

3. Interpolate Planck function

Begin loop 3 over the layers of the actual geometry

4. Calculation of the transmissions

End loop 3

5. Calculation of the radiative transfer

Begin loop 4 over the layers of the actual geometry for which the continuum derivatives are calculated

6. Calculation of the continuum derivatives with respect to the continuum layer values

End loop 4

Begin loop 5 over the levels for which the continuum derivatives are calculated

7. Calculation of the continuum derivatives with respect to the continuum level values

End loop 5
Begin loop 6 over the levels which are T-perturbed for the actual geometry
Begin loop 7 over the layers of the actual geometry
8. Calculation of the T-perturbed transmissions
End loop 7
9. Calculation of radiative transfer for the T-perturbed spectra
End loop 6
End condition 1
End loop 2
Begin condition 2: irregular grid is available for the actual microwindow
Begin condition 3: cubic interpolation has to be used
10. Computation of coefficients for the cubic interpolation for spectrum, temperature perturbed spectrum and continuum derivative
11. Direct cubic interpolation / convolution
else condition 3: linear interpolation has to be used
12. Computation of coefficients for the linear interpolation for spectrum, temperature perturbed spectrum and continuum derivative
13. Direct linear interpolation / convolution
End condition 3
else condition 2
14. Convolution of the spectra and derivatives with the AILS function
End condition 2
15. Finalisation of the temperature perturbed spectra
End loop 1

Detailed description:

loop 1 over geometries valid for the actual microwindow
jgeo=1→*igeo*
if (*iocsim(jgeo,imw)*≠0)

loop 2 over general wavenumber fine grid
ksigma=1→*isigma(imw)*

condition 1: the value of the spectrum corresponding to point ksigma has to be calculated and not interpolated.
Operations 3-9 have to be performed only if either an irregular grid is not available for the considered mw or the irregular grid is available but the point ksigma corresponds to '1' on the irregular grid.
if (.not. lirrgridmw(imw) .or. (lirrgridmw(imw) .and. iigrid(ksigma, imw, 1) .eq. 1))

loop 3 over the layers of the actual geometry
klay=1→*nlay*
nlay=itglev(jgeo)-1 is the tangent layer.

loop 4 over the layers of the actual geometry for which the continuum derivatives are calculated
jder=iderlay(igeocder(jgeo,1),1)→nlay

iderlay(igeocder(jgeo,1),1) is the highest layer for which the continuum derivatives have to be determined.

loop 5 over the levels for which the continuum derivatives are calculated
jder=igeocder(jgeo,1)→igeocder(jgeo,2)

loop 6 over the levels which are T-perturbed for the actual geometry
jpert=igeotder(jgeo,1)→igeotder(jgeo,2)

loop 7 over the layers of the actual geometry
klay=1→itglev(jgeo)-1

Condition 2: irregular grid is available for the actual microwindow
Only if an irregular grid is available for the actual microwindow (if lirrgridmw(imw) is true), either operations 10. and 11. or operations 12. and 13. (i.e. direct interpolation / convolution) are performed, otherwise only convolution is performed.

Begin condition 3: cubic interpolation has to be used
if (cint(imw) .eq. 'cub' .or. 'CUB'), cubic interpolation has to be performed between the points of the spectrum on the compressed grid; if this is not the case, it means that *(cint(imw) .eq. 'lin' .or. 'LIN')*, and as consequence linear interpolation has to be performed between the points of the spectrum on the compressed grid.

1. Initialisation of variables and Planck function for later interpolation
Output variables set to 0.

The total number of points *nsig* of the grid to be used for the Radiative Transfer computation is determined. If an irregular grid is available, the compressed grid is used and *nsig= nused1(imw),* if the irregular grid is not available, *nsig= isigma(imw).*

The Planck function values at the first grid point and the last grid point of the actual microwindow and from this the increment for the later linear interpolation is calculated for the temperatures of the unperturbed profiles of the main gas (*rtmain*) and the temperatures for the perturbed profiles (*rteqpert*). This is done for all different IAPT-numbers ($1 \le jpath \le ipath$).
The formula used for the Planck function is:

$$B = \frac{rc1 \cdot \sigma^3}{\exp\left[\frac{rhck \cdot \sigma}{T}\right] - 1}$$

T = rtmain(jpath) and *T = rteq(jpath,1* and *2)*
σ *= dsigma(1,imw)* or σ *= dsigma(isigma(imw),imw)*
(rc1, rhck: parameters)

2. Initialisation of variables
Here, it is taken care that during the following calculations the middle (*iderlay(jpert,2)*) and the lowest (*iderlay(jpert,3)*) temperature perturbed layer is not lower than the tangent layer (*itglev(jgeo)*-1). Otherwise they are set to *itglev(jgeo)*-1.

These variables have to be set to their old value before the end of loop 1!

if [*iderlay(jpert,2)* > *itglev(jgeo)*-1] then *iderlay(jpert,2* and *3)* = *itglev(jgeo)*-1

3. Interpolate Planck function

Using the values calculated in 1. the Planck function is linearly interpolated to the actual wavenumber for all IAPT numbers (*jpath=1→ipath)*:
The results are the interpolated Planck function values for the original T-profile (*rtmain):*
db(jpath),

and for the perturbed T-profiles *(rteqpert(jpath,1), rteqpert(jpath,2)):*
dbpert(jpath,1), dbpert(jpath,2)

Care has to be taken to perform a correct interpolation of the Plank function when the compressed grid is used (i.e. if lirrgridmw(imw) = true): in this case the value of the Planck function corresponding to the actual point *i* of the compressed grid is obtained adding to the Planck function value at the first grid point the product of the coefficient of the linear interpolation times *(igridc(i,imw)-1)*.

4. Calculation of the transmission

The transmission for each layer is calculated by the formula:

$$rtau(klay) = \exp-\left[\sum_{mgas=1}^{igas}\left\{\begin{array}{l} rclay(klay,imw)\cdot raircol(klay,imw)*10^{-30} + \\ rcross(ksig,ipoint(klay,\ jgeo),mgas)\cdot \\ rcol(klay,\ jgeo,igasnr(mgas,imw)) \end{array}\right\}\right]$$

Two other variables are also determined:

$$rtau1(klay) = \prod_{l=1}^{klay-1} rtau(l)$$

and:

$$rtau2(klay) = rtau1(klay)\cdot rtau(klay)\cdot \prod_{l=klay+1}^{nlay} rtau(l)^2$$

with: $nlay = itglev(jgeo) - 1$, the number of layers for the actual geometry,

and the definition: $\prod_{l=m}^{m-1} x_l \equiv 1$.

5. Calculation of the radiative transfer

The spectrum is determined by the equation:

$$rsp(ksig) = \sum_{klay=1}^{nlay} db(ipoint(klay, jgeo)) \cdot (1 - rtau(klay))(rtau1(klay) - rtau2(klay))$$

with: $nlay = itglev(jgeo) - 1$,
and *db*, the value of the Planck function for each IAPT-number. *db* was determined in 3. by linear interpolation to the actual general fine grid wavenumber.

6. Calculation of the continuum derivatives with respect to the continuum layer values
In this section the continuum derivatives with respect to the layer values of the continuum are calculated for the layers (*jder*). The formula is:

$$rcder2(jder) = -raircol(jder, jgeo) \cdot 10^{-30}$$

$$\left[\begin{array}{l} \sum_{klay=1}^{jder-1} 2 \cdot rtau2(klay) \cdot db(ipoint(klay, jgeo)) \cdot (1 - rtau(klay)) \\ + db(ipoint(jder, jgeo)) \cdot \left(\begin{array}{l} rtau2(jder) - \\ rtau(jder) \cdot (rtau1(jder) + 2 \cdot rtau2(jder)) \end{array} \right) \\ + \sum_{klay=jder+1}^{nlay} \left(\begin{array}{l} db(ipoint(klay, jgeo)) \cdot (1 - rtau(klay)) \cdot \\ (rtau1(klay) + rtau2(klay)) \end{array} \right) \end{array} \right]$$

7. Calculation of the continuum derivatives with respect the continuum level values
The continuum derivatives with respect to the continuum level values (*rcder*) are determined by using the results from 6. which are multiplied by the input *rpartcder*:

$$rcder(ksig, jder) = \sum_{klay=iderlay(jder,1)}^{iderlay(jder,3)} rcder2(klay) \cdot rpartcder(klay, jder, imw)$$

8. Calculation of the T-perturbed transmissions
The transmissions (*rtaupert)* and the variables *rtau1pert* and *rtau2pert* are determined in order to use them in 9. for the calculation of the temperature perturbed spectra:

For the layers which are not perturbed, i.e. for:
$1 \le klay \le iderlay(jpert,1)\text{-}1$ and $iderlay(jpert,3)+1 \le klay \le nlay$:

$$rtaupert(klay) = rtau(klay)$$

and for the perturbed layers above the perturbed level *jpert*:
$iderlay(jpert,1) \le klay \le iderlay(jpert,2)$:

$$rtaupert(klay) = \exp - \left[\begin{array}{l} rclay(klay, imw) \cdot raircol(klay, imw) \cdot 10^{-30} + \\ rcrosspert(ksig, ipoint(klay,\ jgeo), 1) \cdot rcolpert(klay,\ jgeo, 1) + \\ \sum_{mgas=2}^{igas} \left\{ \begin{array}{l} rcross(ksig, ipoint(klay,\ jgeo), mgas) \cdot \\ rcol(klay,\ jgeo, igasnr(mgas, imw)) \end{array} \right\} \end{array} \right]$$

and for the perturbed layers below the perturbed level *jpert*:
iderlay(jpert,2)+1 $\leq$ *klay* $\leq$ *iderlay(jpert,3)*:

$$rtaupert(klay) = \exp - \left[\begin{array}{l} rclay(klay, imw) \cdot raircol(klay, imw) \cdot 10^{-30} + \\ rcrosspert(ksig, ipoint(klay,\ jgeo), 2) \cdot rcolpert(klay,\ jgeo, 2) + \\ \sum_{mgas=2}^{igas} \left\{ \begin{array}{l} rcross(ksig, ipoint(klay,\ jgeo), mgas) \cdot \\ rcol(klay,\ jgeo, igasnr(mgas, imw)) \end{array} \right\} \end{array} \right]$$

rtau1pert and *rtau2pert* are determined from *rtaupert* like *rtau1* and *rtau2* from *rtau* in 4.

9. Calculation of radiative transfer for the T-perturbed spectra

The temperature perturbed spectra are determined by the formula equal to the one in 5., but only using the temperature perturbed Planck functions *dpert* for the corresponding layers:

$$\begin{aligned} rsppert(ksig, jpert) = & \sum_{klay=1}^{iderlay(jpert,1)-1} \left\{ \begin{array}{l} db(ipoint(klay, jgeo)) \cdot \\ \left(1 - rtaupert(klay)\right) \cdot \\ \left(rtau1pert(klay) - rtau2pert(klay)\right) \end{array} \right\} \\ & + \sum_{klay=iderlay(jpert,1)}^{iderlay(jpert,2)} \left\{ \begin{array}{l} dbpert(ipoint(klay, jgeo), 1) \cdot \\ \left(1 - rtaupert(klay)\right) \cdot \\ \left(rtau1pert(klay) - rtau2pert(klay)\right) \end{array} \right\} \\ & + \sum_{klay=iderlay(jpert,2)+1}^{iderlay(jpert,3)} \left\{ \begin{array}{l} dbpert(ipoint(klay, jgeo), 2) \cdot \\ \left(1 - rtaupert(klay)\right) \cdot \\ \left(rtau1pert(klay) - rtau2pert(klay)\right) \end{array} \right\} \\ & + \sum_{klay=iderlay(jpert,3)+1}^{nlay} \left\{ \begin{array}{l} db(ipoint(klay, jgeo)) \cdot \\ \left(1 - rtaupert(klay)\right) \cdot \\ \left(rtau1pert(klay) - rtau2pert(klay)\right) \end{array} \right\} \end{aligned}$$

with: $nlay = itglev(jgeo) - 1$.

10. Computation of coefficients for the cubic interpolation for spectrum,

temperature perturbed spectra and continuum derivatives

For each point *i* of the compressed grid between *2* and *(nused1(imw)-2)*,

do i=2,nsig-2

the coefficients of the cubic interpolation *a, b, c* according to the following equation:

$$y = y_2 + a \cdot (x - x_2)^3 + b \cdot (x - x_2)^2 + c \cdot (x - x_2),$$

with (x_2, y_2) coordinates of the second of the four points used for making the interpolation,

for spectrum, temperature perturbed spectrum and continuum derivatives, are computed in two steps.
First of all the variables which are independent on the value of the spectrum, temperature perturbed spectrum and continuum derivatives are computed:

```
ii2=igridc(i,imw)
ii3=igridc(i+1,imw)
ii1=igridc(i-1,imw)
ii4=igridc(i+2,imw)
iD12 =  ii1 - ii2
iD13 =  ii1 - ii3
iD14 =  ii1 - ii4
iD23 =  ii2 - ii3
iD24 =  ii2 - ii4
iD34 =  ii3 - ii4
rdc1=1.d0/dble(iD12*iD13*iD14)
rdc2=1.d0/dble(iD12*iD23*iD24)
rdc3=1.d0/dble(iD13*iD23*iD34)
rdc4=1.d0/dble(iD14*iD24*iD34)
```

Then the variables c1, c2, c3, c4, dependent on the four points through which the interpolating polynomial is drawn, are computed: in the case of the spectrum we have:

```
c1=rsp(i-1)*rdc1
c2=-rsp(i)*rdc2
c3=rsp(i+1)*rdc3
c4=-rsp(i+2)*rdc4
```

In the case of temperature perturbed spectra:

```
c1= rsppert(i-1,jpert)*rdc1
c2=- rsppert(i,jpert)*rdc2
c3= rsppert (i+1,jpert)*rdc3
c4=- rsppert(i+2,jpert)*rdc4,
```

jpert= igeotder(jgeo,1) → igeotder(jgeo,2)

In the case of continuum derivatives:

*c1=rcder(i-1,jder)*rdc1*
*c2=- rcder(i,jder)*rdc2*
*c3= rcder (i+1,jder)*rdc3*
*c4=- rcder(i+2,jder)*rdc4,*

jder= igeocder(jgeo,1) → igeocder(jgeo,2)

The *a, b, c* coefficients are determined by the following equation:

```
a(i)=c1+c2+c3+c4
b(i)= =-(c1*dble(ii4+ii3+ii2)+c2*dble(ii4+ii3+ii1)+c3*dble(ii4+ii2+ii1)+
        c4*dble(ii3+ii2+ii1))+3.0d0*a(i)*dble(ii2)
c(i)= c1*dble(ii4*ii3+ii4*ii2+ii2*ii3)+c2*dble(ii1*ii3+ii4*ii1+ii4*ii3)+
      c3*dble(ii1*ii2+ii4*ii1+ii4*ii2)+c4*dble(ii1*ii2+ii3*ii1+ii2*ii3)+
      +3.0d0*a(i)*dble(ii2*ii2) -2.0d0*dble(ii2)*(c1*dble(ii4+ii3+ii2)+
      +c2*dble(ii4+ii3+ii1)+c3*dble(ii4+ii2+ii1)+c4*dble(ii3+ii2+ii1))
```

The coefficients *a, b, c* for the temperature perturbed spectra are stored in the following matrices:

apert(i, jpert = igeotder(jgeo,1) → igeotder(jgeo,2))
bpert(i, jpert = igeotder(jgeo,1) → igeotder(jgeo,2))
cpert(i, jpert = igeotder(jgeo,1) → igeotder(jgeo,2))

The coefficients for the continuum derivatives are stored in the following matrices:

ader(i, jder= igeocder(jgeo,1) → igeocder(jgeo,2))
bder(i, jder= igeocder(jgeo,1) → igeocder(jgeo,2))
cder(i, jder= igeocder(jgeo,1) → igeocder(jgeo,2))

11. Direct cubic convolution and interpolation

For the meaning of subsequent computations, please refer to description of routine read_irregular grid_pt, par. 2.2.30.

The direct interpolation / convolution is performed using the coefficients computed in 10. and the matrices *rsan(imxi,imxsi2,4,imxmw)* and *lim(2,imxi,imxmw)* computed by routine read_irrgrid_pt.f.

For each *jsam* between *2* and *(nsam(imw)-1)*, the value of the low resolution spectrum is computed using the following formula:

$$rspct(jsam, jgeo) = \sum_{i\lim(1,jsam,imw)}^{i\lim(1,jsam,imw)+i\lim(2,jsam,imw)-1} \left(\begin{array}{l} rsp(i)\cdot rsan(jsam,i,1,imw) + a(i)\cdot rsan(jsam,i,4,imw) + \\ +b\cdot rsan(jsam,i,3,imw) + c\cdot rsan(jsam,i,2,imw) \end{array} \right)$$

At the end the low resolution spectrum is normalised:

$$rspct(jsam, jgeo) = rspct(jsam, jgeo) * rintils(imw)$$

Since the first and the last point of the regular fine grid had not been taken into account during the computation of *rsan*, an addition summation has to be performed for *jsam =1* and *jsam=nsam(imw).*

For *jsam=1*:

$$rspct(1, jgeo) = rsp(1) \cdot rils(nils, imw) + \sum_{i\lim(1,1,imw)}^{i\lim(1,1,imw)+i\lim(2,1,imw)-1} \begin{pmatrix} rsp(i) \cdot rsan(1,i,1,imw) + a(i) \cdot rsan(1,i,4,imw) + \\ +b \cdot rsan(1,i,3,imw) + c \cdot rsan(1,i,2,imw) \end{pmatrix}$$

$$rspct(jsam, jgeo) = rspct(jsam, jgeo) * rintils(imw)$$

For *jsam=j=nsam(imw):*

$$rspct(j, jgeo) = rsp(nsig) \cdot rils(1, imw) + \sum_{i\lim(1,j,imw)}^{i\lim(1,j,imw)+i\lim(2,j,imw)-1} \begin{pmatrix} rsp(i) \cdot rsan(j,i,1,imw) + a(i) \cdot rsan(j,i,4,imw) + \\ +b \cdot rsan(j,i,3,imw) + c \cdot rsan(j,i,2,imw) \end{pmatrix}$$

$$rspct(jsam, jgeo) = rspct(jsam, jgeo) * rintils(imw)$$

The same operations have to be performed also for all the temperature perturbed spectra (*rspcttpert(jsam=1→nsam(imw),jgeo,jpert = igeotder(jgeo,1) → igeotder(jgeo,2))*) and the continuum derivatives (*rspctcder(jsam=1→nsam(imw),jgeo,jder = igeocder(jgeo,1)→igeocder(jgeo,2))*).

```
      j=nsam(imw)

      r8fac=rsp(1)*rils(nils,imw)
     do i=ilim(1,1,imw),ilim(1,1,imw)+
    &                ilim(2,1,imw)-1
      r8fac=r8fac+
    &       rsp(i)*rsan(1,i,1,imw)+
    &       a(i)*rsan(1,i,4,imw)+
    &       b(i)*rsan(1,i,3,imw)+
    &       c(i)*rsan(1,i,2,imw)
     end do
      rspct(1,jgeo)=r8fac*rintils(imw)

    do jsam=2,j-1
     do i=ilim(1,jsam,imw),ilim(1,jsam,imw)+
    &                ilim(2,jsam,imw)-1
      rspct(jsam,jgeo)=rspct(jsam,jgeo)+
    &              rsp(i)*rsan(jsam,i,1,imw)+
    &              a(i)*rsan(jsam,i,4,imw)+
```

```
&              b(i)*rsan(jsam,i,3,imw)+
&              c(i)*rsan(jsam,i,2,imw)
  end do
   rspct(jsam,jgeo)=rspct(jsam,jgeo)*rintils(imw)
 end do


   r8fac=rsp(nsig)*rils(1,imw)
  do i=ilim(1,j,imw),ilim(1,j,imw)+
&              ilim(2,j,imw)-1
   r8fac=r8fac+
&       rsp(i)*rsan(j,i,1,imw)+
&       a(i)*rsan(j,i,4,imw)+
&       b(i)*rsan(j,i,3,imw)+
&       c(i)*rsan(j,i,2,imw)
  end do
   rspct(j,jgeo)=r8fac*rintils(imw)

   do jpert=igeotder(jgeo,1),igeotder(jgeo,2)
   r8fac=rsppert(1,jpert)*rils(nils,imw)
   do i=ilim(1,1,imw),ilim(1,1,imw)+
&                 ilim(2,1,imw)-1
    r8fac= r8fac+
&         rsppert(i,jpert)*rsan(1,i,1,imw)+
&         apert(i,jpert)*rsan(1,i,4,imw)+
&         bpert(i,jpert)*rsan(1,i,3,imw)+
&         cpert(i,jpert)*rsan(1,i,2,imw)
   end do
   rspcttpert(1,jgeo,jpert)=r8fac*rintils(imw)

  do jsam=2,j-1

   r8fac=0.d0
   do i=ilim(1,jsam,imw),ilim(1,jsam,imw)+
&                 ilim(2,jsam,imw)-1
   r8fac=r8fac+rsppert(i,jpert)*rsan(jsam,i,1,imw)+
&            apert(i,jpert)*rsan(jsam,i,4,imw)+
&            bpert(i,jpert)*rsan(jsam,i,3,imw)+
&            cpert(i,jpert)*rsan(jsam,i,2,imw)
   end do
   rspcttpert(jsam,jgeo,jpert)=r8fac*rintils(imw)
  end do

   r8fac=rsppert(nsig,jpert)*rils(1,imw)
   do i=ilim(1,j,imw),ilim(1,j,imw)+
&                 ilim(2,j,imw)-1
    r8fac= r8fac+
&         rsppert(i,jpert)*rsan(j,i,1,imw)+
&         apert(i,jpert)*rsan(j,i,4,imw)+
```

```
&        bpert(i,jpert)*rsan(j,i,3,imw)+
&        cpert(i,jpert)*rsan(j,i,2,imw)
   end do
   rspcttpert(j,jgeo,jpert)=r8fac*rintils(imw)

   end do

  do jder=igeocder(jgeo,1),igeocder(jgeo,2)

  r8fac=rcder(1,jder)*rils(nils,imw)
   do i=ilim(1,1,imw),ilim(1,1,imw)+
&                ilim(2,1,imw)-1
    r8fac=r8fac+rcder(i,jder)*rsan(1,i,1,imw)+
&             ader(i,jder)*rsan(1,i,4,imw)+
&             bder(i,jder)*rsan(1,i,3,imw)+
&             cder(i,jder)*rsan(1,i,2,imw)
   end do
   rspctcder(1,jgeo,jder)=r8fac*rintils(imw)

  do jsam=2,j-1
   r8fac=0.d0
   do i=ilim(1,jsam,imw),ilim(1,jsam,imw)+
&                ilim(2,jsam,imw)-1
    r8fac=r8fac+rcder(i,jder)*rsan(jsam,i,1,imw)+
&           ader(i,jder)*rsan(jsam,i,4,imw)+
&           bder(i,jder)*rsan(jsam,i,3,imw)+
&           cder(i,jder)*rsan(jsam,i,2,imw)
   end do
   rspctcder(jsam,jgeo,jder)=r8fac*rintils(imw)
  end do

  r8fac=rcder(nsig,jder)*rils(1,imw)
   do i=ilim(1,j,imw),ilim(1,j,imw)+
&                ilim(2,j,imw)-1
    r8fac=r8fac+rcder(i,jder)*rsan(j,i,1,imw)+
&             ader(i,jder)*rsan(j,i,4,imw)+
&             bder(i,jder)*rsan(j,i,3,imw)+
&             cder(i,jder)*rsan(j,i,2,imw)
   end do
   rspctcder(j,jgeo,jder)=r8fac*rintils(imw)

  end do
```

12. Computation of coefficients for the linear interpolation for spectrum, temperature perturbed spectrum and continuum derivative

For each point *i* of the compressed grid between *1* and *(nused1(imw)-1)*,

```
      do i=1,nsig-1
```

the coefficient of the linear interpolation *rm* for spectrum, temperature perturbed spectrum and continuum derivatives, is computed as follows:
first of all the variables which are independent on the value of the spectrum, temperature perturbed spectrum and continuum derivatives are computed:

ii2=igridc(i,imw)
ii3=igridc(i+1,imw)
rdc1=1.d0/dble(ii3-ii2)

The coefficient *rm(i)* is determined by the following formula:

rm(i)=rdc1(rsp(i+1)-rsp(i))*

The coefficient *rm* is computed for all the temperature perturbed spectra and all the continuum derivatives.

The coefficient *rmpert(i, jpert = igeotder(jgeo,1) → igeotder(jgeo,2))* for the temperature perturbed spectra is equal to:

*rmpert(i,jpert)= (rsppert(i+1,jpert)-rsppert(i,jpert))*rdc1*

The coefficient *rmcder(i, jder = igeocder(jgeo,1) → igeocder(jgeo,2))* for the continuum derivatives is equal to:

*rmcder(i,jder)= (rcder(i+1,jder)-rcder(i,jder))*rdc1*

13. Direct linear interpolation / convolution

For the meaning of subsequent computations, please refer to description of routine read_irregular grid_pt, par. 2.2.30.

The direct interpolation / convolution is performed using the coefficients computed in 12. and the matrices *rsan(imxi,imxsi2,4,imxmw)* and *lim(2,imxi,imxmw)* computed by routine read_irrgrid_pt.f.

For each *jsam* between *1* and *(nsam(imw))*, the value of the low resolution spectrum is computed using the following formula:

$$rspct(jsam, jgeo) = \sum_{i\lim(1,jsam,imw)}^{i\lim(1,jsam,imw)+i\lim(2,jsam,imw)-1} \left(rsp(i) \cdot rsan(jsam, i, 1, imw) + rm(i) \cdot rsan(jsam, i, 2, imw) \right)$$

At the end the low resolution spectrum is normalised:

$$rspct(jsam, jgeo) = rspct(jsam, jgeo) \text{ *rintils(imw)}$$

The same operations have to be performed also for all the temperature perturbed spectra (*rspcttpert(jsam=1→nsam(imw),jgeo,jpert = igeotder(jgeo,1) → igeotder(jgeo,2))*) and the continuum derivatives

(*rspctcder(jsam=1→nsam(imw),jgeo,jder = igeocder(jgeo,1)→igeocder(jgeo,2)*)).

```
 do jsam=1,nsam(imw)
  r8fac=0.d0
  do i=ilim(1,jsam,imw),ilim(1,jsam,imw)+
&                ilim(2,jsam,imw)-1
   r8fac=r8fac+rsp(i)*rsan(jsam,i,1,imw)+
&             rm(i)*rsan(jsam,i,2,imw)
  end do
  rspct(jsam,jgeo)=r8fac*rintils(imw)

  do jpert=igeotder(jgeo,1),igeotder(jgeo,2)
   r8fac=0.d0
   do i=ilim(1,jsam,imw),ilim(1,jsam,imw)+
&                ilim(2,jsam,imw)-1
    r8fac=r8fac+
&       rsppert(i,jpert)*rsan(jsam,i,1,imw)+
&       rmpert(i,jpert)*rsan(jsam,i,2,imw)
   end do
   rspcttpert(jsam,jgeo,jpert)=r8fac*rintils(imw)
  end do

  do jder=igeocder(jgeo,1),igeocder(jgeo,2)
   r8fac=0.d0
   do i=ilim(1,jsam,imw),ilim(1,jsam,imw)+
&                ilim(2,jsam,imw)-1
    r8fac=r8fac+rcder(i,jder)*rsan(jsam,i,1,imw)+
&             rmcder(i,jder)*rsan(jsam,i,2,imw)
   end do
   rspctcder(jsam,jgeo,jder)=r8fac*rintils(imw)
  end do

 end do
```

14. Convolution of the spectra and derivatives with the AILS function

In a call to module **conv_pt** the convolution with the AILS function *rils* is performed for the original spectrum *rsp*, for the continuum derivatives *rcder* and for the temperature perturbed spectra *rsppert*. The results are the spectra and derivatives on the general coarse wavenumber grid: *rspct, rspctcder, rspcttpert.*

15. Finalisation of the temperature perturbed spectra

The spectra which are not affected by the temperature perturbation are set to the original spectra:
For 1 ≤ jpert ≤ *igeotder(jgeo,1)*-1 and *igeotder(jgeo,2)*+1 ≤ jpert ≤ *ipar*:
rspcttpert(ksig,jgeo,jpert)=rspct(ksig,jgeo)

2.2.11.19 FOV_PT

```
FOV_PT
  |((((+FOV3_PT
       | (( -----INTCON_PT
       |(----GRAVITY *
```

Description

This module calculates the contribution of field of view to all the perturbed (*rspcttper*t) and not-perturbed (*rspct*) spectra corresponding to the observed ones, and to the derivatives of the spectrum with respect to continuum (*rspctcder*).
Besides, the module calculates the analytical derivate with respect to tangent pressure for each not-perturbed spectrum.

For some explanations of the reasons of the choices implemented in this module, refer to *T.N. on 'High Level algorithm definition and physical and mathematical optimisations', (TN-IROE-RSA9601),* par. 6.6.

Variables exchanged with external modules:

Name:	**Description:**
igeo	total number of simulated geometries
iocsim	iocsim(imxgeo,imxmw) = occupation matrix for the simulations to be performed = 0 no simulation required, = 1 simulation required without FOV = 2 simulation required with FOV
itglev	itglev(imxgeo) = number of the tangent-level for each geometry
rspct	rspct(imxi,imxgeo) = low-resolution spectrum without FOV
rspctcder	rspctcder(imxi,imxgeo,imxlmb) = convolved continuum derivative spectra for each geometry and each parameter level
rspcttpert	rspcttpert(imxi,imxgeo,imxlmb) = low resolution spectra for the perturbed temperature profiles
nsam	nsam(imxmw) = n. of sampling points in each MW (coarse grid)
imw	number corresponding to the actual microwindow
rzmod	rzmod(imxlev) = heights of levels used for the radiat. tranf. calc.
rzmodpert	rzmodpert(imxlev,imxlmb) = perturbed altitude grids after the perturbation of temp. profiles.
rpmod	rpmod(imxlev) = pressure on levels used for the radiat. transf. calc.
rtmod	rtmod(imxlev) = temperature on levels used for the radiat. transf. calc.
rbase	greater base of the trapezium that approximates the antenna pattern of FOV
rsl	half-difference between the two bases of the FOV trapezium
rlat	latitude (deg.)
igeocder	igeocder(imxgeo,2) = for each geometry the highest (x,1) and lowest (x,2) continuum derivative (in the parameter-grid) which has to be calculated
igeotder	igeotder(imxgeo,2) = for each simulated geometry the highest (x,1) and lowest (x,2) parameter for which the perturbed spectra are calculated
rspfov	rspfov(imxi,imxgeo,imxmw) = simulated spectra corresponding to the different tangent pressures and different microwindows on the frequency coarse grid: (rspct * FOV)
rptander	rptander(imxi,imxlmb) = derivatives with respect to the tangent pressure
rcderfov	rcderfov(imxi,imxgeo,imxlmb) = derivative with respect to continuum after fov convolution
rtpertfov	rtpertfov(imxi,imxgeo,imxlmb) = temperature-perturbed spectra after fov-convolution

Module structure:

1. Inizialisation of the output variables

Begin loop 1 on simulated geometries

Begin condition 1: the spectrum simulated at the actual geometry corresponds to an observation in the actual microwindow

2. Definition of tangent altitude, tangent pressure, tangent temperature of the actual spectrum and tangent altitudes of the two contiguous spectra.

3. Calculation of the contribute of F.O.V. to the spectrum, temperature perturbed spectrum, derivate of the spectrum with respect to continuum, and computation of analytical derivatives with respect to tangent pressure

4. Warning message if a particular condition is verified.

End condition 1

End loop 1

Detailed description:

1. Inizialisation of the output variable

All the elements of matrices *rspfov*, *rptander*, *rcderfov*, *rtpertfov* are set equal to 0.

Besides, the variable *iactugeo*, that counts the number of the geometries corresponding to observations in the actual MW, is set equal to 0.

The area of the trapezium that approximates FOV function is calculated:

rarea=rbase-rsl

Loop 1 over the simulated geometries

jgeo=2→ igeo -1

Geometry no.1 and no. igeo do not surely correspond to observations.

Condition 1

This condition checks the value of the iocsim matrix for the given MW and the given geometry (remember that this module is located in a loop on all the selected MWs).

If iocsim(jgeo,imw) is equal to 2, it means that the simulated spectrum we are considering corresponds to an observation, hence it will be convolved with FOV function.

If that condition is not verified, all the calculations for taking in account FOV will be skipped and the spectrumof subsequent geometry will be analysed.

```
imw: index of microwindow
mgeo: index of simulated geometry
do k=2,igeo-1
  if ( iocsim(k,imw) = 2) then
    mgeo=mgeo+1
    {  Operations 2.,  3.,  4.}
  end if
end do
```

Let's consider the case in which iocsim(jgeo,imw) = 2; then:

2. Definition of tangent altitude, tangent pressure, tangent temperature of the actual spectrum and tangent altitudes of the two contiguous spectra.

The counter of the observed geometries in the actual MW *iactugeo* is increased of 1 unit.

Determination of the tangent altitude (*rztan = rzmod(itglev(jgeo))*), pressure (*rptan = rpmod(itglev(jgeo))*) and temperature (*rttan=rtmod(itglev(jgeo))*) of the considered spectrum.
Determination of the tangent altitudes of the two contiguous spectra: the above spectrum, characterised by the geometry *jgeoup=jgeo-1*, has tangent altitude *rztanup=rzmod(itglev(jgeoup))* and the one below, characterised by the geometry *jgeodown=jgeo+1*, has tangent altitude *rztandown=rzmod(itglev(jgeodown)).*
Calculation of the distance between the tangent altitudes corresponding to geometries *jgeoup* and *jgeodown*: *rdiff=rztanup-rztandown*.

3. Calculation of the contribute of F.O.V. to the spectrum, temperature perturbed spectrum, derivate of the spectrum with respect to continuum and computation of analytical derivative with respect to tangent pressure
All these calculations are performed by module **fov3_pt (***iactugeo, imw, jgeo, rztan, rptan, rttan, jgeoup, rztanup, jgeodown, rztandown, igeocder, igeotder, rspct, rspctcder, rspcttder, nsam, rbase, rsl, rlat, rspfov, rptander, rcderfov, rtpertfov, rzmodpert, itglev***)**.
4. Warning message
If the variable *rdiff* is smaller than the greater base of the trapezium that approximate the antenna pattern of FOV, a warning message is written: 'An extrapolation has been done for spectrum no.', *jspectrum.*

2.2.11.20 FOV3_PT

Description
After the interpolation in altitude between the spectra at three contiguous tangent altitudes, this module performs the analytical convolution of the interpolated spectrum with the FOV function. This procedure is repeated for the spectra calculated for perturbed temperature profiles and the derivatives of the spectrum with respect to continuum.
Besides, the module calculates the analytical derivate with respect to tangent pressure for each not-perturbed spectrum.

Variables exchanged with external modules:

Name:	Description:
iactugeo	local counter of the geometries of the actual MW corresponding to the observations
imw	number of the actual microwindow
jgeo	actual index of simulated spectrum
rztan	tangent altitude of the spectrum of which we are calculating convolution with FOV.
rptan	tangent pressure of the spectrum of which we are calculating convolution with FOV.
rttan	tangent temperature of the spectrum of which we are calculating convolution with FOV.
jgeoup	index of the geometry above the considered one
rztanup	tangent altitude corresponding to geometry *jgeoup*
jgeodown	index of the geometry below the considered one

rztandown	tangent altitude corresponding to geometry *jgeodown*
igeocder	igeocder(imxgeo,2) = for each geometry the highest (x,1) and lowest (x,2) continuum derivative (in the parameter-grid) which has to be calculated
igeotder	igeotder(imxgeo,2) = for each simulated geometry the highest (x,1) and lowest (x,2) parameter for which the perturbed spectra are calculated
rspct	rspct(imxi,imxgeo) = low-resolution spectrum (rsp * ILS)
rspctcder	rspctcder(imxi,imxgeo,imxlmb) = the convolved continuum derivative spectra for each geometry and each parameter level
rspcttpert	rspcttpert(imxi,imxgeo,imxlmb) = low resolution spectra for the perturbed temperature profiles
nsam	nsam(imxmw) = no. of sampling points in each MW (coarse grid)
rbase	greater base of trapezium-shape that approximates Field of View pattern
rsl	half-difference between the bases of the trapezium (1/rsl gives the slope)
rlat	latitude of the actual limb-scan (deg.)
rspfov	rspfov(imxi,imxgeo,imxmw) = simulated spectra corresponding to the different tangent pressures and different microwindows on the frequency coarse grid: (rspct * FOV)
rptander	rptander(imxi,imxlmb) = derivatives with respect to the tangent pressure
rcderfov	rcderfov(imxi,imxgeo,imxlmb) = derivative with respect to continuum after fov convolution
rtpertfov	rtpertfov(imxi,imxgeo,imxlmb) = temperature-perturbed spectra after fov-convolution
rzmodpert	rzmodpert(imxlev,imxlmb) = perturbed altitude grids after the perturbation of temp. profiles.
itglev	itglev(imxgeo) = number of the tangent-level for each geometry

Module structure:

1. Definition of the vector *rxa* containing the tangent heights of the spectra used for the interpolation and calculation of some geometrical quantities useful for next convolutions

Begin loop 1 on the frequencies of the actual MW

2. Definition of the vector *rya* containing the values of the spectra corresponding to *rxa* for the actual frequency.
3. Analytical convolution and normalisation
4. Storing of the results
5. Calculation of analytical derivative of the spectrum with respect to tangent pressure

Begin loop 2 on the continuum parameters that affect the spectrum corresponding to geometry jgeo

6. Definition of the vector *rya* containing the values of the derivatives with respect to continuum corresponding to *rxa* for the actual frequency.
7. Analytical convolution and normalisation
8. Storing of the results

End loop 2

Begin loop 3 on the temperature parameters that affect the spectrum corresponding to geometry jgeo

9. Definition of the vector *rxa1* containing the tangent heights of the spectra used for the interpolation in the case of the particular perturbed profile

10. Definition of the vector *rya* containing the values of the perturbed spectra corresponding to *rxa1* for the actual frequency.
11. Analytical convolution and normalisation
12. Storing of the results

End loop 3

End loop 1

Detailed description

1. Definition of the vector *rxa* containing the tangent heights of the spectra used for the interpolation and calculation of some geometrical quantities useful for next convolutions

The vector *rxa* is filled with the tangent altitudes of the spectra considered for the interpolation, starting from the lowest tangent altitude.

The FOV function is represented by a trapezium-shape function whose greater base is *rbase* and the half-difference between the two bases is *rsl*.

Begin loop 1 on the frequencies of the actual MW
jsig=1→ nsam(imw) ,
nsam(imw) is total number of sampling points in MW *imw*.

2. Definition of the vector *rya* containing the values of the spectra corresponding to the tangent altitudes contained in *rxa* for the actual frequency.

The vector *rya* is filled with the values of the three considered spectra at the frequency *jsig*:
rya(1)=rspct(jsig,jgeodown), rya(2)=rspct(jsig,jgeo), rya(3)=rspct(jsig,jgeoup).

3. Analytical convolution and normalisation

These operations are performed by module **intcon_pt** (*rxa, rya, 3, rbase, rsl, rarea, rztan, rcof, rp*)

4. Storing of the results

The result of this procedure *rp* is stored:
rspfov(jsig, iactugeo, imw)= rp

5. Calculation of analytical derivative of the spectrum with respect to tangent pressure

First the variable *rconst* is calculated:
*rconst=gravity(rztan,rlat) * rmovr,*
using the function **gravity** that calculates gravity acceleration.
Then the analytical derivatives with respect to tangent pressure is calculated using the following expression:

$$rptander(jsig, iactugeo) = -\frac{rttan}{rconst \cdot rptan} \cdot \left(rcof(2) + 2 \cdot rcof(3) \cdot rztan\right).$$

Begin loop 2 on the continuum parameters that affect the spectrum corresponding to geometry jgeo
jpar = igeocder(jgeo,1) → igeocder(jgeo,2)
The inputs of this section are, for each observed geometry, all the derivatives of the corresponding spectrum with respect to the continuum parameters that affect it:
rspctcder(jsig,jgeo,jpar);

igeocder(jgeo,1) and igeocder(jgeo,2) represent respectively the highest and lowest parameter level, whose value of continuum affects the spectrum corresponding to the geometry jgeo.
All these quantities have to be convolved with the FOV function, so the operations from 2. to 4. are repeated for the matrix rspctcder(jsig,jgeo,jpar).

6. Definition of the vector *rya* containing the values of the derivatives with respect to continuum corresponding to *rxa* for the actual frequency.
The vector *rya* is filled with the values of the three considered continuum derivatives at the frequency *jsig*:
rya(1)=rspctcder(jsig,jgeodown,jpar),
rya(2)=rspctcder(jsig,jgeo,jpar),
rya(3)=rspctcder(jsig,jgeoup,jpar).

7. Analytical convolution and normalisation
See 3.

8. Storing of the results
The result of this procedure *rp* is stored:
rcderfov(jsig, iactugeo, jpar)= rp

Begin loop 3 on the temperature parameters that affect the spectrum corresponding to geometry jgeo
jpar=igeotder(jgeo,1) → igeotder(jgeo,2)
The inputs of this section are, at the considered geometry jgeo, all the spectra obtained perturbating, one to a time, the temperature on all the parameter levels (these spectra are used for performing the numerical derivatives with respect to tangent temperature):
rspcttder(jsig,jgeo, igeotder(jgeo,1)→igeotder(jgeo,2));
igeotder(jgeo,1) and igeotder(jgeo,2) represent respectively the highest and lowest parameter level, whose perturbation affects the spectrum corresponding to the geometry jgeo.
All these quantities have to be convolved with the FOV function, so the operations from 2. to 4. are repeated for the matrix rspcttder(jsig,jgeo,jpar).

9. Definition of the vector *rxa1* containing the tangent heights of the spectra used for the interpolation for the particular perturbed profile
When the temperature is changed on one parameter level, also the altitudes of some of these levels are perturbed. Therefore, in this case, the vector containing the tangent altitudes of the spectra considered for the interpolation has to be redefined for each parameter *jpar* we consider.
The new vector *rxa1* is filled with the perturbed tangent altitudes of the spectra considered for the interpolation, starting from the lowest tangent altitude.
rxa1(1)=rzmodpert(itglev(jgeodown),jpar)
rxa1(2)=rzmodpert(itglev(jgeo),jpar)
rxa1(3)=rzmodpert(itglev(jgeoup),jpar)

10. Definition of the vector *rya* containing the values of the perturbed spectra corresponding to *rxa* for the actual frequency.
The vector *rya* is filled with the values of the three considered perturbed spectra at the frequency *jsig*:
rya(1)=rspcttder(jsig,jgeodown,jpar),
rya(2)=rspcttder(jsig,jgeo,jpar),

rya(3)=rspcttder(jsig,jgeoup,jpar).

11. Analytical convolution and normalisation
See 3.

12. Storing of the results
The result of this procedure *rp* is stored:
rtpertfov(jsig, iactugeo, imw)= rp

2.2.11.21 POLCOE_PT

Description
This module is taken from 'Numerical Recipes in FORTRAN' and calculates the coefficients of the interpolating polynomium which crosses a given number of points.

Variables exchanged with external modules

Name:	Description:
rx	rx(n) vector containing the x values of the tabulated points
ry	ry(n) vector containing the y values of the tabulated points
n	number of tabulated points
rcof	rcof(n) returned coefficients, such that: $y_i = \sum_j rcof(j) \cdot x_i^{j-1}$

Detailed description:
see 'Numerical Recipes in FORTRAN', pag.114.

2.2.11.22 INTCON_PT

|| INTCON_PT
    || POLCOE_PT

Description
This module calculates the coefficients for the interpolation between the points contained in the vectors *rxa* and *rya* and then computes the analytical convolution with FOV.

Variables exchanged with external modules:

Name	Description
rxa	rxa(3) vector containing the x values of the tabulated points
rya	rya(3) vector containing the y values of the tabulated points
n	number of tabulated points
rbase	greater base of the trapezium
rsl	half-difference between the two bases of the trapezium
rarea	area of the trapezium, used for normalisation
rztan	altitude in correspondence of which spectrum with FOV is calculated
rcof	rcof(3) interpolating polynomial coefficients
rp	returned value of the convolution and normalisation

Module structure:

Detailed description:
1. Calculation of the coefficients for the interpolation
2. Calculation of analytical convolution and normalisation

1. Calculation of the coefficients for the interpolation
The coefficients of the interpolation, contained in the vector *rcof*, are calculated by the module **polcoe(***rxa,rya,n,rcof***)**, so that the interpolated spectrum at altitude *r* and frequency *jsig* results given by $\sum_{i=1,n} rcof(i) \cdot r^{n-1}$.

2. Analytical convolution and normalisation
The result of the analytical convolution is given by the following expression:

$$rp = \frac{\left[\begin{array}{l} rcof(1) \cdot rarea + rcof(2) \cdot rarea \cdot rzc + rcof(3) \cdot rzc^2 \cdot rarea + \\ + rcof(3) \cdot \left(\frac{rbase^3}{12} - \frac{rbase^2 \cdot rsl}{4} + \frac{rsl^2 \cdot rbase}{3} - \frac{rsl^3}{6} \right) \end{array} \right]}{rarea}$$

2.2.11.23 TEMDER_PT

Description
Determination of the temperature derivatives from the temperature-perturbed spectra.

Variables exchanged with external modules:

Name:	Description:
rtpertfov	temperature-perturbed spectra after fov-convolution (input) temperature-derivatives (output) 1st index: wavenumber coarse grid point 2nd index: fov-simulated geometry 3rd index: parameter level (which was perturbed and for which the derivatives are made)
rspfov	'original' spectra after fov-convolution
imw	number of the actual microwindow
igeo	number of simulated geometries
iocsim	occupation matrix for the simulations to performed = 0 no simulation required, = 1 simulation required without FOV = 2 simulation required with FOV
ipar	number of parameter-levels
rdt	temperature perturbation [K]
nsam	number of sampling points in each mw (general coarse grid)
igeotder	for each simulated geometry the highest (x,1) and lowest (x,2) parameter for which the perturbed spectra are calculated

Module structure

Begin loop 1 over all geometries for which fov-simulations are performed
  Begin loop 2 over all parameter levels valid for the actual geometry
    1. Calculation of the temperature derivatives
  end loop 2
end loop 1

Detailed description

loop 1 over all geometries for which fov-simulations are performed:
jgeo=1→igeo
if [*iocsim(jgeo,imw)* = 2]: $jlimb = jlimb + 1$

Begin loop 2 over all parameter levels valid for the actual geometry:
kpar=igeotder(jgeo,1)→igeotder(jgeo,2)

1. Calculation of the temperature derivatives:
The temperature derivatives are simply calculated numerically:
For $1 \le l \le nsam(imw)$:

$$rtpertfov(l, jlimb, kpar) = \frac{rtpertfov(l, jlimb, kpar) - rspfov(l, jlimb, imw)}{rdt}$$

2.2.11.24 JACSETMW_PT

Description
For the actual microwindow the temperature- and tangent pressure- derivatives are written into the jacobian matrix.
The derivatives of the specta with respect to the fitted continuum parameters are calculated by multiplication of the derivatives 'rcderfov' (with respect to the parameter-levels) with the derivatives 'rjaccon' of the continuum on the parameter-levels with respect to the fitted continuum parameters

Variables exchanged with external modules:

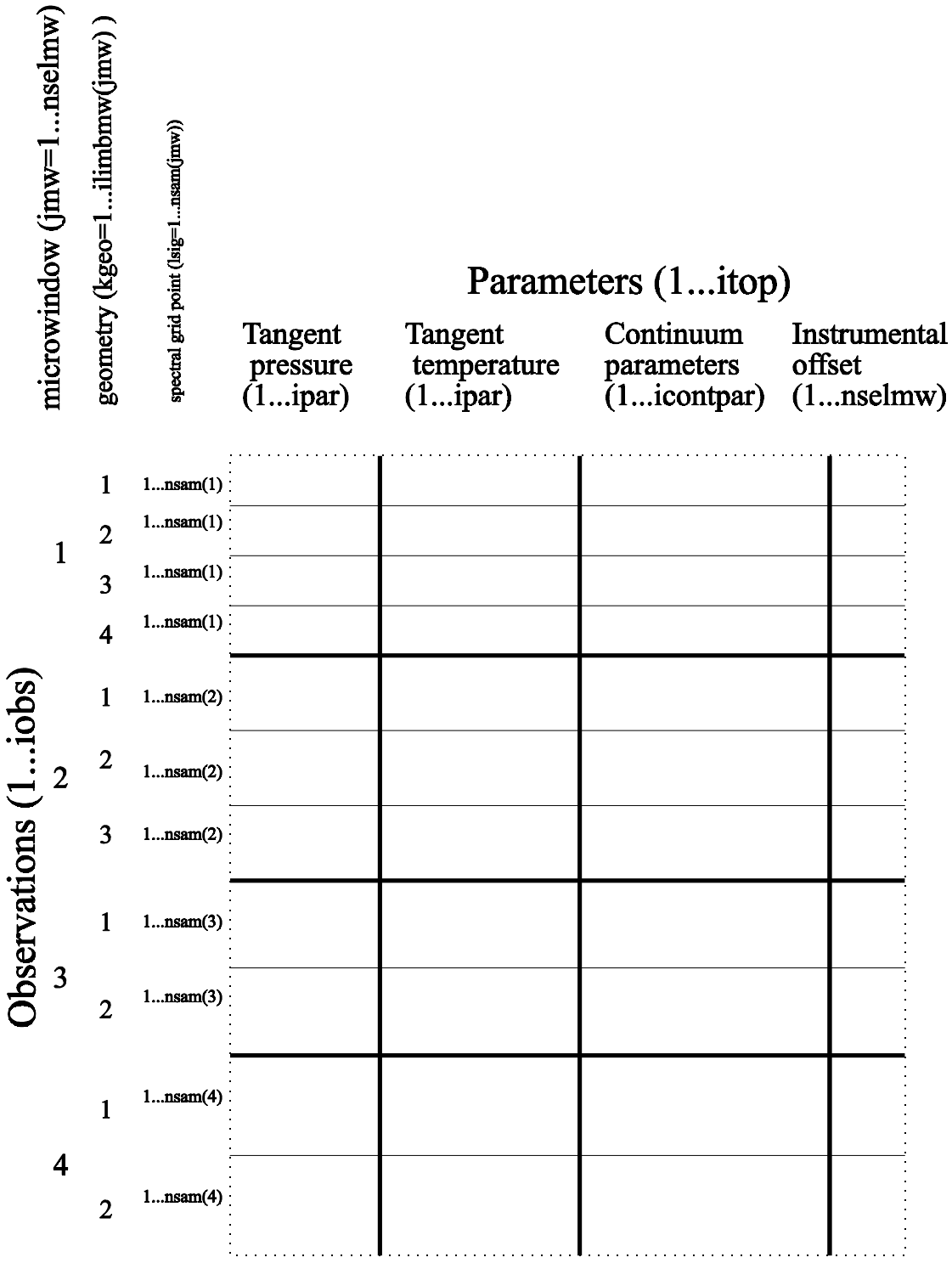
Name:	Description:
rtpertfov	temperature-derivative spectra
rptander	derivatives with respect to the tangent pressure
imw	number of the actual microwindow
ilimbmw	number of valid measured geometries per microwindow
ipar	number of parameter-levels
nsam	number of sampling points in each Mw (general coarse grid)
lokku	occupation matrix used for the selection of operational Mw's for each observation geometry
nucl	nucl+1 = upper parameter level for continuum fit
ilimb	number of measured geometries
rcderfov	derivate with respect to continuum after fov convolution
icontpar	total number of continuum parameters to be fitted
rjaccon	jacobian matrix for the derivative of the continuum parameter-level values with respect to the continuum parameters
irowmw	the row of the Jacobian matrix where the actual mirowindow starts
rjacob	Jacobian Matrix 1st index: observations 2nd index: parameters
lparbase	lparbase(imxpro) = logical vector which identifies the altitudes where the T profile is fitted, among the altitudes rzbase.
ibase	ibase = number of base-levels

Module structure

1. Writing the tangent pressure derivatives into the Jacobian matrix
2. Writing the temperature derivatives into the Jacobian matrix
3. Multiplication of the 'local' continuum derivatives by the continuum jacobian matrix and writing the result into the Jacobian matrix
4. Writing the instrumental offset derivatives into the Jacobian matrix

Detailed description

Before describing the single steps of the code we give an overview of the structure of the Jacobian matrix which is the matrix of the derivatives of all observations with respect to all parameters:



1. Writing the tangent pressure derivatives into the Jacobian matrix:

The derivatives with respect to the tangent pressure are only different from 0 if the parameter level is identical to the geometry.

We begin with the first row of the Jacobian matrix for the actual Mw: *lrow=irowmw(imw)* and the counting index for the geometry of the derivative spectra *mgeo* is set to 0.

The following happens inside a loop over the parameter levels: $1 \leq jpar \leq ipar$:

lcol=jpar

if [*lokku(jpar,imw)*] :
The geometry is counted up: *mgeo=mgeo*+1
and in a loop over the frequency course grid $1 \leq ksig \leq nsam(imw)$ the Jacobian matrix is set up:
rjacob(lrow,lcol)=rptander(ksig,mgeo)
lrow=lrow+1

2. Writing the temperature derivatives into the Jacobian matrix:
For all parameter levels $1 \leq jpar \leq ipar$:

The actual column of the parameters for the temperature derivatives is:
lcol=ipar+jpar
and the starting row is:
lrow=irowmw(imw)
Then for all geometries $1 \leq kgeo \leq ilimbmw(imw)$ and all frequency grid points $1 \leq lsig \leq nsam(imw)$:
rjacob(lrow,lcol)=rtpertfov(lsig,kgeo,jpar)
lrow=lrow+1

3. Multiplication of the 'local' continuum derivatives with the continuum jacobian matrix and writing the result into the Jacobian matrix:

Begin loop I on continuum parameters: *jpar=1,...,icontpar*
*lcol=2*ipar+jpar*
lrow=irowmw(imw)
Begin loop II on the goeometries of the current MW: *kgeo=1,..,ilimbmw(imw)*
Begin loop III on frequency: *lsig=1, ..., nsam(imw)*
m3=0.
r1=0.
Begin loop IV on the 'base' levels: *m1=1, ..., ibase*
if (*lparbase(m1)*)then
m3 = m3 + 1
*m2=(imw-1)*ibase+m1*
*r1=r1+rcderfov(lsig,kgeo,m3)*rjaccon(m2,jpar)*
end if
End loop IV on the 'base' levels
rjacob(lrow,lcol)=r1
lrow=lrow+1
End loop III on frequency
End loop II on geometries of the current MW
End loop I on continuum parameters

4. Writing the instrumental offset derivatives into the Jacobian matrix:
The derivatives with respect to the instrumental continuum are equal to 1.
The column where the derivatives are written for the actual Mw is:
lcol=2·ipar+icontpar+imw
The starting row is: *lrow=irowmw(imw)*

Then, for all geometries $1 \leq kgeo \leq ilimbmw(imw)$ and all frequency grid points $1 \leq lsig \leq nsam(imw)$:

rjacob(lrow,lcol)=1

lrow=lrow+1

2.2.11.25 ADDOFF_PT

Description

This module adds the instrumental offset to the spectra

Variables exchanged with external modules

Name	Description
rspfov	rspfov(imxi,imxgeo,imxmw) = spectra corresponding to the different tangent altitudes on the coarse frequency grid
imw	index of the actual microwindow
ilimbmw	ilimbmw(imxmw) = number of valid measured geometries per microwindow
nsam	nsam(imxmw) = number of sampling points in each microwindow (coarse grid)
roffs	roffs(imxmw) = instrumental continuum, for each microwindow

Detailed description

For each valid limb view of the considered microwindow (*jlimb=1* $\rightarrow$ *ilimbmw(imw)*)

for each spectral point of the given microwindow (*ksig=1* $\rightarrow$ *nsam(imw)*)

the instrumental offset *roffs(imw)* is added to the spectral point:

rspfov(ksig,jlimb,imw) = rspfov(ksig,jlimb,imw) + roff(imw)

Note that the result of this operation is stored in the original vector *rspfov.*

2.2.12 ABCALC_PT

Description

This module calculates the matrices $\mathbf{A} = \mathbf{K}^T\mathbf{S}^{-1}\mathbf{K} + \mathbf{K}_1^T(\mathbf{V}^z)^{-1}\mathbf{K}_1$, $\mathbf{B}^T = \left(\mathbf{K}^T\mathbf{S}^{-1}\right)^T$ and $\mathbf{B}_1 = \mathbf{K}_1^T(\mathbf{S}^z)^{-1}$ (see AD6 for the definition of these matrices).

Variables exchanged with external modules

Name	Description
rjacob	The **K** matrix (used dimensions (itop*iobs))
rvcmobinv	Inverse of the VCM of the observations, not divided by the square of the noise

ra	**A** matrix of [AD6]
rbt	**Transpose of the B** matrix of [AD6]
iobs	total N. of observations
itop	total n. of unknown parameters
nselmw	Number of selected microwindows (used to build S^{-1} matrix)
ilimbmw	ilimbmw(imxmw) = n. of sweeps at which the current MW is used
nsam	nsam(imxmw) = n. of sampling points in each MW (coarse grid)
rnoise	Noise used to build the S^{-1} matrix
ilimb	ilimb = N. of considered sweeps
lokku	lokku(imxgeo,imxmw) = MW occupation matrix
rjaclos	jacobian matrix of engineering pointings (matrix $\mathbf{K}_1$ in [AD6])
ipar	n. of retrieved points in the T (VMR) profile
rinvclos	rinvclos(imxlmb,imxlmb) = inverse of the VCM of the engineering pointings
lextinf1	logical switch for enabling the use of LOS engineering info
rblos	rblos(imxtop,imxlmb) = matrix $\mathbf{B_1} = \mathbf{K_1^T (V^z)^{-1}}$ defined in the alorithms document [AD6]
lifend	switch for using LOS info only at the end of p,T retrieval
liflc	logical variable indicating whether this is the last call to this module

Module structure

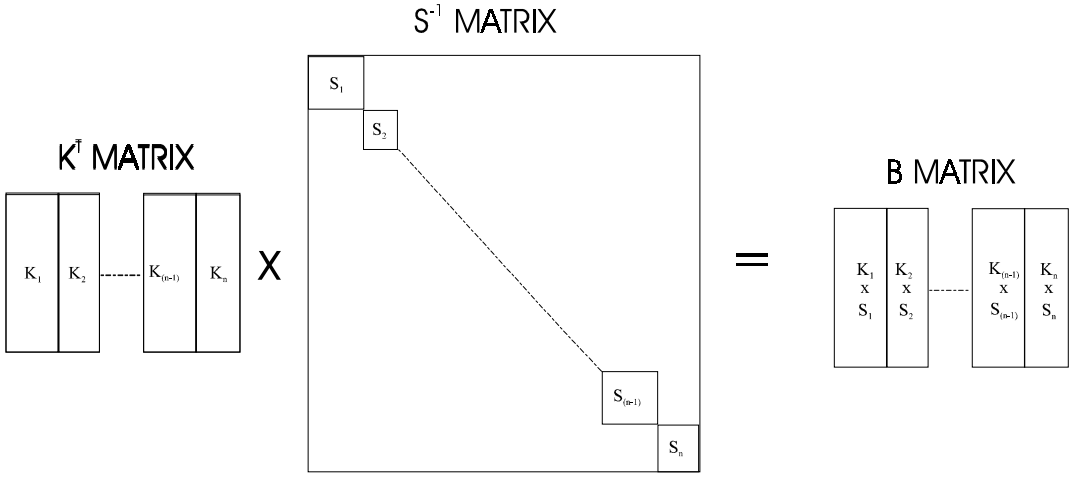
This module computes **A**, $\mathbf{B^t}$ (equal to transpose of **B**) and $\mathbf{B}_1$ matrices. The $\mathbf{S^{-1}}$ matrix is a block diagonal matrix. The square sub-block referring to the *j-th* MW has dimension equal to *nsam(j).* The number *'Nblocks'* of blocks of $\mathbf{S^{-1}}$ is given by the summation on all the microwindows (*j=1..nselmw*) of *ilimbmw(j).*

- Step 1

The rjacob matrix (*itop,iobs*) is divided in blocks. The block referring to a considered geometry of microwindow *j* has dimensions (*itop*nsam(j)*).

- Step 2

Each block is multiplied by the corresponding *j-th* square block (dimension *nsam(j)· nsam(j*) of *rvcmobinv* matrix; the result is copied in the corresponding block of matrix B.



- Step 3

Matrix **B** is multiplied by **K** matrix to get **A** matrix.

- Step 4

If LOS info is to be used it calculates matrix $\mathbf{B}_1$ and adds the contribution $\mathbf{K}_1^T(\mathbf{V}^z)^{-1}\mathbf{K}_1$ to matrix **A.**

Detailed description

```
* Standard calculation (A=K'*S^-1*K):
     icount=0
          do j=1,nselmw
      do k=1,ilimb
            if (lokku(k,j))then
          rnoise2 = SNGL(1.0D0/rnoise(j,k)**2)
* Multiplication rjacob * rvcmobinv (Kt * V**-1):
            do l=1, itop
             do m=1, nsam(j)
              rtemp1=rjacob(1+icount,l)*rvcmobinv(1,m,j)
              do m1=2,nsam(j)
                rtemp1=rtemp1+
    &                 rjacob(m1+icount,l)*rvcmobinv(m1,m,j)
              end do
               rbt(icount+m,l)=rtemp1*rnoise2
             end do
            end do

                icount = icount + nsam(j)
                end if
              end do     ! end loop LS (k=1,...ilimb)
            end do       ! end loop MWs (j=1,...nselmw)

* Multiplication of rbt *rjacob ---> ra
     do k=1,itop
      do j=1,k
       rtemp1=rbt(1,j)*rjacob(1,k)
       do l=2,iobs
        rtemp1=rtemp1+rbt(l,j)*rjacob(l,k)
       end do
       r1=dble(rtemp1)
       ra(j,k)=r1
       ra(k,j)=r1
      end do
     end do



* if engineering LOS info has to be used:
     if (lextinf1.or.(lifend.and.liflc)) then

* we compute rblos = rjaclos(transposed) * rinvclos:
     do j=1,ilimb-1
```

```
        do i=1,ilimb+ipar
          r1 = rjaclos(1,i)*rinvclos(1,j)
          do k=2,ilimb-1
            r1 = r1 + rjaclos(k,i)*rinvclos(k,j)
          end do
          rblos(i,j) = r1
        end do
      end do
* we compute then rblos * rjaclos and at the same time we add it to ra:
      do j=1,ilimb+ipar
        do i=1,ilimb+ipar
          r1 = rblos(i,1)*rjaclos(1,j)
          do k=2,ilimb-1
            r1 = r1 + rblos(i,k)*rjaclos(k,j)
          end do
          ra(i,j) = ra(i,j) + r1
        end do
      end do

      end if

      end
```

2.2.13 DIFCHI_PT

DIFCHI_PT]
|-----CHISQ_PT *

Description
It calculates the χ^2 function that has to be minimised in retrieval procedure.
After the computation of the vector of the residuals (*rnres*) from the observed (*robs*) and simulated (*rspfov*) spectra, it performes the matrix product between the transpose of *rnres* and *rnres*, weighted by the inverse of the variance covariance matrix of the observations (*rvcmobinv).*

Variables exchanged with external modules:

Name	Description
iobs	total number of observations
itop	total number of parameters to be fitted
robs	robs(imxi,imxgeo,imxmw)): observed spectra corresponding to the different tangent altitudes and different microwindows (on the coarse frequency grid)
rspfov	rspfov(imxi,imxgeo,imxmw): simulated spectra corresponding to the different tangent pressures and different microwindows on the frequency coarse grid: (rspct * FOV)
rvcmobinv **real*4**	rvcmobinv(imxi,imxi): elementary block of the inverse of the variance covariance matrix of the observations associated to the wider microwindow.
rnoise	rnoise(imxmw,imxgeo):NESR dependent on geometry and microwindow
nsam	nsam(imxmw): no. of sampling points in each MW (coarse grid)
nselmw	total number of selected microwindows for the retrieval
ilimb	number of measured geometries
lokku	lokku(imxgeo,imxmw) occupation matrix used for the selection of operational MW's for each observation geometry
ilimbmw	ilimbmw(imxmw): number of valid measured geometries per microwindow number of 2 in each column of iocsim)
iterg	iterg = index of the actual iteration
rnres	rnres(imxobs) : vector of the differences between the observated spectraand the calculated ones;first all the geometries of the first microwindow starting from the first geometry, then all the other microwindows
rchisq	rchisq(0:imxite): total chi-square for each iteration
rchisqp	rchisqp(imxlmb,imxmw) chi-square for each observation geometry and each microwindow
rztang	rztang(imxgeo) = actual values of the tangent altitudes of the considered sweeps
rdzeng	rdzeng(imxlmb) = engineering differences between tangent altitudes
lextinf1	lextinf1 = switch for enabling the use engineering LOS info
rnreslos	rnseslos(imxlmb) = residuals of LOS info
rinvclos	rinvclos(imxlmb,imxlmb) = inverse of the VC matrix of LOS data

Module structure

1. Calculation of the vector of the residuals
2. Control on the correctness of the computations
3. if LOS info is to be used: calculation of the LOS residuals
4. Definition of *lpart*
5. Calculation of chi-square
6. Storage of chi-square.

Detailed description

```
* Calculation of the residuals vector:
      jobs=0
      do imw=1,nselmw
        kgeo1=0
        do kgeo=1,ilimb
          if (lokku(kgeo,imw)) then
            kgeo1=kgeo1+1
            do jsig=1,nsam(imw)
              jobs=jobs+1
              rnres(jobs) = robs(jsig,kgeo,imw) - rspfov(jsig,kgeo1,imw)
            end do
          end if
        end do
      end do
* Internal consistency check (the program never stops here if there are no bugs):
      if (iobs.ne.jobs)stop 'program stopped in difchi'
* if LOS info is to be taken into account we compute the vector ‘rnreslos’ of the LOS residuals:
      if (lextinf1) then
        do jobs=1,ilimb-1
          rnreslos(jobs) = rdzeng(jobs) - (rztang(jobs+1) - rztang(jobs))
        end do
      end if
* Since also the partial chi-square has to be computed we set lpart = TRUE in order to ask ‘chisq’
also
* for the computation of the partial chi-square.
      lpart=.true.
* Calculation of chi-square by using chisq_pt module
      call chisq_pt(rnres, iobs, itop, nselmw, ilimbmw, nsam, rnoise, rvcmobinv, lpart, rchi, rchisqp,
ilimb,
                    lokku, rnreslos, rinvclos, lextinf1)
* Storage of the computed chi-square
      rchisq(iterg)=rchi
      end
```

2.2.13.1 CHISQ_PT

Description

It calculates χ^2 value, performing the matrix product between the transpose of *rnres* and *rnres*, weighted by the inverse of the variance covariance matrix of the observations (*rvcmobinv)*. If this subroutine is called with 'lpart' = TRUE, it calculates also the partial chi-square related to the differnt MWs and sweeps.

Variables exchanged with external modules

Name	Description
rnres	rnres(imxobs): vector of the differences between the observed spectra and the calculated ones; first all the geometries of the first microwindow starting from the first geometry, then all the other microwindows
iobs	total number of observations
itop	total number of parameters to be fitted
nselmw	total number of selected microwindows for the retrieval
ilimbmw	ilimbmw(imxmw): number of valid measured geometries per microwindow (number of 2 in each column of iocsim)
nsam	nsam(imxmw): n. of sampling points in each MW (coarse grid)
rnoise	rnoise(imxmw,imxgeo): NESR dependent on geometry and microwindow
rvcmobinv **real*4**	rvcmobinv(imxi,imxi): elementary block of the inverse of the variance covariance matrix of the observations related to the widest microwindow.
lpart	switch for enabling the storage of the partial chi-square
rchi	returned value of the χ^2 function
rchisqp	rchisqp(imxlmb,imxmw) chi-square for each observation geometry and each microwindow temperature profiles
ilimb	number of measured sweeps
lokku	lokku(imxgeo,imxmw): MW occupation matrix used for the selection of operational MW's for each observation geometry.
rnreslos	rnreslos(imxlmb) = residuals of LOS engineering data
rinvclos	rinvclos(imxlmmb,imxlmb) = inverse of VC matrix of LOS data
lextinf1	switch for enablong the use of engineering LOS data

Module structure:

1. Calculation of the matrix product: $(rnres)^T \cdot (S)^{-1} \cdot (rnres)$
2. Storage of partial chi-square
3. If the LOS info is to be used the contribution of LOS residuals to the chi-square is computed
4. Calculation of total reduced chi-square

Detailed description:

* Calculation of the n. of degrees of freedom 'ifrede' of the problem: observations - parameters
  ifrede = iobs - itop

```
* Some initialisations:
      rchi=0.      ! initialisation of chi-square
      inres=0      ! index of vector rnres (vector of the residuals)
* calculation of the chi-square:
      do 10 imw=1,nselmw
       kgeo1=0
       do 20 kgeo=1,ilimb
         if (lokku(kgeo,imw))then
           kgeo1=kgeo1+1
           rchi1=0.
           do 30 jsig=1,nsam(imw)
             rpart(jsig)=0.
             do 40 jsig1=1,nsam(imw)
               rpart(jsig) = rpart(jsig) + rnres(jsig1+inres) * rvcmobinv(jsig1,jsig,imw)  ! transpose of
nres * rvcmobinv
 40          continue
             rchi1=rchi1+rnres(jsig+inres)*rpart(jsig)
 30        continue
           inres=inres+nsam(imw)
           rchi1=rchi1/(rnoise(imw,kgeo)*rnoise(imw,kgeo))
           if(lpart)rchisqp(kgeo1,imw)=rchi1
           rchi=rchi+rchi1
         endif
20     continue
10   continue

* if LOS info is to be used, we have to add to the above computed chi-square also the contribution
of the pointings,
* i.e. (nreslos)^T * rinvclos * nreslos
      if (lextinf1) then
       do i=1,ilimb-1
        r1 = 0.0d0
        do j=1,ilimb-1
          r1 = r1 + rinvclos(i,j) * rnreslos(j)
        end do
        rchi = rchi + r1 * rnreslos(i)
       end do
* if LOS info is used, the n. of degrees of freedom is updated as well:
      ifrede = ifrede + ilimb - 1
      end if

*     Calculation of the total reduced chi-square:
       rchi = rchi / ifrede

      end
```

2.2.14 AMODIF_PT

Description
Multiplication of the diagonal elements of the matrix *ra* by (1+*rlambda*).

Variables exchanged with external modules:

Name	Description
ra	matrix defined as (transpose of rjacob) * rvcmobinv * rjacob (as the output the diagonal elements are multiplied with 1+*rlambda*
rlambda	Marquardt damping factor
itop	total number of parameters to be fitted
ipar	ipar = number of parameter-levels (i.e. N. of elements of rzpar vector)
icontpar	n. of fitted continuum parameters

```
      do 10 j=1,itop
      if(j.gt.2*ipar.and.j.le.2*ipar+icontpar) then
        ra(j,j)=ra(j,j)*(1+rlambda*100.0)
      else
        ra(j,j)=ra(j,j)*(1+rlambda)
      end if
10    continue
     end
```

Module structure
1. Multiplication of the diagonal elements of matrix *ra* by 1+*rlambda*. A personalised damping factor is used for the elements which correspond to continuum parameters.

Detailed description
1. Multiplication of the diagonal elements by 1+*rlambda*:

For $1 \le j \le itop$:

if $j > 2 * ipar$ AND $j \le 2 * ipar + icontpar$ then
$ra(j, j) = ra(j, j) \cdot (1 + rlambda * 100)$
else
$ra(j, j) = ra(j, j) \cdot (1 + rlambda)$
end if

2.2.15 NEWPAREST_PT

NEWPAREST_PT]
|((((+CHISQ_PT *

Description
Calculates the new estimate of the vector of the unknown parameters *rxpar* and, if *iterm*=0, calculates the χ^2 in the linear approximation as well.

Variables exchanged with external modules:

Name	Description
rainv	matrix inverse of *ra*
rbt	matrix defined as transpose((transpose of *rjacob*) * *rvcmobinv)*
rnres	vector of the differences between the observed spectra and the calculated ones
rxparold	vector of the fitted parameters at the previous iteration
itop	total number of parameters to be fitted
iobs	total number of observations to be fitted
iterm	micro - iteration index (Marquardt)
rjacob	Jacobian Matrix
rxpar	vector of the fitted parameters
rlinchisq	χ^2 calculated in the linear approximation
rvcmobinv (real*4)	elementary block of inverse of the variance covariance matrix of the observations associated to the widest microwindow
rnoise	NESR dependent on geometry and microwindow
nsam	number of sampling points in each MW (general coarse grid)
nselmw	total number of selected microwindows for the retrieval
ilimbmw	number of valid measured geometries per microwindow (total number of '2's in each column of *iocsim*)
ilimb	number of measured geometries (sweeps)
lokku	occupation matrix used for the selection of operational MW's for each observation geometry
rblos	rblos(imxtop,imxlmb) = matrix defined as $rjaclos^T$ * *rinvclos*
rnreslos	rnreslos(imxlmb) = residuals of LOS information
lextinf1	switch for enabling the use of engineering LOS engineering data
ipar	number of fitted points in the T profile
rjaclos	rjaclos(imxlmb,imxtop) = jacobian matrix of LOS data
rinvclos	rinvclos(imxlmb,imxlmb) = inverse of the VC matrix of LOS data
lifend	switch fo enabling the use of LOS data only at the last iteration
liflc	switch telling whether this is the last call to this module

Module structure
1. Set *rxparold = rxpar*
2. Calculate the correction for the parameters
3. If the routine is called during a macro-iteration, the linear χ^2 is computed

4. Calculation of the new parameters

Detailed description

```
    lextbkp = lextinf1                    ! makes the backup of the switch lextinf1
    lextinf1 = lextinf1.or.(lifend.and.liflc)        ! new value of lextinf1

* Makes the backup of the parameters vector:
    do jpar=1,itop
      rxparold(jpar) = rxpar(jpar)
    end do

* Initialisation to 0 of vectors r2v(imxtop) and rxpar(imxtop)

do 15 k=1,itop
       r2v(k)=0.d0
       rxpar(k)=0.d0
15    continue


* calculates the correction parameter vector: y=(A^-1)Bn (rxpar is overwritten by this!!)

do 20 k=1,itop
      do 30 l=1,iobs
         r2v(k)=r2v(k)+rbt(l,k)*rnres(l)
30     continue
* if LOS information is to be used we add to r2 also the contribution from rblos * rnreslos:
      if (lextinf1.and.k.le.ilimb+ipar) then
         do 35 i=1,ilimb-1
           r2v(k)=r2v(k)+rblos(k,i)*rnreslos(i)
35         continue
      end if
20    continue

do 40 k=1,itop
       do 45 jpar=1,itop
         rxpar(jpar)=rxpar(jpar)+rainv(jpar,k)*r2v(k)
45      continue
40    continue


* if iterm=0 (macro-iteration) it calculates the 'linear difference vector' of the observations: n{lin} =
n - K y
* i.e. rnreslin = rnres - rjacob * rxpar
    if (iterm.eq.0) then                  ! begin condition on macro-iteration
      do 50 jobs=1,iobs
        r1=0.
        do 60 kpar=1,itop
          r1=r1+rjacob(jobs,kpar)*rxpar(kpar)
```

```
60       continue
        rnreslin(jobs)=rnres(jobs)-r1
50      continue

* if the LOS info is to be used, also the linear residuals of the pointings are computed:
       if (lextinf1) then ! begin condition if LOS info is used
        do jobs=1,ilimb-1
         r1=0.
         do kpar=1,ilimb+ipar
           r1 = r1 + rjaclos(jobs,kpar) * rxpar(kpar)
         end do
         rnresloslin(jobs) = rnreslos(jobs) - r1
        end do
       end if              ! end condition if LOS info is used

* calculates the linear chi square; rchisqp is not calculated new for the linear chi square:
       lpart=.false.
       call chisq_pt(rnreslin, iobs, itop, nselmw, ilimbmw, nsam, rnoise, rvcmobinv, lpart, rlinchisq,
rchisqp, ilimb,                    lokku, rnresloslin,rinvclos,lextinf1)
     end if                        ! end condition on macro-iteration

* calculates the new estimate of the parameters vector:
     do 70 jpar=1,itop
       rxpar(jpar)=rxparold(jpar) + rxpar(jpar)
70   continue

* the initial status of lextinf1 is restored:
     lextinf1 = lextbkp
     end
```

2.2.16 UPDPROF_PT

```
UPDPROF_PT ]
     |((((+FICARRA_PT *
     |((((+LOGINT_PT *
     |((((+GRAVITY *
     |((((+LOGINT_PT *
     |((((+LINP_PT *
     |((((+MWCONT_PT *
     |((((+FICARRA_PT *
```

Description

Updates the atmospheric profiles on the basis of the new estimate of the parameters vector *rxpar*.

Variables exchanged with external modules

Name	Description
rxpar	rxpar(imxtop) = vector of the fitted parameters

itop	itop = total number of parameters to be fitted
ipar	ipar = number of parameter-levels (i.e. N. of elements of rzpar vector)
rzpar	rzpar(imxlmb) = vector of the altitudes where the temperature profile is fitted
rzbase	rzbase(imxpro) = altitude of the base-levels
rtbase	rtbase(imxpro) = temperature of the base levels
rpbase	rpbase(imxpro) = pressure on the base-levels
ibase	ibase = number of base-levels
rcbase	rcbase(imxpro,imxmw) = continuum on the base-levels for each MW
nselmw	nselmw = total number of selected microwindows for the retrieval
rvmrbase	rvmrbase(imxpro,imxgas) = volume mixing ratio of the gases on the base levels
igas	igas = total number of different gases
roffs	roffs(imxmw) = instrumental offsets personalised for microwindow
lparbase	lparbase(imxpro) = logical vector which identifies the altitudes where the T profile is fitted, among the altitudes rzbase.
rlat	rlat = latitude of the actual limb-scan (deg.)
rztang	rztang(imxgeo) = vector containing the engineering values of tangent altitudes.
ilimb	ilimb = number of measured geometries
rzsi	rzsi(imxgeo) = tangent altitudes of the geometries to be simulated
igeo	igeo = number of simulated geometries
rbase	rbase = greater base of trapezium of Field of View function
lokku	lokku(imxgeo,imxmw) = occupation matrix used for the selection of operational MW's for each observation geometry
ilimbmw	ilimbmw(imxmw) = number of valid measured geometries per microwindow (number of 2 in each column of iocsim)
icontpar	icontpar = total number of continuum parameters to be fitted
isaved	isaved(imxsav) = vector containing all the necessary quantities for the reconstruction of continuum profiles performed by *ficarra* subroutine
nsam	nsam(imxmw) = number of sampling points in each MW (general coarse grid)
ifspmw	ifspmw(imxmw) = index of the first sampling point of each MW * NOTE: the sampling point at frequency=0 has index=1
dstep	dstep = distance between coarse-wavenumber grid points [cm-1]
rjaccon	rjaccon(imxpro*imxmw,imxcop) = jacobian matrix for the derivative of the continuum base-level values with respect to the continuum parameters
nucl	nucl = number of limb geometries to be skipped before starting continuum fit; numbering starts from top.
rperc	rperc = maximum relative (with respect to rconint) distance between central frequencies of two microwindows which are defined as close-close ones for the definition of continuum emission
rconint	rconint(imxlmb,imxmw) = frequency range around each MW, for each sweep tangent altitude, in which the continuum can be considered as varying linearly.
lcfit	lcfit(imxgeo,imxmw) = continuum occupation matrix

Module structure and detailed description

The module proceeds along the steps identified by the following bullets:

- First we verify that the tangent altitudes have not crossed:
  lhavecrossed = .false.
  do *j*=2,*ipar*
  if (*rxpar(j-1)*.ge.*rxpar(j)*) *lhavecrossed* = .true.
  end do
  if (*lhavecrossed*) then
  write(*,*)'WARNING from UPDPROF:'
  write(*,*)'The pointings tried to cross each other;'
  write(*,*)'the tangent altitudes will not be updated'
  write(*,*)'in the current iteration !!!!!'
  end if

- All the '*base*' input profiles are saved into '*old*' vectors and matrices:
  ibaseold = ibase
  begin loop I on '*base*' levels *j=1, ..., ibaseold*
  rzbaseold(j) = rzbase(j)
  rtbaseold(j) = rtbase(j)
  rpbaseold(j) = rpbase(j)
  lparbaseold(j) = lparbase(j)
  begin loop II on gases: *k=1, ..., igas*
  rvmrbaseold(j,k) = rvmrbase(j,k)
  end loop II on gases
  begin loop III on MW's: *k=1, ..., nselmw*
  rcbaseold(j,k) = rcbase(j,k)
  rcbaseor(j,k) = rcbase(j,k)
  end loop III on MW's
  end loop I on 'base' levels

- Now the indexes of the '*base*' profiles that correspond to altitudes where the T profile is fitted are identified:
  k = 1
  begin loop on the '*base*' levels: *j=1, ..., ibase*
  if *lparbase(j)* = TRUE then: *imodif(k)=j, k=k+1*
  end loop on the '*base*' levels

- At this point *k-1* should be equal to *ipar*. If these two quantities are different a fatal error is produced and the program is stopped. The occurency of this error is linked to the presence of a bug in the program.

- calculates now the scaling factors for the T profile for the regions above the highest fitted point *'rtscalabove'* and below the lowest fitted point *'rtscalbelow'*:
  rtscalabove = rxpar(ipar+1)/rtbaseold(imodif(1))
  *rtscalbelow = rxpar(ipar*2)/rtbaseold(imodif(ipar))*

- Updates now the temperature profile at the OLD altitudes, in 3 steps. The obtained profile is recorded in the vector *rtbaseor* (*rtbaseor* = rtbase-old-representation).
  Step 1: region above the highest fitted point¸ the profile is scaled
  begin loop on levels: *k=1, ..., imodif(1)-1*
  *rtbaseor(k) = rtbaseold(k) * rtscalabove*
  end loop on levels *k*
  Step 2: region between first and last fitted points (linear interpolation used).
  begin loop on parameter levels: *j=1, ..., ipar-1*
  begin loop on levels where the temp. is changed by the current parameter:
  k=imodif(j),..,imodif(j+1)
  r3=rxpar(ipar+j+1)-rxpar(ipar+j)
  r4=rzbaseold(imodif(j+1))-rzbaseold(imodif(j))
  r5 = rzbaseold(k)-rzbaseold(imodif(j))
  *rtbaseor(k) = rxpar(ipar+j) + ((r3/r4)*r5)*
  end loop on levels *k* where the temp. is changed by the current parameter
  end loop on parameter levels *j*.
  Step 3: region below lowest fitted point, the profile is scaled
  begin loop on levels: *k=imodif(ipar)+1,..., ibase*
  *rtbaseor(k) = rtbaseold(k) * rtscalbelow*
  end loop on levels *k.*

- Update of the vector *rcpar* of the continuum parameters:
  *rcpar(j) = rxpar(2*ipar+j)* for *j=1, ..., icontpar*

- Update then continuum profiles at the 'old' pressures, by using **FICARRA_PT** module. The computed profiles are recorded in the array *rcbaseor*.
  call **ficarra_pt**(*nsam, dstep, ifspmw, rcbaseor, rpbaseold, ibaseold, nselmw, ilimb, rcpar, isaved, rjaccon*)

- *rpbase* is now put in the new representation, i.e. we change the points represented in the vector *rpbase*; note that this operation does not correspond to a modification of the pressure profile, pressure is indeed the independent variable!
  begin loop on '*base*' levels: *j=1, ..., ibaseold*
  begin condition. if *rpbase(j)* < *rxpar(1)* AND *rpbase(j+1)* ≥ *rxpar(1)* then
  ihigh=j
  ibase = ihigh + ipar + 1
  begin loop on levels *k=ihigh+1, ..., ibase-1*
  rpbase(k)=rxpar(k-ihigh)
  end loop on levels *k*
  exit from loop on '*base'* levels *j*
  end condition
  end loop on '*base'* levels *j*.
  rpbase(ibase)=rpbaseold(ibaseold)

- *lparbase* vector is now updated. This operation is necessary only when *ibase* and *ibaseold* are different.
  begin loop on '*base´* levels: *j=1, ..., ibase*
  begin condition I. if *j* ≤ *ilimb+1* then:
  lparbase(ibase-j+1) = lparbaseold(ibaseold-j+1)

otherwise set:

lparbase(ibase-j+1) = FALSE

end condition I.

end loop on '*base´* levels *j*

- Put now the temperature profile *rtbaseor* in the new representation identified by *rpbase,* using log interpolation, **LOGINT** module is used:

begin loop on '*base*' levels *k=1, ..., ibase*

LOGINT*(rpbaseold,rtbaseor,ibaseold,rpbase(k),rtbase(k))*

end loop on '*base*' levels *k*

- The altitudes *rzbase* are now updated on the basis of hydrostatic equilibrium law.

The tangent altitude of the lowest sweep, provided by engineering information is assumed to be correct:

rzbase(ibase-1) = rztang(ilimb)

Calculates now the altitude of the lowest point of the profiles (the process is re-iterated once, in order to have the gravity computed more precisely):

r1 = ((rtbase(ibase)+rtbase(ibase-1)).5)*
*r2 = r1/(rmovr** GRAVITY*(rzbase(ibase-1),rlat))*
*rzbase(ibase)=rzbase(ibase-1)-r2**log*(rpbase(ibase)/rpbase(ibase-1))*
*r2 = r1/(rmovr** GRAVITY*((rzbase(ibase)+rzbase(ibase-1))/2.,rlat))*
*rzbase(ibase)=rzbase(ibase-1)-r2**log*(rpbase(ibase)/rpbase(ibase-1))*

Calculates then the other altitudes:

begin reverse loop on levels *k = ibase-2, ..., 1*, step = -1
r1 = ((rtbase(k)+rtbase(k+1)).5)*
*r2 = r1/(rmovr**GRAVITY*(rzbase(k+1),rlat))*
*rzbase(k)=rzbase(k+1) - r2**log*(rpbase(k)/rpbase(k+1))*
*r2 = r1/(rmovr** GRAVITY*((rzbase(k)+rzbase(k+1))/2.,rlat))*
*rzbase(k)=rzbase(k+1) - r2**log*(rpbase(k)/rpbase(k+1))*
end loop on levels *k*

- Interpolates VMR and continuum profiles to the new pressure grid:

begin loop I on levels: *k=1, ..., ibase*
begin loop II on gases: *j=1, ..., igas*
LOGINT_PT*(rpbaseold,rvmrbaseold(1,j),ibaseold,rpbase(k),rvmrbase(k,j))*
end loop II on gases *j*
begin loop III on MW's: *j=1, ..., nselmw*
LINP_PT*(rpbaseold,rcbaseor(1,j),ibaseold,rpbase(k),rcbase(k,j))*
end loop II on gases *j*
end loop I on levels *k*

- Computation of *rjaccon* at the new pressure grid:

1) re-computation of *rcpar:*

icpar = icontpar
MWCONT_PT*(dstep,nsam,ifspmw,rcbase,ibase,isaved,nucl,lokku,*
nselmw,lcfit,lccmat,ilimb,rperc,rconint,rcpar,icontpar)

2) if now *icontpar* is NOT equal to *icpar* a fatal error is produced: the results of the grouping of the continua at the different MW's has to produce the same results for a fixed set of MW's.

3) computation of *rjaccon*:

FICARRA_PT*(nsam,dstep,ifspmw,rcbase,rpbase,ibase,nselmw, ilimb,rcpar,isaved,rjaccon)*

- Updates the vector *roffs* of the instrumental offset:
  begin loop on microwindows: *j=1, ..., nselmw*
  roffs(nselmw-j+1) = rxpar(itop-j+1)
  end loop on microwindows *j*

- Updates now the tangent altitudes of the measurements (the lowest measurement of course is not included):
  begin loop on sweeps, *k=2, ..., ilimb*
  rztang(ilimb-k+1)=rzbase(ibase-k)
  end loop on sweeps *k*

- Updates now the vector *rzsi* that corresponds to the tangent altitudes of the simulated spectra.
  begin loop on sweeps, *k=2, ..., ilimb+1*
  rzsi(k) = rztang(ilimb-k+2)
  end loop on sweeps *k*
  rzsi(1)=rzsi(2)-(rbase/2)
  rzsi(ilimb+2) = rzsi(ilimb+1)+(rbase/2)

- Updates the vector of the unknown parameters:

A - Tangent temperatures:

```
j=0
do k=1,ibase
 if (lparbase(k)) then
  j = j + 1
  rxpar(ilimb+j)=rtbase(k)
 end if
end do
```

B - Tangent continuum:

```
do k=ilimb+ipar+1, ilimb+ipar+icontpar
 rxpar(k) = rcpar(k-ilimb-ipar)
end do
```

2.2.17 CONVCHK_PT

Description
Checks whether the convergence has been reached or a further iteration (iterg) is required.

Variables exchanged with external modules

Name	Description
rchisq	rchisq(0:imxite) = total chi-square at each iteration
iterg	iterg = index of the current macro-iteration
rlinchisq	rlinchisq = value of the chi-square, in the linear approximation, relative to the current macro-iteration

rxpar	rxpar(imxtop) = vector of the fitted parameters at the current iteration
rxparold	rxparold(imxtop) = vector of the fitted parameters in the previous iteration
ipar	n. of fitted points in the T profile
itop	itop = total number of retrieved parameters
iobs	iobs = total number of fitted spectral data points
rlambda	rlambda = Marquardt's damping factor
rthres1	rthres1 = threshold n.1 used to check convergence criteria
rthres2	rthres2 = threshold n.2 used to check convergence criteria
rthres3	rthres3 = threshold n.3 used to check convergence criteria
lconverg	lconverg = logical variable which is TRUE only if the convergence has been reached

Detailed description

- Initialisation of *lconverg*:
  lconverg = .FALSE.

- Chechk that *iterg* > 0, otherwise stops the program. This is only a consistency check, i.e. the convergence has to be checked only after the initial iteration.
```
 if (iterg.lt.1) then
   write(*,'(a)')'FATAL ERROR in CONVCHK: '
   write(*,'(a)')'Subroutine CONVCHK has been called with iterg < 1.'
   write(*,'(a)')'--------- PROGRAM STOPPED ----------'
   stop
 end if
```

- Evaluation of the first convergence criterion, i.e. variation of the chi-square. The result is stored in the logical variable *lcrit1*:
```
rchivar = abs((rchisq(iterg)-rlinchisq)/rchisq(iterg))
lcrit1 = rchivar.le.rthres1
```

- Evaluation of the second convergence criterion, i.e. max. relative variation of tangent pressure parameters. The result is stored in the logical variable *lcrit2*:
```
 rmaxvarpar = 0.
 do 100 j=1, ipar
    if (rxparold(j).ne.0) then
      rvarpar = abs((rxparold(j) - rxpar(j))/rxparold(j))
    end if
    if (rvarpar.gt.rmaxvarpar) rmaxvarpar = rvarpar
100 end do
   lcrit2 = rvarpar.le.rthres2
```

- Evaluation of the third convergence criterion on max. relative variation of tangent temperature paremeters. The result is stored in the logical variable *lcrit3*:
```
  rmaxvarpar = 0.
  do 102 j=ipar+1, 2*ipar
```

rvarpar = abs*(rxparold(j) - rxpar(j))*
if (*rvarpar*.gt.*rmaxvarpar*) *rmaxvarpar = rvarpar*
102 end do
lcrit3 = rvarpar.le.rthres3

- The final result of the convergence checks in then evaluated:
  lconverg = lcrit1 .or. (*lcrit2* .and. *lcrit3*)

2.2.18 AINVCAL_PT

AINVCAL_PT
|-----JACOBI_PT]

Description
This subroutine calculates the inverse of the matrix **A** by using the singular value decomposition method as explained in AD6. It uses the Numerical Recipes subroutine 'jacobi' in order to compute eigenvectors and eigenvalues of matrix **A**.

Variables exchanged with external modules:

Name	Description
ra	Original matrix A
itop	Dimension of the matrix ra (total n. of parameters to be fitted)
rainv	Inverse of matrix A (or 'generalised' inverse)

Detailed description: please refer to the Numerical Recipes book (RD2).

2.2.19 OUTPUT_PT

Description: Routine which generates output files according to [AD7]. Source code of this module is listed in [AD7]. Please note that this routine writes into the file 'pt_out.dat' the qualifiers characterizing continuum retrieved parameters, by using (a call to) the subroutine con_char_pt.f described in Sect. 3.2.35.

Variables exchanged with external modules

Name	Description
rxpar	See description in section 2.3
ipar	See description in section 2.3
icontpar	See description in section 2.3
rainv	See description in section 2.3
rztang	See description in section 2.3
rztanginit	Vector which contains the inital values of the rztang vector (See code in Appendix)
rvchcorr	rvchcorr(imxlmb,imxlmb) = VMC of the tangent heights corrections
nsam	See description in section 2.3
robs	See description in section 2.3
rspfov	See description in section 2.3
rchisq	See description in section 2.3
iobs	See description in section 2.3
itop	See description in section 2.3
iterg	See description in section 2.3
iterm	See description in section 2.3
rlambda	See description in section 2.3
rlinchisq	See description in section 2.3
ilimb	See description in section 2.3
igeo	See description in section 2.3
nselmw	See description in section 2.3
rchisqp	See description in section 2.3
slab	See description in section 2.3
lokku	See description in section 2.3
linloop	Swich which allows (if true) to save only information concerning each iteration and not information concerning the entire retrieval. (See the source code in appendix of AD7)
lcfit	lcfit(imxgeo,imxmw), see description in section 2.3
lccmat	lccmat(imxgeo,imxmw), see description in section 2.3
nucl	See description in section 2.3

2.2.20 LININT_PT

Description: Module used to calculate linear interpolations in the altitude domain.

Variables exchanged with external modules

Variable	Description:
rx	rx(imxpro) = vector of 'ipro' elements containing the values to which 'ry(imxpro)' profile is referred.
ry	ry(imxpro) = vector of 'ipro' elements containing the profile used for the interpolation
ipro	ipro = number of elements in rx(i) and ry(i) profiles
rx1	rx1 = value of rx where the value of the profile is required.
ry1	ry1 = value of the profile corresponding to rx1

Algorithm Description

We have a vector *ry(imxpro)*containing a general profile, the elements of this vector are referred to the altitudes recorded in the vector *rx(imxpro).* The problem is to find the value of the profile corresponding to the altitude *rx1* assuming a linear behaviour of the profile within the points represented in *ry(imxpro).* For optimisation purposes the vectors *rx* and *ry* are supposed as sorted starting from high altitudes.

Detailed description

The calculation proceeds in the following two steps:

- Search for the index *j* so that: $rx(j+1) \leq rx1 \leq rx(j);$if such index does not exist, a fatal error is produced (this can happen only if the profiles are not ordered starting from high altitudes or the requested altitude *rx1* does not belong to the range covered in the vector *ry*).
- Linear interpolation is then performed:

$$ry1 = ry(j)+((ry(j+1)-ry(j))/(rx(j+1)-rx(j)))*(rx1-rx(j)).$$

2.2.21 GRAVITY

Description: This module calculates the gravity acceleration as a function of altitude and geodetic latitude.

Variables exchanged with external modules:

Variable	Description:
rz	rz = current altitude measured on the sea level (km)
rlat	rlat = current geodetic latitude (deg.)
gravity	gravity = gravity acceleration at latitude rlat and altitude rz (m/s**2)

Detailed description

For the calculation of gravity as a function of altitude and latitude, the scientific code uses the theory described in the document AD6. We report here below the expressions which are evaluated in sequence by the code.

Let's define:

$\Phi = rlat * \frac{\pi}{180}$ the latitude in expressed in radiants,

The following expressions are then evaluated:

$$g_0 = 9.80616 \cdot [1 - 0.0026373 \cdot \cos(2\Phi) + 0.0000059 \cdot \cos^2(2\Phi)]$$

$$f = \frac{ra}{\sqrt{1 - \left(1 - \frac{rb^2}{ra^2}\right) \text{sen}^2 \Phi}}$$

Where *ra* and *rb* are respectively the equatorial and the polar radii of the earth and are defined as parameters in 'parameters.inc'.
Then we evaluate:

$$R = \sqrt{f^2 \cos^2 \Phi + \left(\frac{rb^2}{ra^2} f \cdot \text{sen}\, \Phi\right)^2}$$

Afterwards:

$$\tilde{g} = g_0 + \Omega^2 \frac{f^2}{R} \cos^2\Phi \cdot 1000$$

where: $\Omega^2 \equiv rom2 = \left(\frac{2\pi}{86400 \text{ sec./day}} / 1.002737904 \right)^2$ is the square of the angular speed of the earth. *rom2* is a parameter defined in 'parameters.inc'.

Finally we have:

$$gravity = \tilde{g}\left(\frac{R}{R+rz} \right)^2 - \Omega^2 f \left(\frac{f+rz}{R} \right) \cos^2 \Phi .$$

2.2.22 ESPINT_PT

Description
Module used to calculate exponential interpolations.

Variables exchanged with external modules:

Variable	Description:
rx	rx(imxpro) = vector of 'ipro' elements containing the values to which 'ry(imxpro)' profile is referred.
ry	ry(imxpro) = vector of 'ipro' elements containing the profile used for the interpolation
ipro	ipro = number of elements in rx(i) and ry(i) profiles
rx1	rx1 = value of rx where the value of the profile is required.
ry1	ry1 = value of the profile corresponding to rx1

Algorithm Description

We have a vector *ry(imxpro)*containing a general profile, the elements of this vector are referred to the altitudes recorded in the vector *rx(imxpro).* The problem is to find the value of the profile corresponding to the altitude *rx1* assuming exponential behaviour of the profile within the points represented in *ry(imxpro).* For optimisation purposes the vectors *rx* and *ry* are supposed as sorted starting from high altitudes.

Detailed description

The calculation proceeds in the following two steps:

- Search for the index *j* so that: $rx(j+1) \leq rx1 \leq rx(j);$if such index does not exist, a fatal error is produced (this can happen only if the profiles are not ordered starting from high altitudes or the requested altitude *rx1* does not belong to the range covered in the vector *ry*).
- Exponential interpolation is then performed:

$$ry1 = \log(ry(j)) + ((\log(ry(j+1)/ry(j)))/(rx(j+1)-rx(j)))*(rx1-rx(j))$$
$$ry1 = \exp(ry1)$$

2.2.23 LOGINT_PT

Description
Module used to calculate logarithmic interpolations of profiles that are function of pressure.

Variables exchanged with external modules

Variable	Description
rx	rx(imxpro) = vector of 'ipro' elements containing the values to which 'ry(imxpro)' profile is referred.
ry	ry(imxpro) = vector of 'ipro' elements containing the profile used for the interpolation
ipro	ipro = number of elements in rx(i) and ry(i) profiles
rx1	rx1 = value of rx where the value of the profile is required.
ry1	ry1 = value of the profile corresponding to rx1

Algorithm Description

We have a vector *ry(imxpro)*containing a general profile, the elements of this vector are referred to the pressures recorded in the vector *rx(imxpro).* The problem is to find the value of the profile corresponding to the pressure *rx1* assuming logarithmic behaviour of the profile within the points represented in *ry(imxpro).* For optimisation purposes the vectors *rx* and *ry* are supposed as ordered starting from low pressures.

Detailed description

The calculation proceeds in the following two steps:

- Search for the index *j* so that: $rx(j) \leq rx1 \leq rx(j+1)$;if such index does not exist, a fatal error is produced (this can happen only if the profiles are not ordered starting from low pressures or the requested pressure *rx1* does not belong to the range covered by the vector *ry*).
- Log interpolation is then performed:

$$r1 = ry(j+1)-ry(j)$$
$$r2 = \log(rx(j)/rx(j+1))$$
$$r3 = \log(rx(j)/rx1)$$
$$ry1 = ry(j)+((r1/r2)*r3)$$

where *r1, r2* and *r3* are scratch variables.

2.2.24 PTFROMZ_PT

PTFROMZ]
|-----GRAVITY *

Description
This module is used to calculate pressure and temperature at a given altitude, starting from pressure and temperature profiles as a function of altitude.

Variables exchanged with external modules

Variable	Description:
rzprof	rzprof(imxpro) = vector of 'ipro' elements containing the altitudes to which press. and temp. profiles are referred
rpprof	rpprof(imxpro) = vector of 'ipro' elements containing pressure profile
rtprof	rtprof(imxpro) = vector of 'ipro' elements containing temperature profile
ipro	ipro = number of elements in p, t profiles
rlat	rlat = latitude of the measurements (deg.)
rz1	rz1 = altitude where p and t are required
rp1	rp1 = pressure at the altitude rz1
rt1	rt1 = temperature at the altitude rz1

Algorithm Description
Pressure and temperature profiles are contained respectively in the vectors *rtprof* and *rpprof* which are referred to the altitudes contained in the vector *rzprof*. Given the altitude *rz1* the problem is of finding the pressure *rp1* and the temperature *rt1* that correspond to the altitude *rz1*.
The temperature *rt1* is computed using linear interpolation in the altitude domain, while pressure *rp1* is obtained using hydrostatic equilibrium law.

Detailed description
The calculations proceed in the following steps:

- Search for the index *j* so that: $rx(j+1) \leq rx1 \leq rx(j)$;if such index does not exist a fatal error is produced (this can happen only if the profiles are not ordered starting from high altitudes or the requested altitude *rx1* does not belong to the range covered in the vector *ry).*
- Linear interpolation of temperature is performed:

  rt1 = rtprof(j)+((rtprof(j+1)-rtprof(j))/(rzprof(j+1)-rzprof(j)))(rz1-rzprof(j))*

- Pressure is then computed by means of hydrostatic equilibrium law:

  *rconst = - gravity(rz1,rlat) * rmovr*
  *rp1 = rpprof(j)**exp*(rconst*(rz1-rzprof(j))/[(rt1 + rtprof(j))/2)*

Where: *rconst* is an internal scratch variable, *rmovr* is a parameter described in ‘parameters.inc’ and *gravity* is computed by **GRAVITY** function.

2.2.25 CONV_PT

Description
This module is used to perform convolution of the spectra and derivatives with the AILS function. In a call to module CONV_PT the convolution with the AILS function *rils* is performed for the original spectrum *rsp*, for the continuum derivatives *rcder* or for the temperature perturbed spectra *rsppert*.

Variables exchanged with external modules

imw	imw = index of the actual microwindow
igsim	igsim = index of the actual geometry
nsam	nsam(imxmw) = number of observed sampling points for the microwindow
nils	nils = number of elements of rils. <WARNING> nils must be an odd number
rsplank	rsplank(imxsig) = function to be convolved provided in the fine grid
rspct	rspct(imxi) = result of the convolution
rils	rils(imxils,imxmw) = instrument-line-shape function provided in the fine grid
rintils	ratio between the frequency step approximating infinitesimal spectral resolution and the summation of the interpolated-AILS-function values
nrd	ratio between the frequency steps of the coarse and the fine grid. <WARNING> this ratio must be algebrically integer

Algorithm Description
This module calculates the convolution integral between an input function, provided in the fine frequency-grid, and the AILS function as it comes from module FAILS. The result of the convolution is calculated in the coarse grid in a frequency interval that is reduced on both sides with respect to that of the input function. The reduction is equivalent to the broadening introduced by *iadd* and eliminates the truncation effects introduced by the convolution process. The resulting frequency interval and frequency grid coincide with those of the observations.

Module Structure
1. Convolution between the input function and the AILS function providing the result in the coarse frequency grid.
2. Normalisation of the results of convolution.

Detailed Description
1. Convolution between the input function and the AILS function.
The convolution integral is computed, at the i^{th} frequency as:

$$rspct_i = \sum_{k=1}^{nils} rplank_{[(i-1)*nrd+k]} \cdot rils_{(nils-k+1)}$$

where *k* is incremented by steps of 1.
The computation of $rspct_i$ is repeated for values of *i* going from 1 to *nsam(imw)*.

2. Normalisation of the results of convolution.
All the values computed at step 1 are normalized multiplying them by *rintils*.

2.2.26 MWCONT_PT

Description

This subroutine performs the grouping of the close mws among the selected ones in order to reduce the number of the continuum parameters which have to be considered for each mw in the retrival process.

The grouping is made possible by the hypothesis of linear variation of the atmospheric continuum value on frequency ranges ('umbrellas') that are larger than average width of a selected mw, so that it is possible to linearly interpolate the continuum value of a internal mw between the values of sufficiently close external mws. This routine looks for the possible grouping starting from the knowledge of the position of all the mws, their width , the range of linearity of the continuum for each mw and for each geometry.

Variables exchanged with external modules

Name	Description
dstep	dstep = distance between coarse-wavenumber grid points [cm-1]
nsam	nsam(imxmw) = number of sampling points in each mw (coarse grid)
ifspmw	ifspmw(imxmw) = index of the first sampling point of each MW
rcbase	rcbase(imxpro,imxmw) continuum on the base-levels for each MW
ibase	ibase = number of base-levels
isaved	isaved(imxsav) = vector containing all the necessary quantities for the continuum interpolation in ficarra subroutine
nucl	nucl=index of the higher covered geometry
lokku	lokku(imxgeo,imxmw) = logical occupation matrix
nselmw	nselmw = N. of selected microwindows
lcfit	lcfit(imxgeo,imxmw) = occupation matrix of continuum fitted parameters
lccmat	lccmat(imxgeo,imxmw) = occupation matrix of Close-Close Mw's
ilimb	ilimb = N. of considered sweeps
rcperc	rperc = percent of overlap between two MW (if dist% < rperc then CC)
rconint	rconint(imxgeo,imxmw) = width of the umbrella for each geo and MW
rcpar	rcpar(imxcop) = vector of the considered continuum parameters
icontpar	icontpar = numper of the considered continuum parameters

Algorithm Description:

The goal of this module is to group, for each geometry, the close (in frequency domain) mws in order to pass to the following subroutines a reduced number of continuum parameters. If no grouping is applied, for each mw a continuum profile has to be considered in retrival algorithm; instead, if some grouping is performed only the continuum value for edge mws of grouped ones can be passed (see Fig. 4). So in this particular case if no grouping is made the continuum parameters are c1,c2,c3,c4,c5 , while if there is an overlap of the 'umbrella' of the mws from #1 to #4, a grouping can be performed and the continuum parameters are only c1,c4 and c5.

This routine also performs a 'holes hunting' in occupation matrix, looking for corrupted spectra which break the continuity of True values in the columns of *lokku*.

The found 'holes' are catalogued and the related information is stored in the vactor *isaved*. This information is used at a later stage, when continuum profiles are reconstructed by FICARRA_PT module starting from the vector *rcpar* of the continuum parameters.

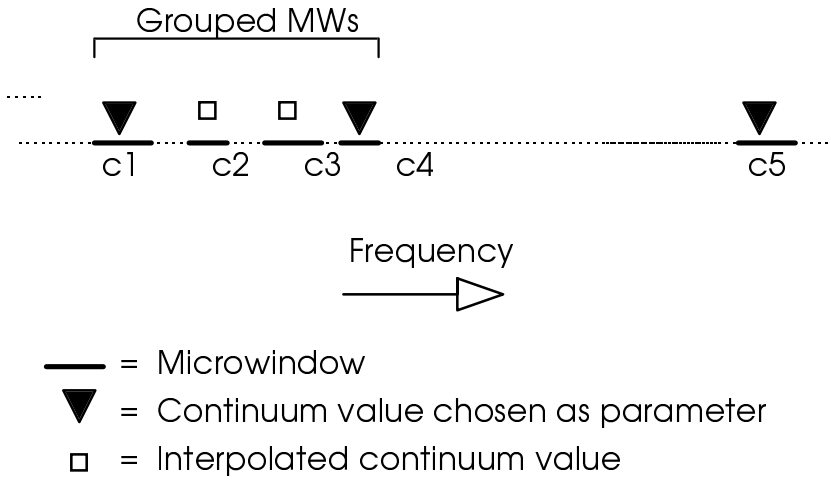


Fig 4

Module structure

The module operates by the following steps:

1. building, for each mw, *rsig1*,*rsig2* and *mwcen* , which respectively are the two edge in frequency range of the mw and its middle value, starting from *nsam*,*dstep* and *ifspmw*.
2. building *iguv* vector
3. building *iquale* and *nmw* arrays

DoLoop on geometries

- 4.a filling of *itab*
- 4.b test of close-close mws condition
  - 4.b.1 if true then filling of *cclist* array
- 4.c filling of *ipunt* array
- 4.d filling of *icontch* array
- 4.e filling of *intrp_s* array

End Do

5. building *icontpr* array
6. building *rcpar* vector
7. start of hole hunting : if some hole exists then
   - 7.a filling of *nholes vector*
   - 7.b filling *holelist*,*ntotholes*

8. building *isaved* sequentially written vector which contains all the necessary quantities for *ficarra* subroutine.

Detailed description

0. *nucli=nucl+1*

1. *rsig1*,*rsig2*,*rmwcen* are built as:

Do Loop j: 1 → *nselmw* ! number of selected mws

rsig1(j) = (*ifspmw(j)-1)*dstep*
rsig2(j) = rsig1(j) + (*nsam(j)-1)*dstep*
rmwcen is the middle point between rsig1 and rsig2
End Do

2. *iguv(2,imxmw)* is built by finding , for each mw, the upper and the lower True value in the columns of *lokku(imxgeo,imxmw)* (logical occupation matrix) between *nucli* line and the bottom of the matrix. It rapresents for each mw the upper and the lower covered geometry.

```
Do Loop K: 1 → nselmw
 Do Loop jgeo: nucli → ilimb
    if(lokku(jgeo,K)) idown=jgeo
    if(lokku(ilimb + nucli - jgeo,K)) iiup=ilimb + nucli - jgeo
 end do
 iguv(1,K)=iiup     !iguv(1, . ) = upper element
 iguv(2,K)=idown          !iguv(2, . ) = downer element
end do
```

3. *iquale(imxgeo,imxmw)* contains the absolute index (i.e. a number from 1 to that of selected mws) of True mws for each geometry. It is built by running, for each geometry, between *nucli* line and the bottom, on a row of *lokku* matrix and looking for True value.For example if *lokku* matrix is something like that in Fig. 5 , the *iquale* matrix will be as shown in Fig. 6.

```
Do Loop  jgeo: nucli → ilimb
    kont=0
    Do Loop  j: 1 → nselmw
     if(lokku(jgeo,j))then
       kont=kont+1
       iquale(jgeo,kont)=j
     end if
    end do
    Nmw(jgeo)=kont   (Number of True on the jgeo-th row of lokku)
end do
```

```
    1 2 3 4 5 6 7 8 9
1   F T T T T F F F F
2   T T T T F F F T T
3   T T T T F T T T T
    . . . . . . . . .
```

Fig.5 - Example of occupation matrix

```
    1 2 3 4 5 6 7 8 9
1   2 3 4 5 . . . . .
2   1 2 3 4 8 9 . . .
3   1 2 3 4 6 7 8 9 .
```

Fig.6 - example of

***iquale* matrix**

The *Nmw(imxgeo)* contains, for each geometry, the number of mws that we have to consider: if the occupation matrix is that shown in Fig. 5 then the first three elements of *nmw* will be : *Nmw(4 , 6 , 8 , ...)*

4. Building the main arrays of the routine.
   All the mws have a range in frequency where their value of the atmospheric continuum can be cosidered as linear. This range has a width equal to *rconint(imxgeo,imxmw)* on the right and the left side with respect to the centre of the mw; this frequency range will be called 'umbrella'. Now, in order to make easier the construction of the grouping routine, the better choice is to represent the overlap between the mws for each geometry in a scheme such as that shown in Fig.7.

```
  1 2 3 4 5 6 7 8 9 . . .
1 * * o o o o o o o
2 * * * o o o o o o
3 o * * o o o o o o
4 o o o * * o o o o
5 o o o * * * * o o
6 o o o o * * * o o
7 o o o o * * * * o
8 o o o o o o * * o
9 o o o o o o o o *
.
```

Fig.7 - Example of an overlap map for a fixed geometry

The rows and the columns represents the mws. If on the map there is a * then the relative mws are overlapped otherwise they are not overlapped.
Hence this square map shows, for a fixed geometry, how many mws are each other overlapped: from the figure it can be seen that, for example, the first mw (row 1) covers with its umbrella also the second mw, while the second mw (2th row) covers the first but also the third mw. Obviously since the range of linearity of the atmospheric continuum (the width of the umbrellas) is very similar for close mws, if a mw covers the just next one, it's clear that also this following mw has an umbrella which covers the previous mw. Therefore this map will be always *symmetric* with respect to the diagonal.

The grouping routine performs the search of the *maximum length* horizontal 'stream' of '*' that starts from each column of the matrix shown in Fig.7. That will be the maximum number of laying in a linearity zone mws. Then the grouping is performed between the outer mws of this stream: only the continuum value of these mws has to be considered as parameter, instead all the other, corresponding to internal mws, will be linearly interpolated.

This procedure has to be repeated, starting now from the following mw the 2nd edge of the previous range, until the border of the array is reached.

In this example, the grouping will be (fig.8):

```
  1 2 3 4 5 6 7 8 9 . . .
1 * • •
2 • = •
3 • • *
```

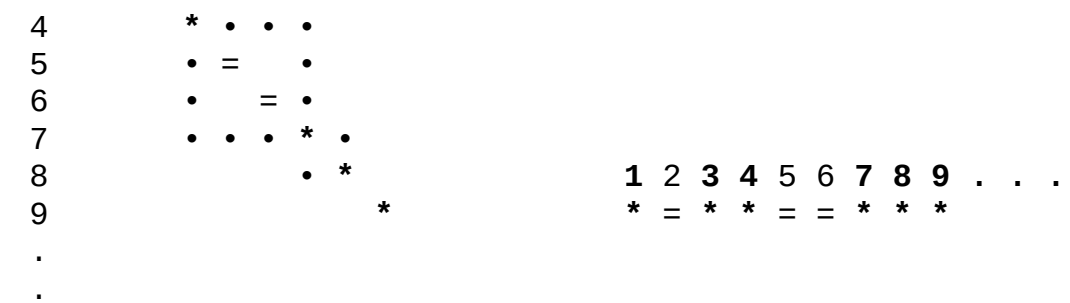


Fig.8 - Map of grouping for the mws at a fixed geometry: The bold stars indicate the mws which have to be chosen as parameter, all the other mws will be linearly interpolated:
mw #2 will be interpolated between the #1 and the #3
mw #5 and #6 between the #4 and the #7

kccgeo=1 :counter of geometries with CC mws

[A] Do loop on the geometries : *jgeo=nucli* → *ilimb*

iTABold1=0 :old value of itab(. ,1)
iTABold2=0 :old value of itab(. ,2)
kccmw=1 :counter of CC mw coditions for actual geometry
Lexcc=.FALSE. :logical flag that indicates if exists CC at least a CC mw condition for actual geometry
rMWcenf=0. :value of the center of the previous mw

[B] Do Loop on the mws : *ind=1,Nmw(jgeo)*

4.a.1) Calculation of edge frequencies of the *ind*-th mw:
imw=iquale(jgeo,ind)
rSIGmin=rMWcen(imw) – rconint(jgeo,imw) :lower edge of umbrella
rSIGmax=rMWcen(imw) + rconint(jgeo,imw) :upper edge of umbrella

4.a.2) Filling of *itab(imxmw,2)* array.
It contains in its two columns the starting point and the length of each stream of '*' in Fig.7. For example, in the case shown in Fig.7, *itab* will have the values:

itab	1	2
1	***1***	***2***
2	***1***	***3***
3	***2***	***2***
4	***4***	***2***
5	***4***	***4***
6	***5***	***3***
7	***5***	***4***
8	***7***	***2***
9	***9***	***1***

```
        kont=1                                  Init. of a counter
[C]     Do Loop on the mws : ind2=1,Nmw(jgeo)       doloop on all the mws
          imw2=iquale(jgeo,ind2)
          if (rMWcen(imw2)  > rSIGmin and           Is the center of ind2-th mws
             rMWcen(imw2) <  rSIGmax)               under the umbrella?
          then
             kont=kont+1
             kont2=ind2
          end if
[eC]    End Do
          itab(ind,1)=kont2-kont+2      initial mw
          itab(ind,2)=kont-1            number of covered mw
```

4.a.3) Test on the Close-Close mws condition
Filling of *NmwCC(imxgeo),CClist(imxgeo,imxmw)*

This module looks for close-close mws, i.e. for mws the central points of which are far each other less then a percent *rperc* of their umbrella. If some cc mw exists, in *nmwcc(imxgeo)* the number of cc pairs for each geometry will be contained, while in *cclist(imxgeo,imxmw)* each row represents a geometry where is localized a cc pair:
cclist(geo , 1) = geometry where is at least a cc pair
cclist(geo , 2 → nmwcc(geo)+1) = parent mw for each of the *nmwcc(geo)* cc pair

```
          If  (iTAB(ind,1)-iTABold1 = 0)    AND                 condition on stream
              (iTAB(ind,2) = 2 AND iTABold2 = 2) AND              condition on stream
             (rMWcen(imw)-rMWcenf)/rconint(jgeo,imw) < rperc)     centers very close
        then
         Lexcc=.TRUE.          :a CC couple exists.
         LvetCC(ind-1)=.TRUE.    :the previous mw is the first of the two CC mws
         Nmwcc(Kccgeo)=Nmwcc(Kccgeo)+1   :in this geometry the number of CC
                                         :increased of 1
         CClist(Kccgeo,1 + Kccmw)=ind - 1       :the index of this first mw is stored
              Kccmw=Kccmw+1                  :the counter is increased of 1
        end if
             iTABold1=iTAB(ind,1)     : old value = new value
             iTABold2=iTAB(ind,2)      :old value = new value
             rMWcenf=rMWcen(imw)  :old mw center = new mw center
[eB]    End Do

        IF Lexcc is TRUE            :If at least a CC mw exists
        then
          CClist(Kccgeo,1) = jgeo    :geometry which contains the CC mws
          Kccgeo=Kccgeo+1           :the counter is increased of 1
        end if
```

4.a.4) Filling of *ipunt(imxmw)* internal array.
It indicates how long is the longest stream that leaves from each column of the matrix in Fig. 7. In this case it is:

ipunt (3 , 2 , 0 , 4 , 4 , 0 , 2 , 0 , 1)

ipunt can be filled in the following way:

```
kont=1    :initialization of counters
imw=1
Do while    imw ≤ Nmw(jgeo)
  if  itab(imw,1) =  kont
  then
    ipunt(kont) = itab(imw,2)
    imw=imw+1
  else
    kont=kont+1
  end if
end do
```

4.a.5) Filling of *icontch(imxgeo,imxmw),itotmwcont(imxgeo)* arrays.

icontch(,) is a fundamental array which contains, in each row (for each geometry), the list of the relative indexes of the mws that have to be chosen as parameter. It is filled according to a criterion that groups a number as great as possible of mws. *itotmwcont()* stores, for each geometry, the number of chosen mws. For each geometry, the elements of *icontch(,)* are the indexes pointed by *ipunt( )* vector according to the following scheme:

ipunt (3 , 2 , 0 , 4 , 4 , 0 , 2 , 0 , 1)

	1	2	3	4	5	6	7	8	9
1	● #	o							
2	o	o	o						
3		o	● #						
4				● #	o				
5				o	o	o	o		
6					o	o	o		
7					o	o	● #	o	
8							o	● #	
9									● #

In this table the filled circles represent the mws that will be chosen, while all the circles indicate the streams. Starting from 1st mw, the second mws can be chosen by skipping a number of mws according to the 1st element of *ipunt* minus one , i.e. 2 , and jumping with the index on the 3rd element of *ipunt* as next value. The skipped mws will be interpolated. Therefore the mw #2 can be skipped and its continuum value can be interpolated: infact from the 2nd row in this table it's clear how the range of linearity for the continuum of the 2nd mw covers also the 1st and the 3rd mw. So, in this case, the first two value of *icontch* will be 1 and 3. Then the process goes on: the 3rd element of *ipunt* is null. Infact from the 3rd column of the table no stream leaves. The counter is then increased of 1, the 4th mw is chosen as parameter, and the index jumps forward of (4 – 1) (4 is the value of the 4th element of *ipunt*), 4+(4-1) is chosen as the 4th element of *icontch*: so the 7th mw has to be considered. In this jump two continuum parameters have been saved, so the mws #5 and #6 will be interpolated. Newly the 7th mws is considered as an edge of an interval, which we are

looking for the second edge of. The 7th element of *ipunt* will tell us how many mws we have to skip in order to found the next valid mw (2 – 1), so the next mws is just the following one. From this no stream leaves, so the counter is increased and the successive (and last) mw is chosen.
For this geometry the value of *icontch(,)* will be:

icontch(this geom, 1→ itotmwcont(jgeo)) = (1 , 3 , 4 , 7 , 8 , 9)

where *itotmwcont(jgeo)* is the total number of mw where the atmospheric continuum has to be considered: 6.

This algorythm has to be changed when two CC mws are encountered.
In this particular case two conditions will be TRUE at the same time:
1.) the lenght of the stream is 2
2.) LvetCC(imw) (calculated in [4.a.2]) is TRUE
When both these conditions are satisfied, the mws has to be skipped and the counter increased of 1.

```
kont=1
imw=1
iCONTch(jgeo,1)=1
Do while imw ≤ Nmw(jgeo)
  iwhere = ipunt(imw)            :where does ipunt point ?
  if  iwhere ≥ 2
  then
    if (iwhere = 2) AND (LvetCC(imw) is TRUE)   :it's a CC mw
    then
      imw=imw+1          :It isn't a new parameter (CCmw)
      kont=kont-1
    else                                        :iwhere > 2
      iCONTch(jgeo,kont+1)=imw+iwhere-1         :chosen parameter
      imw=imw+iwhere-1
    end if
  else          :if iwhere = 0 , 1 (end of the stream or isolated point)
    iCONTch(jgeo,kont+1)=imw+1
    imw=imw+1
  end if
  kont=kont+1
end do
iTOTmwcont(jgeo)=kont-1      :This sets the total number of considered mws
iCONTch(jgeo,kont)=0         :This kills unuseful last value...
```

4.a.6) Filling of the vector of the to be interpolated mws *iNtrp(imxmw,imxgeo)*

The number of continuum values that have to be interpolated is equal to the difference between the indexes of two successive chosen mws:
icontch(. , i) - icontch(. , i-1). If the previous mw is the first of a couple of CC mws, no interpolation has to be performed.

```
Do Loop : ind=2 → iTOTmwcont(jgeo)
  if LvetCC(iCONTch(jgeo,ind-1)) is TRUE    :if the previous mw is the first
  then                                      :of a couple of CC mws
```

```
 iNtrp_s(ind-1,jgeo)=0                    : → no interpolation

 else                                     :Number of the values
                                          :that have to be interpolated
  iNtrp_s(ind-1,jgeo)=iCONTch(jgeo,ind)-iCONTch(jgeo,ind-1)-1
 end if
end do
```

4.a.7) Re-initializations

```
Do Loop ind2  =1 → nselmw
 itab(ind2,1)=0
 itab(ind2,2)=0
 ipunt(ind2)=0
End Do
```

[eA] End Do :end of outer loop on the geometries

5.) Filling of *icontpr(imxgeo,imxmw)* array

It contains the progressive enumeration of the chosen mws, from *nucli* to *ilimb* and, for each geometry, from 1 to *itotmwcont(*act. geometry*).*

```
kont=1
Do Loop : jgeo = nucli → ilimb
 Do Loop : ind =1 → iTOTmwcont(jgeo)
  iCONTpr(jgeo,ind)=kont
  kont=kont+1
 End Do
End Do
```

6.) Filling of *rcpar(imxcop)* vector. Calculation of *icontpar*

rcpar(imxcop) contains the continuum values for the chosen mws. The enumeration is that of 5.). The *icontpar* is the total number of these continuum parameters. The values are got from the *rcbase(imxpro,imxmw)* by pointing the correct elements by *iquale* and *icontch* array.

```
icontpar=1
Do Loop : jgeo=nucli → ilimb
  Do Loop : ind=1 → itotmwcont(jgeo)
    rcpar(icontpar)=rcbase(  ibase-1-ilimb+jgeo ,
...                          iquale(jgeo,icontch(jgeo,ind))  )
    icontpar=icontpar+1
  End do
End do
icontpar=icontpar-1
```

(jgeo,ind) → indexes on geos and mws

*icontch(*jgeo,ind*)* → relative index of the chosen mws
The relative index runs only on the *chosen as parameter* mws (for a given geometry), from 1 to *itotmwcont(current geom).*

iquale(jgeo , *icontch(*jgeo,ind*))* → absolute index of the chosen mws
The absolute index indicates instead the column of *lokku* array where there is the TRUE.

The meaning of these array is shown in the following example. Given an occupation matrix and marked with a bold **T** the mws that have to be chosen, *iquale* and *icontch* will be:

```
    1 2 3 4 5 6 7 8 9
1   F T T T T F F F F
2   T T T T F F F T T
3   T T T T F T T T T
    . . . . . . . . .
```

Example of occupation matrix

```
    1 2 3 4 5 6 7 8 9
1   2 3 4 5 . . . . .
2   1 2 3 4 8 9 . . .
3   1 2 3 4 6 7 8 9 .
```

example of *iquale* matrix

```
    1 2 3 4 5 6 7 8 9
1   1 4 . . . . . . .
2   1 3 4 5 6 . . . .
3   1 4 5 8 . . . . .
```

example of *iCONTch* matrix

Infact for the first considered geometry, I chose the 1st and the 4th 'True' in the occupation matrix, in the second geometry I chose the 1st , the 3rd , the 4th , the 5th and the 6th 'True' et cetera.
It is possible to know what is the absolute index of a chosen mws in this way:

iquale(jgeo , *icontch(*jgeo,ind*))*

1st Example: Geometry 2

3rd chosen mw (iCONTch (2 , 3) = 4)
It is the 4th 'True' in occupation matrix at the geometry 2
iquale(2 , iCONTch(2 , 3)) → *iquale*(2 , 4) = 4
(from 1 to nselmw, the absolute index of the chosen mw is 4)

2nd Example: Geometry 3

3rd chosen mw (iCONTch (3 , 3) = 5)
It is the 5th 'True' in occupation matrix at the geometry 3
iquale(3 , iCONTch(3 , 3)) → *iquale*(3 , 5) = 6
(from 1 to nselmw, the absolute index of the chosen mw is 6)

7.) Holes hunting in occupation matrix

This module looks for 'holes' , i.e. corrupted spectra, in the columns of *lokku* matrix.
A hole is an interruption of the continuity in TRUE values in the columns of *lokku*.
The search is made by testing if between *iguv(1,imw)* and *iguv(2,imw)* there are values FALSE of *lokku*. If it happens the counter of the total number of holes is increased of 1 and the information of where is the hole is stored in *holelist* array. In this array there are how many rows as the number of mws where holes have been found, in the first column the relative index of the holed mw is stored and the *Nholes(imxmw)* following columns contain the list of the holed geometries. *nholes(imxmw)* is an array which contains the number of holed geometries for each mw.

```
irow=1                          :
NtotHoles=0                     :initializations
Nholes(1→imxmw)=0                    :
Nmwholed=0                        :
LMB=.FALSE.                          :

Do Loop : imw=1 → nselmw
 Do Loop : jgeo=iguv(1,imw) → iguv(2,imw)
        if  .NOT.lokku(jgeo,imw) is TRUE   :if the lokku element is false
        then
         NtotHoles=NtotHoles+1             :total number of holes  = + 1
         Nholes(irow)=Nholes(irow)+1       :number of holes for this mw
         LMB=.TRUE.                        :logical flag : 'the mw is holed'
          Holelist(irow,1)=imw                 :what is the holed mw?
          Holelist(irow,1+Nholes(irow))=jgeo    :what is the corrupted geom?
        end if
 End Do
  if  LMB is TRUE     :if this is a holed mw
  then
   Nmwholed=Nmwholed+1
   LMB=.FALSE.
   irow=irow+1
  end if
End Do
```

8.) Filling of the vector *isaved(imxsav))* containing all the quantities used in *ficarra* sub.

In this block all the useful data are sequentially written on a vector of integers. These quantitities are respectively:

0.) *nucli*
1.) *itotmwcont( )* (from *nucli* to *ilimb)*
2.) *nmw( )* (from *nucli* to *ilimb)*
3.) *icontch(* a ,b *)* *(a* from *nucli* to *ilimb)*

(b from 1 to *itotmwcont(*a*))*

4.) *icontpr(* a , b*) (a* from *nucli* to *ilimb)*
(b from 1 to *itotmwcont(*a*))*

5.) *iquale(* a , b*) (a* from *nucli* to *ilimb)*
(b from 1 to *nmw(*a*))*

6.) *intrp_s(* a , b-1*) (a* from *nucli* to *ilimb)*
(b from 2 to *itotmwcont(*a*))*

7.) *nmwholed*

8.) *ntotholes*

9.) *nholes( )* *(*from 1 to *nmwholed)*

10.) *holelist(* a , b*) (a* from 1 to *nmwholed)*
(b from 1 to *nholes(*a*)+1)*

11.) *ngeocc*

12.) *nmwcc( )* *(*from 1 to *ngeocc)*

13.) *cclist(* a , b*) (a* from 1 to *ngeocc)*
(b from 1 to *nmwcc(*a*)+1)*

14.) *iguv(* a , b *) (a* from 1 to 2)
(b from 1 to *nselmw)*

2.2.27 FICARRA_PT

FICARRA_PT
|-----FINDPAR_P +

Description
This module generates the new profiles of the atmospheric continuum (*rcbase*), starting from the vector of the continuum parameters *rcpar* and from the control array *isaved* which is returned by *mwcont* subroutine. In the module, the matrix of the derivatives (*rjaccon*) of all continuum profiles with respect to the continuum parameters is also computed.

Variables exchanged with external modules

Name:	Description:
nsam	nsam(imxmw) = number of sampling points in each mw (coarse grid)
dstep	dstep = distance between coarse-wavenumber grid points [cm-1]
ifspmw	ifspmw(imxmw) = index of the first sampling point of each MW
rcbase	rcbase(imxpro,imxmw) continuum on the base-levels for each MW
rpbase	rpbase(imxpro) = pressure of the base-levels
ibase	ibase = number of base-levels
nselmw	nselmw = N. of selected microwindows
ilimb	ilimb = N. of considered sweeps
rcpar	rcpar(imxcop) = vector of the considered continuum parameters
isaved	isaved(imxsav) = vector containing all the necessary quantities for the continuum interpolation in ficarra subroutine
rjaccon	rjaccon(imxpro*imxmw,imxcop) = array of the derivatives of the atmospheric continuum profiles respect to the continuum parameters

Module structure

1. Filling of the control arrays generated in *mwcont* subroutine by loading their values from *isaved* vector
2. Filling of internal *rcon* array from *rcpar* vector by using the informations stored in the control arrays. *rcon* contains the values of the atmospheric continuum of the parameters in their correct positions *rcon*(*index geom, index mw*) for geometry from *iguv(1,index mw)* to *iguv(2,index mw).*

3. Copying the continuum value of the 'parent' mws in the 'son' mws of the close-close couples of mws (if they exist).
   3.1 Computation of the derivative with respect to the continuum parameters of these elements

4. Interpolations and scaling
   4.1 Interpolations of the continua in frequency domain (if needed): this permits to obtain the continuum values for internal mws of grouped ones.
   Computation of the derivatives with respect to the continuum parameters.
   4.2 Scaling of the continuum profiles in the altitude range outside the altitude range covered by the considered mw.
   Computation of the derivatives with respect to the continuum parameters.

4.3 Filling of the gaps (if any) in the altitude domain of the corrupted spectra with linearly interpolated values in pressure.

Computation of the derivatives with respect to the continuum parameters.

5. Copying of *rcon* in *rcbase* array
6. Body of the function *findpar_p*

Structure of the matrix containing the derivatives with respect to the continuum parameters (*rjaccon)*.

This matrix has *nselmw*ibase* rows and a number of colums equal to the number of the continuum fitted parameters (*icontpar*). It contains, for each column, the derivative of the elements of all continuum 'base' profiles for all the mws, with respect to the parameter corresponding to that column.

Detailed description:

1. Filling of the control arrays by loading values from *isaved* vector

The following quantities and arrays are sequentially loaded from *isaved* vector:

nucli,itotmwcont( ),nmw( ),icontch(,),icontpr(,),iquale(,),intrp_s(,),
nmwholed,ntotholes,nholes( ),holelist(,),nmwCC( ), ngeocc,cclist(,),iguv(,)

The meaning of all these quantities is given in the description of *mwcont* routine.

Filling of rQT matrix from rpbase matrix . This matrix contains the pressure value at the altitudes corresponding to the geometries of observation.

.

```
do jgeo=1,ilimb
  rQT(jgeo)=rpbase(ibase-1-ilimb+jgeo)
end do
```

2. Filling of internal *rcon* array from *rcpar* vector

rccon is now filled with only the continuum values corresponding to the chosen mws (parameters). All the other continua will be or computed by linear interpolation in frequency range (continua related to grouped mws) or by scaling (continua above the upper covered altitude and below the lower covered altitude) or by linear interpolation in pressure range (continua related to corrupted spectra).

kont=1
Do loop 1 *jgeo=nucli* $\rightarrow$ *ilimb*
  Do loop 2 *ind_mw=1* $\rightarrow$ *itotmwcont(jgeo)*
   rcon(jgeo,iquale(jgeo,icontch(jgeo,ind_mw)))=rcpar(kont)
   kont=kont+1
  End do 2
End do 1

Where *iquale(geo,mws)* contains, for each geometry, the absolute indexes of the True elements in the rows of *lokku* matrix (i.e. the index of their column), while *icontch(geo,mws)* contains, for each

geometry, the relative index (from 1 to *itotmwcont(geo),* (total number of chosen mw for actual geometry)) of the chosen mws. These two arrays are set up in *mwcont* module. Here can be inserted the initialization of the array *LCCarr(imxgeo,imxmw):*

```
Do loop 1  jgeo=nucli → ilimb
   Do loop 2 ind=1→ nselmw
    LCCarr(jgeo,ind)=.FALSE.
   end do 2
 end do  1
```

3. Copying the continuum value of the 'parent' mws in the 'son' mws of the close-close couples of mws (if they exist).

```
      Do loop 1 jgeo=1→ ngeocc                !do loop on geom. with at least a pair of cc mws
         k=cclist(jgeo,1)                     !geometry with the close-close couple of mws
          Do loop 2 ind_mw=1→ nmwcc(jgeo)     !do loop on the Close-Close mws
          k1=cclist(jgeo,ind+1)               !local position of the 1st cc element
          rcon(k,iquale(k,k1+1))=rcon(k,iquale(k,k1)) !copy of the continuum value in the son mw
```

If a geometry with close-close couple of mws exists (*ngeocc* $\neq 0$), the value of the continuum for the 'son' mw is set equal to that of the 'parent' mw. The informations about the geometry and the position of cc couples of mws have been stored (by *mwcont* module) in *cclist(,)* array: for each row, the first element *cclist(1, .)* contains the geometry where there is at least a cc couple of mws , the following elements, from 2 to *nmwcc(geo),* contain the relative index of the 'parent' mws of each cc couple.

3.1 Computation of the derivative of the continuum value of cc mws with respect to the continuum parameters.

The *ind_par* index points to the correct column of the *icont_pr(,)* array. This has the same structure of the *icontch(,)* matrix and it contains, for each geometry, the progressive index of the parameter corresponding to each chosen mw. *ind_par* can be found, within these two nested do-loops, for each geometry, by running from 1 to the total number of chosen mws for the considered geometry (*ind:* from 1 to *itotmwcont(geo)*) and by testing if the relative index of the mw (i.e. the value of *icontch(act geo, ind)*) is equal to the index of the 'parent' mw, *k1*.

```
   do loop 3 k2=1,itotmwcont(k)   !looking for the index in icontpr
     if(icontch(k,k2).eq.k1) ind_par=k2
   End do 3
```

Within the two previous nested do-loops can be put also the computation of the derivative of the continuum values of the two close-close mws with respect to the continuum parameters. In this particular case the derivative of the 'parent' mw continuum value (which is a parameter) is not zero (and it is equal to 1.) only when computed respect to itself, as well as the derivative of the 'son' mw continuum value is 1. respect to only the 'parent' mw parameter and zero elsewhere.

Therefore the *rjaccon(,)* array can be filled only in non zero value sites by setting equal to "1." only those two elements:

```
  rjaccon( (iquale(k,k1)-1)*ibase+(ibase-1-ilimb)+k,
&       icontpr(k,ind_par))=1.                  !for the first of cc mw
  rjaccon( (iquale(k,k1+1)-1)*ibase+(ibase-1-ilimb)+k,
&       icontpr(k,ind_par))=1.                  !for the son mw

        LCCarr(k,k1)=.True.
     End do 2
   End do 1
```

4. Interpolations and scaling

4.1 Interpolation in frequency domain between two mws of which continuum value has been chosen as parameter

This interpolation allows to restore the continuum values of mws which have their central point laying in the covering range of the umbrellas of edge mws of a set of grouped mws.
The continuum value for these internal mws is computed by linear interpolation in frequency domain between the continuum values of the two chosen as parameter external mws, according to the expression:

$$C_i = C_1 + \frac{(C_2 - C_1)}{(\sigma_2 - \sigma_1)} \cdot (\sigma_i - \sigma_1)$$

where C_1 and C_2 are the continuum values for the edge mws, σ_1 and σ_2 their central frequencies and σ_i the central frequency of the mw of which the continuum value has to be interpolated. For each geometry, by using the arrays *iquale*, *icontch* and *intrp_s*, we can correctly point the absolute index of the to be interpolated mw.

The structure is the following:

```
Do loop 1  jgeo= nucli → ilimb
      kont=1      !on all the 'True' mws on a row of lokku
      kont2=1     !only on the chosen mws
  (set of the quantities for the first chosen mw at this geometry)
  iq=iquale(jgeo,icontch(jgeo,1))   !absolute index of the first chosen mw
 vc1=rcon(jgeo,iq)                  !continuum value for this mw
 sig1=rmwsig(iq)                    !central point for the chosen mw

 (does some interpolated value exist between this mw and the next one ?)
 if( intrp_s(kont2,jgeo) ≠ 0 ) then
  kont2=kont2+1
  iq=iquale(jgeo,icontch(jgeo,1))   !absolute index of the second chosen mw
 vc2=rcon(jgeo,iq)                  !continuum value for this mw

 !central point for the chosen mw. It can be computed starting from dstep , nsam and ifspmw
 sig2=rmwsig(iq)

      factor=(vc2-vc2)/(sig2-sig1)
```

```
          (cycle on the interpolated mws)
          Do loop  2  ind_mw = 1 → intrp_s(kont2-1,jgeo)
           kont=kont+1
           iq=iquale(jgeo,kont)
            rcon(jgeo,iq)=vc1+(rmwsig(iq)-sig1)*factor
          End do 2

       kont=kont+1
    end if
```

Here can be placed also the code which calculates the analitical derivatives of all the interpolated values with respect to the two continuum parameters of the edge mws, i.e. the derivative of *rcon(jgeo,iq)* with respect to *vc1* and *vc2*. In particular the row and the column indexes of *rjaccon* have to be correctly pointed. In order to calculate the derivatives the previous code can be modified in the following way:

```
    Ngs=ibase-1-ilimb            !The index upper quote of the scan
    do jgeo=NUCLI ,ilimb         !DoLoop on the geometries

      kont=1                     !on all the mws of a row of LOKKU
      kont2=1                    !on the parameters only
     iq=iQUALE(jgeo,iCONTch(jgeo,1))       !absolute index
     VC1=rCON(jgeo,iq)                     !Continuum of the first mw
     sig1=rMWsig(iq)                       !sigma of the first mw
      do while(kont2.le.iTOTmwcont(jgeo)-1)     !Loop on all the chosen as parameter mws
        iq=iQUALE(jgeo,iCONTch(jgeo,kont2))  !Absolute index of the present mw
        VC1=rCON(jgeo,iq)                          !Its continuum value
        sig1=rMWsig(iq)                      !Its central sigma
       iro1 = (iq-1)*ibase + Ngs + jgeo      !Row in rJACcon (for computation of the derivative)
        ico1 = iCONTpr(jgeo,kont2)                !Column in rJACcon (for computation of the
derivative)
                                    !iCONTpr contains the progressive index of the
                                !parameters related to the chosen mws (i.e. by this value it
                                !is possible to point the columns in rJACcon)

        rJACcon(iro1,ico1)=1.  ! This is the derivative of the continuum for a chosen mw or
(eventually)        for the first "edge" mw (of a group) with respect to itself. It useful set it to 1.
here)

       if(iNtrp_s(kont2,jgeo).ne.0)then          !if some interpolated value exists
        kont2=kont2+1
        iq=iQUALE(jgeo,iCONTch(jgeo,kont2))   !absolute index of the 2nd end
          VC2= rCON(jgeo,iq)
          sig2=rMWsig(iq)
            iro2 = (iq-1)*ibase + Ngs +jgeo  !Row in rJACcon
            ico2 = iCONTpr(jgeo,kont2)       !Column in rJACcon
            rJACcon(iro2,ico2)=1.            ! This is the der. of the continuum for thw 2nd edge
             rFT=(VC2-VC1)/(sig2-sig1)       !factor
```

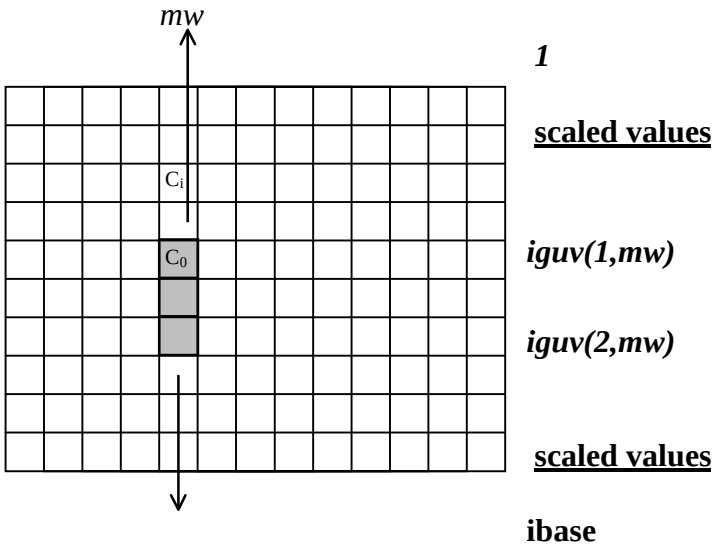
```
              do ind1=1,iNtrp_s(kont2-1,jgeo)  !cycle on to be interpolated values
                kont=kont+1
                iq=iQUALE(jgeo,kont)          !Absolute index of the current interpolated mw
                rFT1=(rMWsig(iq)-sig1)/(sig2-sig1)
                rCON(jgeo,iq)=VC1+(rMWsig(iq)-sig1)*rFT  !Continuum for the current interpolated
mw

! Note that for each interpolated value new row in rJACcon is pointed, but the same two columns
           iro = (iq-1)*ibase + Ngs + jgeo   !Row in rJACcon
            rJACcon(iro,ico1)= 1-rFT1     !derivative of interpolated continuum with respect the first
edge
            rJACcon(iro,ico2)= rFT1   !derivative of interpolated continuum with respect the second
edge
               end do
               kont=kont+1

          else
          end if

         if(iNtrp_s(kont2,jgeo).ne.0)then          !Next mw is interpolated

          else                                     !Next mw is not interpolated

          KONT2=KONT2+1

        ! Test on Close-Close condition: this further test is necessary to avoid a couple of CC mws is
        ! processed in this interpolation algorythm
          IF(LCCarr(jgeo,kont))then     ! Close-Close condition found
           kont=kont+2                  ! Skip the two mws
          else
           kont=kont+1
          end if

         end if
        end do       !End of Do While

        if(kont2.gt.iTOTmwcont(jgeo))then                    !test on the last value
           iq=iQUALE(jgeo,iCONTch(jgeo,iTOTmwcont(jgeo))) !last value of the Jacobian
           iro1 = (iq-1)*ibase + Ngs + jgeo                  !Row in rJAC
           ico1 = iCONTpr(jgeo,kont2 - 1)                    !Column in rJAC
           rJACcon(iro1,ico1)=1.
        else
           iq=iQUALE(jgeo,iCONTch(jgeo,iTOTmwcont(jgeo)))  !last value of the Jacobian
           iro1 = (iq-1)*ibase + Ngs + jgeo                  !Row in rJAC
           ico1 = iCONTpr(jgeo,kont2)                        !Column in rJAC
           rJACcon(iro1,ico1)=1.
        end if

      end do          !on geometries jgeo
```

4.2 Scaling in geometry of the continuum profiles

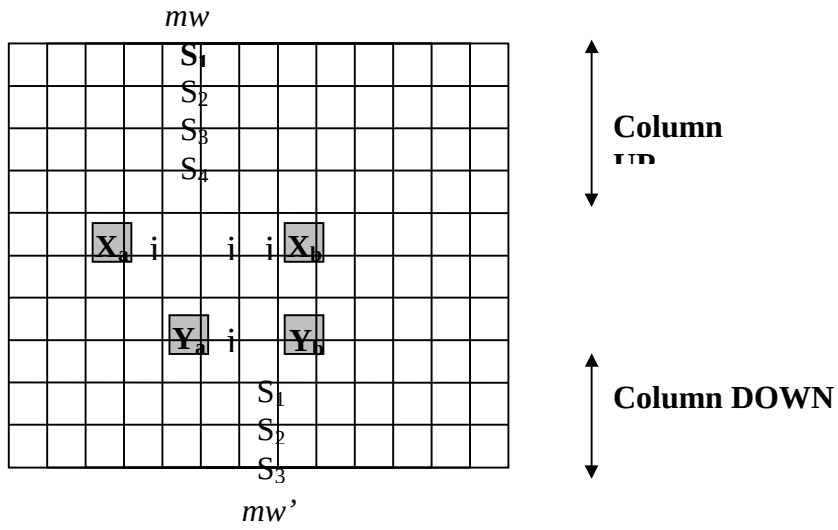
The continuum profiles, above the upper covered geometry (uppermost 'True' element in *lokku* matrix below *nucl*), i.e. for geometries from 1 to *iguv(1,mw)-1*, and below the lowest covered geometry (i.e. for geometries from 1 to *iguv(2,mw)*), are scaled by a factor equal to the ratio between the new value of these extreme elements and their old values. The new values are that contained in *rcpar* vector (and therefore also in *rcon* array), the old ones are that contained in *rcbase*. The scaled profiles will be computed by using the following expression:

$$C_i^{new} = \frac{C_0^{new}}{C_0^{old}} * C_i^{old}$$

where C_i^{new} is the new value for the continuum , C_i^{old} the old one, and C_0^{new} , C_0^{old} are respectively the new and the old value of the continuum at the upper or lower extreme geometry.
This scaling is computed, for the microwindow *mw*, from *iguv(1,mw)* up to the highest altitude and from *iguv(2,mw)* down to *ibase*.



The module that computes the derivatives of the continuum values can be inserted here. It can happen that if the extreme (upper or lower covered geometry) values of continuum are already interpolated ones in the frequency range, the derivatives of all elements of the scaled profiles are not zero if computed only with respect to the two parameters corresponding to the edge mws which have been used for the interpolation. This situation is shown in the following figure, where two columns of scaled values (above and below *iguv* for two different mws) of *rcbase* are shown as example.



In this case the derivative of the S_1 , for example, and that of all the scaled values will be not zero (and it has to be computed) only with respect to the two parameters $\mathbf{X_a}$ and $\mathbf{X_b}$,the continua related to the edge mws of a set of grouped mws. "i" indicates the interpolated values of the continuum; C_0 is also an interpolated value, so a composite derivative has to be calculate. Infact S_1 is scaled on the value of C_0 , but C_0 is a linearly interpolated value between $\mathbf{X_a}$ and $\mathbf{X_b}$. Therefore the expression for the derivative of the continuum S_1 in reference to all the other continua is different from zero only when is computed with respect to X_a and X_b and it reads:

$$\text{(A)} \quad \frac{dS_1}{dX_a} = \frac{dS_1}{dC_0^{new}} \frac{dC_0^{new}}{dX_a} = \frac{C(S_1)^{new}}{C_0^{new}} \left(1 - \frac{\sigma_0 - \sigma_a}{\sigma_b - \sigma_a}\right)$$

$$\text{(B)} \quad \frac{dS_1}{dX_a} = \frac{dS_1}{dC_0^{new}} \frac{dC_0^{new}}{dX_a} = \frac{C(S_1)^{new}}{C_0^{new}} \left(\frac{\sigma_0 - \sigma_a}{\sigma_b - \sigma_a}\right)$$

Where $C(S_1)^{new}$ is the updated value of the continuum (by scaling) at the quote 1, C_0^{new} is the new value of the continuum at the upper covered altitude (*iguv(1,mw)*), σ_0 is the central frequency of the microwindow "*mw*", σ_1 and σ_2 the central frequencies of the edge mws. Therefore the derivative is a product of two simple derivatives. Please note that the same argument can be easily applied to the computation of the derivative of scaled continua below the lowest covered geometry, i.e. below *iguv(2,mw).* Naturally, if neither upper nor lower extreme are already interpolated in frequency, the expression for the derivative will be given by the first of the two terms in Eq. (A).

The following code performs the scaling in geometry and the computation of the derivatives:

```
do ind_mw=1,nselmw                !loop on all the selected mws
   if(iguv(1,ind_mw).ne.0)then    !if this mw is used at some altitude then go in

 Column Up
     Nga=iguv(1,ind_mw)           !index of the current upper geometry
     rK0=rCON(Nga,ind_mw)         !value of the continuum at this upper geometry
   if (rcbase(Ngs+Nga,ind_mw).ne.0.) then    !Only if the continuum is not zero
```

```
  rMF=rCON(Nga,ind_mw)/rcbase(Ngs+Nga,ind_mw) !SCALING FACTOR  (new/old)
else                !continuum equal to zero in iguv(1,ind_mw)
  rmf = 0.          !put the scaling factor equal to zero
end if
 rcbase_old=rcbase(Ngs+Nga,ind_mw)
rcbase_old1=rcbase(Ngs+iguv(2,ind_mw),ind_mw)  !this is used after

do jgeot=1,Ngs+Nga          !rescaling of the continua above the upper quote
 rcbase(jgeot,ind_mw)=rcbase(jgeot,ind_mw)*rMF
end do
```

Derivatives UP

Derivatives of the continua above the upper geometry

Calling of the function *findpar_p* that returns (if they exist) the edge mws of grouped mws (here iextinfd, iextsupd), a logical value (here Lbif_do) which is true if a composite derivative has to be computed and icod1 and icod2 which are the columns of the rJACcon matrix (i.e. the indexes of the continuum parameters related to the edge mws)

```
call findpar_p(Nga,ind_mw,iQUALE,iTOTmwcont(Nga),iNtrp_s,
&   iCONTch,iCONTpr,icod1,icod2,Lbif_do,iextinfd,iextsupd)

 if(Lbif_do)then   !if the "column" above the upper geometry lays on an interpolated value
```

$$rRT = \frac{\sigma_0 - \sigma_a}{\sigma_b - \sigma_a}$$

```
 rRT=(rMWsig(ind_mw)-rMWsig(iQUALE(Nga,iCONTch(Nga,iextinfd))))/
&    (rMWsig(iQUALE(Nga,iCONTch(Nga,iextsupd)))-
&    rMWsig(iQUALE(Nga,iCONTch(Nga,iextinfd))))

! Computation of  rRS =  C_i new / C_0 new
do ind2=1,Ngs+Nga
 if (RCON(nga,ind_mw).ne.0.) then   !Only if C_0 new is not zero
   rRS=rcbase(ind2,ind_mw)/RCON(nga,ind_mw)
 else
   rRs=0.D0
 end if
 iro=(ind_mw-1)*ibase + ind2          !Row in the rJACcon matrix for this continuum value

 rJACcon(iro,icod1)= rRS * (1-rRT)  !composite derivative with respect to the first edge
 rJACcon(iro,icod2)= rRS * rRT      !composite derivative with respect to the second edge

end do

 else   !C_0 is not an already interpolated value

do ind2=1,Ngs+Nga
 if (rcon(Nga,ind_mw).ne.0.) then
```

```
 rRS=rcbase(ind2,ind_mw)/rcon(Nga,ind_mw)
 else
  rRS = 0.D0
 end if
 iro=(ind_mw-1)*ibase + ind2     !Row in the rJACcon matrix of this continuum value

 rJACcon(iro,icod1) =  rRS           !Simple derivative
end do

end if
```

Column Down

```
  Ngd=iguv(2,ind_mw)                    !index of the current upper geometry
  rcbase(Ngs+Ngd,ind_mw)=rcbase_old1
rK0=rCON(Ngd,ind_mw)        !value of the continuum at this lower geometry
  if (rcbase(Ngs+Ngd,ind_mw).ne.0.) then          !Only if C0 new is not zero
   rMF=rCON(Ngd,ind_mw)/rcbase(Ngs+Ngd,ind_mw) !SCALING FACTOR
  else
   rMF = 0.D0
  end if

rcbase_old=rcbase(Ngs+Ngd,ind_mw)

do jgeot=Ngs+Ngd,ibase   !rescaling of the continua below the lower quote
  rcbase(jgeot,ind_mw)=rcbase(jgeot,ind_mw)*rMF
end do
```

Derivatives Down

Derivatives of the continua below the lower geometry.

Calling of the function *findpar_p* that returns (if they exist) the edge mws of grouped mws (here iextinfu, iextsupu), a logical value (here Lbif_up) which is true if a composite derivative has to be computed and icou1 and icou2 which are the columns of the rJACcon matrix (i.e. the indexes of the continuum parameters related to the edge mws).

```
  call findpar_p(Ngd,ind_mw,iQUALE,iTOTmwcont(Ngd),iNtrp_s,
&   iCONTch,iCONTpr,icou1,icou2,Lbif_up,iextinfu,iextsupu)

 if(Lbif_up)then   !The C0 value is an already interpolated value
```

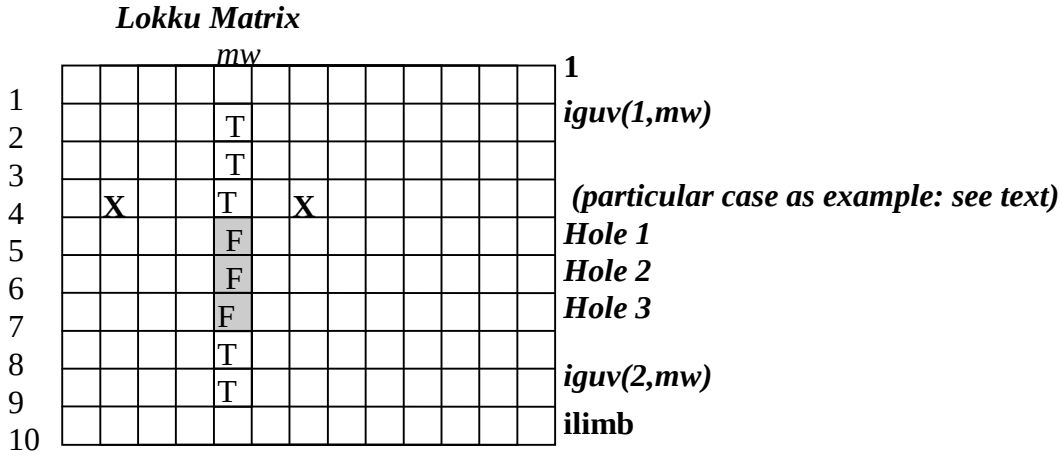
$$rRT = \frac{\sigma_0 - \sigma_a}{\sigma_b - \sigma_a}$$

```
rRT=(rMWsig(ind_mw)-rMWsig(iQUALE(Ngd,iCONTch(Ngd,iextinfu))))/
&    (rMWsig(iQUALE(Ngd,iCONTch(Ngd,iextsupu)))-
&    rMWsig(iQUALE(Ngd,iCONTch(Ngd,iextinfu))))
```

```
! Computation of   rRS = C_i new / C_0 new
do ind2=Ngs+Ngd,ibase
 if (RCON(Ngd,ind_mw).ne.0.) then
   rRS=rcbase(ind2,ind_mw)/RCON(Ngd,ind_mw) !rcbase_old
 else
   rRS = 0.D0
 end if

 iro=(ind_mw-1)*ibase + ind2       !Row in the rJACcon matrix of this continuum value

 rJACcon(iro,icou1)= rRS * (1-rRT) !composite derivative with respect to the first edge
 rJACcon(iro,icou2)= rRS * rRT     !composite derivative with respect to the second edge

end do

 else  !C_0 is not an already interpolated value

do ind2=Ngs+Ngd,ibase
 if (rcon(Ngd,ind_mw).ne.0.) then
  rRS=rcbase(ind2,ind_mw)/rcon(Ngd,ind_mw)
 else
  rrs = 0.D0
 end if
 iro=(ind_mw-1)*ibase + ind2    !Row in the rJACcon matrix of this continuum value

 rJACcon(iro,icou1) =  rRS    !Derivative
end do

end if  !Already interpolated or not

end if   !on iguv(1,.) ≠ 0

end do  !on all selected mws
```

4.3 Filling of the corrupted spectra (if any) with linear interpolated values in pressure domain.

This part of the code fills the 'holes' in the continuum profiles (corrupted spectra), if they exist. *mwcont* subroutine looks for the holes, i.e. it looks for 'False' values which break the continuity of the 'True' in the columns of *lokku* between *iguv(1, mw)* and *iguv(2, mw)*. If some hole exists, *mwcont* generates useful arrays which make possible to know how many holes have been found and where (*NmwHoled, NtotHoles( ), Nholes( ), HoleList(,)*). *ficarra* fills these holes by using linear interpolation in pressure range between the two nearest 'True' elements in *lokku*, just above and below the hole(s).



In this figure a column of *lokku* is shown where there are three holes at geometry 5, 6 and 7 . The continuum values, between which the linear interpolation in pressure range has to be performed, are those corresponding to geometry 4 and 8.

The value for the continuum in the corrupted spectrum can be simply computed as:

$$C_i = C_1 + \frac{C_2 - C_1}{p_2 - p_1} \cdot (p_i - p_1)$$

Where C_i is the continuum related to the holed mws, C_2 and C_1 are the continua related to the edge geometry between which the continuum has to be linearly interpolated (in this case Geom.4 and Geom.8), while "p_1" and "p_2" are the corresponding pressures to the quotes of the edge geometries (4 and 8).

As a particular case, the two edge continuum values (in this case 4 and 8) can be already linearly interpolated in frequency domain themselves. Therefore the conditions of composite derivative can occur. There are 4 cases: 1) No double derivative, 2) double derivative only for upper edge, 3) double derivative only for lower edge, 4) double derivative for both upper and lower edge. As a consequence the continuum parameters with respect to which we have to compute the derivatives will be the edges of the range of linear interpolation in *frequency*. In the figure shown above the particular case is the 2) : when the derivative of the continuum in the holes is computed , it will be not zero only in reference to the parameters X_a and X_b (the interpolated continuum values in frequency range *are not* parameters) as we have already seen in the case of the composite derivative for the scaled continua above and below the upper and lower geometry (see 4.2).

The expression for the derivative with respect to the upper and lower edge (geometries 4 and 8 in the previous example) will be:

1) In the case the two edge are not previously interpolated , the only not zero elements of the derivative of the continua in reference to all the parameters are:

$$\frac{dC_i}{dC_1} = (1 - \frac{p_i - p_1}{p_2 - p_1})$$
$$\frac{dC_i}{dC_2} = (\frac{p_i - p_1}{p_2 - p_1})$$

2) In the case only the upper edge is previously interpolated in frequency (as shown in the figure above) , the only not zero expressions for the derivative will be:

$$\frac{dC_i}{dX_a} = \frac{dC_i}{dC_1}\frac{dC_1}{dX_a} = (1 - \frac{p_i - p_1}{p_2 - p_1})(1 - \frac{\sigma_i - \sigma_a}{\sigma_b - \sigma_a})$$
$$\frac{dC_i}{dX_b} = \frac{dC_i}{dC_1}\frac{dC_1}{dX_b} = (1 - \frac{p_i - p_1}{p_2 - p_1})(\frac{\sigma_i - \sigma_a}{\sigma_b - \sigma_a})$$
$$\frac{dC_i}{dC_2} = (\frac{p_i - p_1}{p_2 - p_1})$$

3) In the case only the lower edge is previously interpolated in frequency, the only not zero expressions for the derivative will be (coherently with that shown above):

$$\frac{dC_i}{dC_1} = (\frac{p_i - p_1}{p_2 - p_1})$$
$$\frac{dC_i}{dY_a} = \frac{dC_i}{dC_2}\frac{dC_2}{dY_a} = (\frac{p_i - p_1}{p_2 - p_1})(1 - \frac{\sigma_i - \omega_a}{\omega_b - \omega_a})$$
$$\frac{dC_i}{dY_b} = \frac{dC_i}{dC_2}\frac{dC_2}{dY_b} = (\frac{p_i - p_1}{p_2 - p_1})(\frac{\sigma_i - \omega_a}{\omega_b - \omega_a})$$

4) In the case both the upper edge and the lower edge are previously interpolated in frequency, the only not zero expressions for the derivative will be:

$$\frac{dC_i}{dX_a} = \frac{dC_i}{dC_1}\frac{dC_1}{dX_a} = (1 - \frac{p_i - p_1}{p_2 - p_1})(1 - \frac{\sigma_i - \sigma_a}{\sigma_b - \sigma_a})$$
$$\frac{dC_i}{dX_b} = \frac{dC_i}{dC_1}\frac{dC_1}{dX_b} = (1 - \frac{p_i - p_1}{p_2 - p_1})(\frac{\sigma_i - \sigma_a}{\sigma_b - \sigma_a})$$
$$\frac{dC_i}{dY_a} = \frac{dC_i}{dC_2}\frac{dC_2}{dY_a} = (\frac{p_i - p_1}{p_2 - p_1})(1 - \frac{\sigma_i - \omega_a}{\omega_b - \omega_a})$$
$$\frac{dC_i}{dY_b} = \frac{dC_i}{dC_2}\frac{dC_2}{dY_b} = (\frac{p_i - p_1}{p_2 - p_1})(\frac{\sigma_i - \omega_a}{\omega_b - \omega_a})$$

Where X_a X_b and Y_a Y_b are the continuum parameters related to the edge mws (for the interpolation in frequency) respectively for the upper and the lower geometry . σ_a σ_b ω_a ω_b are the centres of these edge mws, while σ_i is the centre of the current mw , C_i and p_i are respectively the continuum

and the pressure of the current holed geometry. p_1 and p_2 are the pressure values corresponding to the edge geometries (4 and 8 in the example)

The following code performs the computation of the linearly interpolated values in pressure domain and of the derivatives with respect to all the parameters.

```
if(NtotHoles.gt.0)then              !If some hole exists in any mw
    Do ind_mw=1,NmwHoled            !DoLoop on the holed mw
      imwh=HoleList(ind_mw,1)       !What is the index of the holed mw?
      ind_hol=1

      Do while(ind_hol.le.Nholes(ind_mw))                !DoLoop on the holes of the current mw
         rk1=rCON(HoleList(ind_mw,1+ind_hol)-1,imwh)  !Continuum value of the first edge geometry
        igeo1 =HoleList(ind_mw,1+ind_hol)-1             !index of the first edge geometry

! Calling of the function findpar_p for igeo1 (the first edge geometry)
The function  returns (if they exist) the edge mws of grouped mws (here iextinfu , iextsupu), a logical value (here Lbif_up) which is true if a composite derivative has to be computed and icou1 and icou2 which are the columns of the rJACcon matrix (i.e. the indexes of the continuum parameters related to the edge mws).

        call findpar_p(igeo1,imwh,iQUALE,iTOTmwcont(igeo1),iNtrp_s,
   &         iCONTch,iCONTpr,icou1,icou2,Lbif_up,iextinfu,iextsupu)

       K=0
10      IF(HoleList(ind_mw,1+ind_hol+(K+1))             !counting of near holes
   &        - HoleList(ind_mw,1+ind_hol+K). EQ .1)then
          K=K+1
          GOTO 10
        else
        END IF
  rk2=rCON(HoleList(ind_mw,1+ind_hol+K)+1,imwh) !Continuum value of the second edge geometry

        igeo2 =HoleList(ind_mw,1+ind_hol+K)+1      !index of the first edge geometry

! Calling of the function findpar_p for igeo2 (the second edge geometry)
The function  returns (if they exist) the edge mws of  grouped mws (here iextinfd , iextsupd), a logical value (here Lbif_do) which is true if a composite derivative has to be computed and icod1 and icod2 which are the columns of the rJACcon matrix (i.e. the indexes of the continuum parameters related to the edge mws).

        call findpar_p(igeo2,imwh,iQUALE,iTOTmwcont(igeo2),iNtrp_s,
   &         iCONTch,iCONTpr,icod1,icod2,LBif_do,iextinfd,iextsupd)


!Computing of the derivatives ( rJACcon  elements)
```

```
rlC1= rk1      ! Now that the linear interpolation in  pressure substituted
rlC2= rk2      ! the exp. interpolation in altitude range, it is no more necessary
               ! to pass in the semilogarithmic space (see old versions)
```

! Computation of FAT $= \dfrac{C_2 - C_1}{p_2 - p_1}$

```
rdif_1=1.D0/(rQT(igeo2)-rQT(igeo1))
FAT=(rlC2-rlC1)*rdif_1

Do imd=1,K+1                                  !cycle on the holes
  igeo3=HoleList(ind_mw,1+ind_hol+(imd-1))    !geometry of the current hole
```

!Calculation of rRS = $\left(\dfrac{p_i - p_1}{p_2 - p_1}\right)$

```
rRS=(rQT(igeo3)-rQT(igeo1))*rdif_1
```

! Setting of the value of the continuum corresponding at the current hole

!calculation of $C_i = C_1 + \dfrac{C_2 - C_1}{p_2 - p_1} \cdot (p_i - p_1)$

```
rCON(HoleList(ind_mw,1+ind_hol+(imd-1)),imwh)=
&          (rQT(igeo3)-rQT(igeo1))*FAT+rlC1

iro=(imwh-1)*ibase+Ngs+Holelist(ind_mw,1+ind_hol+(imd-1))   ! row in rJACcon
```

!This is the test to verify if one (or both) of the edge geometries is already interpolated in the frequency range. The test uses the value of the logical variables *Lbif_up* and *Lbif_do*:

CASE 1. Neither C_1 nor C_2 already interpolated in frequency

```
if((.NOT.LBif_up).and.(.NOT.LBif_do)) then  !No previously interpolated or Close-Close

   rJACcon(iro,icou1) = (1-rRS)   !simple derivative with respect to  the upper edge
   rJACcon(iro,icod1) =  rRS      !simple derivative with respect to  the lower edge
```

CASE 3. C_2 Already interpolate in frequency

```
elseif((.NOT.LBif_up) .and. LBif_do) then
```

!Computation of rRT $= \dfrac{\sigma_i - \omega_a}{\omega_b - \omega_a}$

```
 rRT=(rMWsig(imwh)-rMWsig(iQUALE(igeo2,iCONTch(igeo2,iextinfd))))/
&          (rMWsig(iQUALE(igeo2,iCONTch(igeo2,iextsupd)))-
&          rMWsig(iQUALE(igeo2,iCONTch(igeo2,iextinfd))))

           rJACcon(iro,icou1) = (1-rRS)
           rJACcon(iro,icod1)=  rRS * (1-rRT)
           rJACcon(iro,icod2)= rRS * rRT
```

CASE 2. C_1 Already interpolate in frequency

```
        elseif( LBif_up .and. (.NOT.LBif_do)) then
```

$$\text{!Computation of rRT} = \frac{\sigma_i - \sigma_a}{\sigma_b - \sigma_a}$$

```
 rRT=(rMWsig(imwh)-rMWsig(iQUALE(igeo1,iCONTch(igeo1,iextinfu))))/
&          (rMWsig(iQUALE(igeo1,iCONTch(igeo1,iextsupu)))-
&          rMWsig(iQUALE(igeo1,iCONTch(igeo1,iextinfu))))

           rJACcon(iro,icou1)= (1-rRS) * (1-rRT)
           rJACcon(iro,icou2)= (1-rRS)* rRT
           rJACcon(iro,icod1)= rRS
```

CASE 4. Both C_1 and C_2 already interpolated in frequency

```
    else( LBif_up .and. Lbif_do ) then
```

$$\text{!Computation of rRT} = \frac{\sigma_i - \sigma_a}{\sigma_b - \sigma_a}$$

```
 rRT=(rMWsig(imwh)-rMWsig(iQUALE(igeo1,iCONTch(igeo1,iextinfu))))/
&          (rMWsig(iQUALE(igeo1,iCONTch(igeo1,iextsupu)))-
&          rMWsig(iQUALE(igeo1,iCONTch(igeo1,iextinfu))))
```

$$\text{!Computation of rRTd} = \frac{\sigma_i - \omega_a}{\omega_b - \omega_a}$$

```
 rRTd=(rMWsig(imwh)-rMWsig(iQUALE(igeo2,iCONTch(igeo2,iextinfd))))/
&    (rMWsig(iQUALE(igeo2,iCONTch(igeo2,iextsupd)))-
&    rMWsig(iQUALE(igeo2,iCONTch(igeo2,iextinfd))))

           rJACcon(iro,icou1)= (1-rRS) * (1-rRT)
           rJACcon(iro,icou2)= (1-rRS)* rRT
           rJACcon(iro,icod1)= rRS* (1-rRTd)
           rJACcon(iro,icod2)= rRS* rRTd

        end if
```

```
        end do          !on a group of near Holes

        ind_hol=ind_hol+1+K

      end do            !on the holes in mw

    end do              !on the holed mws

    end if              !if on the existence of an hole
```

5. Copying of *rcon* in *rcbase* array

At the end of the interpolations, we need to copy all the continuum values of *rcon* into *rcbase* for each mw, from *iguv(1,mw)* to *iguv(2,mw).* This ensures that at the end of *ficarra* routine the *rcbase* array contains the updated continuum profiles. The following code does this work:

```
      do ind_mw=1,nselmw
       if(iguv(1,ind_mw).ne.0)then
         do jgeo=iguv(1,ind_mw),iguv(2,ind_mw)
           rcbase(ibase-1-ilimb+jgeo,ind_mw)=rCON(jgeo,ind_mw)
         end do
       end if
      end do
```

6. The function *findpar_p*

The function *findpar_p* reads the current geometry *kgeo*, the current microwindow *mw* and, from the variables and array *iquale, iTOTmwco,iNtrp_s,iCONTch,iCONTpr* finds out if the current continuum (kgeo,mw) is a parameter or not and, if it is a to be interpolated value, it returns the local index of the edge (in frequency range) mws (*iextinf , iextsup)* and the index of the continuum parameters corresponding to these edge mws (ico1 , ico2). A logical value *Lbif* (set to TRUE if the continuum has to be interpolated in the frequancy range) is also returned. The code of the subroutine follows:

```
    subroutine findpar_p(kgeo,imw,
    &  iQUALE,iTOTmwco,iNtrp_s, iCONTch,iCONTpr,
    &  ico1,ico2,Lbif,iextinf,iextsup)

     LBif=.False.
     iSloc=0

    !DoLoop on all the mws chosen as parameter for this geometry
     Do jj=1,iTOTmwco
          iactmw=iQUALE(KGEO,(iCONTch(KGEO,jj))) !absolute index of current mw

         !True if the current mw imw is one of that chosen as parameter
```

```
 if(iactmw.eq.imw)then
   iSloc=jj
 end if

!This test looks for the last mw before the current mw
 if(iactmw.lt.imw)then
   iextinf=jj
end if

!This test looks for the first mw immediatly after the current mw
if(iQUALE(KGEO,(iCONTch(KGEO,iTOTmwco-jj+1))).gt.imw)then
   iextsup=iTOTmwco-jj+1
end if
 end do
```

!if one of the chosen as parameter mws (at this geomtery) was found with its absolute index equal to that of the current mw, then the continuum related to the current mw is a parameter for this geometry.

```
if(iSloc.ne.0)then
 ico1=iCONTpr(KGEO,iSloc)
 ico2=0
else
```

!else the continuum related to the current mw *is not* a parameter for this geometry. So the continuum value has to be interpolated between the nearest parameters immediatly before and after that related to the current mw (at this geometry). There is another possibility: the current mw is a Close-Close mw: it is not a parameter (only its parent mw is it), but it hasn't to be interpolated. So we have to check that the value $iNtrp\_s(\ iextinf\ ,KGEO) \neq 0$; this ensures the mw is a real to be interpolated mw.

```
 if(iNtrp_s( iextinf ,KGEO).eq.0)then
   LBif=.False.                        !Logical flag
   ico1=iCONTpr(KGEO,iextinf) !index of the parameter related to this mw
   ico2=0
 else
   LBif=.True.                         !Logical flag
     ico1=iCONTpr(KGEO,iextinf)    !index of the parameter related to the lower edge in
frequency
     ico2=iCONTpr(KGEO,iextsup)   !index of the parameter related to the upper edge in
frequency
 end if
end if

 return
 end
```

2.2.28 JACLOSCALC

Description
This module calculates the jacobian matrix which connects LOS engineering data with the unknowns of the retrieval. This matrix corresponds to matrix $\mathbf{K}_1$ defined in Sect. 4.2.6 of AD6.

Variables exchanged with external modules

Name	Description
rxpar	rxpar(imxtop) = vector of the unknown parameters
ilimb	n. of considered sweeps
ipar	n. of points fitted in the T profile
itop	total n. of fitted parameters
rztang	rztang(imxgeo) = tangent altitudes of the considered sweeps (km)
rlat	latitude (deg.)
rjaclos	rjaclos(imxlmb,imxtop) = jacobian matrix of LOS data

Detailed description

- Please refer to Sect. 4.2.6.1 of the Algorithms document (AD6) in order to understand the calculation of this jacobian matrix.

```
* Initialisation:
    do j=1,imxtop
     do i=1,imxlmb
       rjaclos(i,j) = 0.0d0
     end do
    end do

    do i=1,ilimb-1                          ! Begin the loop on the rows of the jacobian

* setup of some quantities:
      rzav = (rztang(i)+rztang(i+1))/2.     ! mean altitude of the layer
      rgamma = 2.*gravity(rzav,rlat)*rmovr  ! quantity 2*'gamma' in AD6
      r1 =  (rxpar(ilimb+i+1)+rxpar(ilimb+i))/rgamma
      r2 =  log(rxpar(i)/rxpar(i+1))/rgamma

* derivatives with respect to pressure:
      rjaclos(i,i) = r1 / rxpar(i)
      rjaclos(i,i+1) = - r1 / rxpar(i+1)

* derivatives with respect to temperature:
      rjaclos(i,ilimb+i) = r2
      rjaclos(i,ilimb+i+1) = r2

    end do                                  ! end loop on the rows of the jacobian
    end
```

2.2.29 VC_HEIGHTCORR

Description
Starting from the vector of the retrieved parameters and the related VCM, this module determines the vector of the corrections to the engineering tangent heights and the related VCM. The derived quantities are not used in the program but are only to be reported in Level 2 products.

Variables exchanged with external modules

Name	Description
rxpar	rxpar(imxtop) = vector of the unknown parameters
ilimb	n. of considered sweeps
ipar	n. of points fitted in the T profile
rainv	rainv(imxtop,imxtop) = VCM of the retrieved parameters
rztang	rztang(imxgeo) = tangent altitudes of the considered sweeps (km)
rztanginit	rztanginit(imxgeo) = initial (=engineering) tangent altitudes of the considered sweeps (km)
rhcorr	rhcorr(imxlmb) = vector of the corrections to be applied to to engineering tangent altitudes (ilimb-1 elements)
rvchcorr	rvchcorr(imxlmb,imxlmb) = VCM of rhcorr

Detailed description

- Please refer to Sect. 4.2.7 of AD6 in order to understand the theory which is behind this procedure.

* Building of the jacobian matrix (rkd) which connects the height corrections with the unknowns of p,T retrieval:
*

```
      do i=1,ilimb-1     !  Begin loop on the height corrections
       do k=1,ilimb       ! Begin loop on tangent pressures (first set of parameters)
        r4 = 0.
        do j=i+1,ilimb              ! Begin loop for summation
         r1 = (rxpar(ilimb+j)+rxpar(ilimb+j-1))/(2.*rmovr)
         r2 = 0.
         r3 = 0.
         if (j.eq.k) r2=1./rxpar(j)
         if (j-1.eq.k) r3=1./rxpar(j-1)
         r4=r4+r1*(r2-r3)
        end do              ! End loop for summation
        rkd(i,k) = r4
       end do               ! End loop on tangent pressures
       do k=1,ipar        ! Begin loop on tangent temperatures (second set of parameters)
        r4 = 0.
        do j=i+1,ilimb   ! Begin loop for summation
         r1=log(rxpar(j)/rxpar(j-1))/(2.*rmovr)
         r2 = 0.
```

```
          r3 = 0.
          if (j.eq.k) r2 = 1.
          if (j-1.eq.k) r3 = 1.
          r4 = r4 + r1 * (r2+r3)
        end do          ! End loop for summation
        rkd(i,k+ilimb) = r4
      end do            ! End loop on tangent temperatures
    end do            !  End loop on the height corrections
*
* End of calculation of the jacobian matrix of the height corrections
```

- Now the VC matrix of the height corrections (rvchcorr) is obtained by transforming the VCM related to p,T retrieved data (rainv) accordingly to rvchcorr = rkd * rainv * $(rkd)^T$.

A - Multiplication rkd * rainv, the result is stored in the scratch matrix rscr(imxlmb,2*imxlmb). Only the components of rainv related to p and T parameters must be onsidered.

```
    do i=1,ilimb-1
      do j=1,ilimb+ipar
        r1 = 0.
        do k=1,ilimb+ipar
          r1 = r1 + rkd(i,k)*rainv(k,j)
        end do
        rscr(i,j) = r1
      end do
    end do
* B - Multiplication rvchcorr = rscr * (rkd)^T
    do i=1,ilimb-1
      do j=1,ilimb-1
        r1 = 0.
        do k=1,ilimb+ipar
          r1 = r1 + rscr(i,k)*rkd(j,k)
        end do
        rvchcorr(i,j) = r1
      end do
    end do
* End of calculation of the VC matrix of the height corrections

* Calculation of the height corrections (in km):
     do i=1,ilimb-1
       rhcorr(i)=rztang(i)-rztanginit(i)
     end do

     end
```

2.2.30 READ_IRRGRID_PT

Description

This module, which is called by the **retr_pt** module only if *lirrgrid* is true, is used for:

- reading the files containing the irregular grids in the hexadecimal representation,
- rebuilding the grid in the binary representation,
- calculating some variables used in routine spectrum_ for the direct interpolation / convolution of the spectra.

Variables exchanged with external modules

Name	Description
lirrgridmw	logical: lirrgridmw(imxmw): logical vector that, for each selected microwindow in the actual retrieval, indicates whether the irregular grid is available.
smw	character*6: smw(imxmw): vector containing the identifier label of the selected microwindows
nselmw	integer*4: total number of selected microwindows
dsigma	real*8: dsigma(imxsig, imxmw): wavenumber fine grid for each microwindow [cm^{-1}]
isigma	integer*4: isigma(imxmw): number of general wavenumber fine grid points in each microwindow
delta	real*8: distance between fine-wavenumber grid points [cm^{-1}]
nrd	integer*4: ratio between the frequency steps of the coarse and the fine grid
igeo	integer*4: total number of simulated limb views
iigrid	integer*4: *iigrid(imxsig,imxgeo,imxmw)*: irregular grid in the '0' and '1' representation for all the fine grid points of the extended microwindow *imw*.
cint	character*3: cint(imxmw): it indicates, for each microwindow, what kind of interpolation has to be performed between the spectral points of the irregular grid.
igridc	integer*4: igridc(imxsi2,imxmw): matrix that, to each microwindow and each point of the compressed grid, associates the corresponding value on the regular fine grid.
nused1	integer*4: nused1(imxmw): total number of points of the compressed grid for each microwindow
rsan	real*8: rsan(imxi,imxsi2,4,imxmw): variable used for making the direct interpolation/convolution of the spectra. *rsan(jsam,i,n,imw)=* $\sum_{j=\max(((jsam-1)*nrd+1),igridc(i,imw))-(jsam-1)*nrd,k=\max(0,-igridc(i,imw)+((jsam-1)\cdot nrd+1)}^{j=\min(igridc(i+1,imw)-1,nils-1+(jsam-1)*nrd)-(jsam-1)*nrd,k=\min(igridc(i+1)-1-igridc(i),((jsam-1)\cdot nrd+nils-igridc(i))} rils(nils-j+1)\cdot k^{n-1}$

ilim	integer*4: ilim(2,imxi,imxmw): variable used for making the direct interpolation/convolution : *ilim(1,jsam,imw):* first point of the compressed grid to be considered for the computation of the low resolution spectral point at *jsam* for microwindow *imw*; *ilim(2,jsam,imw):* total number of points of the compressed grid to be considered for the computation of the low resolution spectral point at *jsam* for microwindow *imw*.logical: *lirrgridmw(imxmw):* logical vector that, for each selected microwindow in the actual retrieval, indicates whether the irregular grid is available.
rils	real*8: rils(imxils,imxmw: instrument-line-shape function in the frequency fine-grid
nils	integer*4: number of elements of rils
nsam	integer*4: nsam(imxmw): number of sampling points in each MW (general coarse grid)

Module structure

1. Reading of the location of the directory containing irregular grids
Begin loop 1 over microwindows: imw=1→ nselmw
2. Initialisation of some variables
Begin condition 1: for the considered microwindow an irregular grid is available
3. Reading of the file containing the irregular grid for the considered microwindow, checks on the read quantities and conversion of the irregular grid from hexadecimal to binary representation .
4. Definition of the array *iigrid*, containing the irregular grid in the binary representation and computation of vector *igridc*, which associates to each '1' value of the irregular grid the corresponding position on the regular fine grid.
5. Mirroring of rils and initialisation of rsan
Begin loop 2 over the points of the local coarse grid
Begin loop 3 over the points of the compressed grid
6. Setting of some variables for the computation of *rsan*
7. Computation of rsan(imxi,imxsi2,4,imxmw) and ilim(2,imxi,imxmw)
End loop 3
End loop 2
End condition 1
End loop 1

Detailed description

1. The location of the directory containing irregular grids is read from the environment variable ORM_IRRGRID.

```
call getenv ("ORM_IRRGRID",siod)
iodl = index(siod," ")-1   ! computes the length of siod
if (iodl.gt.80) then
  write(*,*)' --- FATAL ERROR in main orm ---'
  write(*,*)'Path of the I/O directories too long !'
  stop
```

```
end if
if (siod(iodl:iodl).ne.'/') then
  write(*,*)' --- FATAL ERROR in main orm ---'
  write(*,*)'The environment variable ORM_IRRGRID'
  write(*,*)'must end by / '
  stop
end if
```

Loop 1 over microwindows:
imw=1→nselmw

2. Initialisation of some variables

The arrays *iigrid (imxsig,imxmw)* and *igridc(imxsi2,imxmw)* are initialised to 0.

```
do i=1,imxsig
   iigrid(i,imw)=0
end do

do k=1,imxsi2
   igridc(k,imw)=0
end do
```

The two kinds of interpolation foreseen by the program are contained in the character vector *cinterp.*

```
data cinterp/'lin','cub'/
```

Condition 1
If *lirrgridmw(imw)* is true, then operations 3, 4, 5, 6 and 7 are performed, otherwise the subsequent microwindow is considered.

3.Reading of the file containing the irregular grid for the considered microwindow, checks on the read
quantities and conversion from hexadecimal to binary representation of the irregular grid.

The ascii file containing the irregular grid for the microwindow *smw(imw)* (*'irrgrid_'//smw(imw)//'.dat'*) is opened and read:
For the description of the format of this file, please refer to [RD3].

```
open(99,file=sidir(1:iodl)//'irrgrid_'//smw(imw)//'.dat', status='old')
```

A blanck line is located at the beginning of this file.

```
read(99,*)cspare
```

An arbitrary number of comment lines, starting with '!', can be found at the beginning of this file:

```
100  read (99,*) commnt
     if (commnt .eq. '!') goto 100
     backspace (99)
```

Reading of the label indicating the kind of interpolation to be performed:

```
read(99,*) cint(imw)
```

Check on this label: if this label is different of the ones foreseen by the program (contained in variable *cinterp(2)*), a warning message is written, the logical variable *lirrgridmw(imw)* is set to false and the program skips to consider the next microwindow.

```
      if((cint(imw).ne. cinterp(1)) .and.
&     (cint(imw).ne. cinterp(2)) ) then
   write(*,*) 'An interpolation different of the ones'
   write(*,*) 'that had been foreseen '
   write(*,*) 'has to be performed'
   write(*,*) 'between the points of the irregular grid.'
   write(*,*) 'This kind of interpolation has not been'
   write(*,*) 'anticipated: please improve the program.'
      lirrgridmw(imw)=.false.
      goto [end loop over microwindows]
  end if
```

Reading of the total number of fine grid points *ntot*, the number of points used in the irregular grid *nused*, the frequency of the first point *rwn0*, the frequency step *rdeltasigma*.

```
      read(99,*) ntot, nused, rwn0, rdeltasigma
```

Check whether the frequency step *rdeltasigma* is consistent with the frequency step used in the retrieval program *delta*; if the two values are not consistent, the logical variable *lirrgridmw(imw)* is set to false and the program skips to consider next microwindow.

```
      if(abs(rdeltasigma-delta).gt.1.d-8) then
       lirrgridmw(imw)=.false.
        close(99)
       go to [end loop on microwindows]
      endif
```

Check on the starting frequency of the irregular grid: calculation of the points *istart* of the irregular grid to be skipped (this number will be different of 0 only if the irregular grid is built for a wider interval than the extended microwindow). On the contrary, if the irregular grid is build for a smaller interval than the extended microwindow, the logical variable *lirrgridmw(imw)* will be set to false and the program will skip to consider next microwindow.

```
      xistart=(dsigma(1,imw)-dble(rwn0))/dble(rdeltasigma)
      istart=int(xistart)
      if(istart .lt.0) then
       lirrgridmw(imw)=.false.
       close(99)
       go to [end  loop on microwindows]
      end if
```

Reading of the altitude range of the irregular grid:

```
      read(99,*) rangedown, rangeup
```

Reading of the irregular grid in the hexadecimal representation, conversion of the grid in binary representation and storing of the grid in the vector *igrid(imxsig).*
Each line of the file is made of a maximum of 50 hexadecimal numbers, and each number corresponds to 4 points in the binary representation of the frequency grid. Therefore each line corresponds to 200 points.
The number of lines *jmax* to be read depends on the total number of points *ntot* of the irregular grid, and it is equal to [int(ntot/200) + 1].
The last line is made of : kmax=(real(ntot)/200.-int(ntot/200.))*50+1 hexadecimal numbers.

After the reading of each line, each hexadecimal number is converted in binary number by the module **hex_bin**.
The binary numbers are stored in the vector *igrid(imxsig).*

```
      jmax=int(ntot/200.)+1
      no1=0
     do 30 j=1,jmax
      read(99,'(a50)')cirrgrid
      kmax=50
      if(j .eq. jmax)  kmax=(real(ntot)/200.-int(ntot/200.))*50+1
      do 40 k=1,kmax
       call hex_bin(cirrgrid(k:k),n1,n2,n3,n4)

      igrid((j-1)*200+(k-1)*4+1)=n1
      igrid((j-1)*200+(k-1)*4+2)=n2
      igrid((j-1)*200+(k-1)*4+3)=n3
      igrid((j-1)*200+(k-1)*4+4)=n4
      no1=no1+n1+n2+n3+n4
40     continue
30     continue
```

4. Definition of the array *iigrid*, containing the irregular grid in the binary representation and computation of vector *igridc*, which associates to each '1' value of the irregular grid the corresponding position on the regular fine grid.

The first and the last point of the irregular grid, in the case of linear interpolation, and the first and last two points of the irregular grid, in the case of cubic interpolation, are set to '1'.

The irregular grid is stored in vector *iigrid(1→isigma(imw), imw).*
The use of this additional matrix, other than *igrid(imxsig)*, makes possible for the program to handle irregular grids that have been computed for frequency intervals wider than the extended microwindows.

Besides, for subsequent calculations it is useful to compute the vector *igridc(imxsi2)* that, to each '1' in the irregular grid, associates the corresponding number in the regular fine grid.

```
      i=0
```

```
      do ksig=1,isigma(imw)
        ksig1=istart+ksig
         if ((ksig .eq. 1  .or. ksig .eq. isigma(imw)) .or. (cint(imw).eq.'cub' .and.
     &          (ksig .eq. 2  .or. ksig .eq. (isigma(imw)-1))))then
          iigrid(ksig,imw,1)=1
         else
          iigrid(ksig,imw,1)=igrid(ksig1)
         end if
        if (iigrid(ksig,imw,1).eq.1)then
          i=i+1
          igridc(i)=ksig
        endif
      end do
      nused1(imw)=i
```

5.-6.-7. Computation of variables used by spectrum_pt_ for performing the direct interpolation / convolution

In order to understand the meaning of the subsequent computations, it is necessary to explain how the low resolution spectrum is computed when irregular grids are available. The low resolution spectrum is calculated by the routine spectrum_pt, but when irregular grids are available, some preliminary computations which depends only on the irregular grid and on the AILS function, are performed in this routine.
The low resolution spectrum $rspct(1 \to nsam(imw))$ is the result of the convolution between the high resolution spectrum $rsp(1 \to isigma(imw))$ and the high resolution AILS function $rils(1 \to nils)$, performed only in correspondence of the coarse grid points. (We have not reported the dependence of the spectrum on the microwindow and the geometry and the dependence of the AILS function on the microwindow).

In particular, the value of the low resolution spectrum corresponding to a given point of the frequency coarse grid *jsam* is given by:

$$rspct(jsam) = \sum_{ii=1}^{nils} rsp((jsam-1)\cdot nrd + ii)\cdot rils(nils - ii + 1),$$

nrd is the ratio between the frequency step of the coarse and fine grid.

When an irregular grid is available, only the spectral points corresponding to the '1' points of the irregular grid, i.e. the points of the 'compressed' grid, are computed using the Radiative Transfer equation, the others, i.e. the spectral points corresponding to the '0' points of the irregular grid, are computed performing an interpolation between the values of the spectrum computed on the compressed grid.
In the case of linear interpolation, the interpolated value of the spectrum corresponding to the point *jj* of the regular fine grid is given by:

$$(1) \qquad rsp(jj) = rsp(i) + \frac{rsp(i+1) - rsp(i)}{igridc(i+1) - igridc(i)}\cdot k = rsp(i) + m\cdot k\,;$$

where $k = jj - igridc(i)$ assumes values from *0* to *(igridc(i+1)-igridc(i))*,

while in the case of cubic interpolation, it is given by:

(2) $rsp(jj) = rsp(i) + a \cdot k^3 + b \cdot k^2 + c \cdot k$;

where *a, b, c* are the coefficients of the cubic interpolation dependent on *rsp(i), rsp(i-1), rsp(i+1), rsp(i+2).*

When irregular grids are available, instead of performing first the interpolation in the regular fine grid and then the convolution with the AILS in correspondence of only the coarse grid points, it is possible to save time computing both operations at the same time.
Merging equs. 1 and 2 we find that the contribution to the value of $rspct(jsam)$ given by all the points of the regular fine grid between two near points of the compressed grid (*i* and *i+1*) is equal, in the case of linear interpolation, to:

$$rsp(i) \cdot \sum_{j=igridc(i)-(jsam-1)\cdot nrd,k=0}^{igridc(i+1)-1-(jsam-1)\cdot nrd,k=igridc(i+1)-1-igridc(i)} rils(nils - j + 1) + m \cdot \sum_{j=igridc(i)-(jsam-1)\cdot nrd,,k=0}^{igridc(i+1)-1-(jsam-1)\cdot nrd,k=igridc(i+1)-1-igridc(i)} rils(nils - j + 1) \cdot k$$

and, in the case of cubic interpolation, to:

$$rsp(i) \cdot \sum_{j=igridc(i)-(jsam-1)\cdot nrd,k=0}^{igridc(i+1)-1-(jsam-1)\cdot nrd,k=igridc(i+1)-1-igridc(i),} rils(nils - j + 1) + a \cdot \sum_{j=igridc(i)-(jsam-1)\cdot nrd,k=0}^{igridc(i+1)-1-(jsam-1)\cdot nrd,k=igridc(i+1)-1-igridc(i)} rils(nils - j + 1) \cdot k^3 +$$

$$+b \cdot \sum_{j=igridc(i)-(jsam-1)\cdot nrd,,k=0}^{igridc(i+1)-1-(jsam-1)\cdot nrd,k=igridc(i+1)-1-igridc(i)} rils(nils - j + 1) \cdot k^2 + c \cdot \sum_{j=igridc(i)-(jsam-1)\cdot nrd,k=0}^{j=igridc(i+1)-1-(jsam-1)\cdot nrd,k=igridc(i+1)-1-igridc(i)} rils(nils - j + 1) \cdot k$$

This routine computes, for each point *jsam* of the frequency coarse grid and for each point *i* in the compressed grid which affects the value of convolved spectrum at *jsam,* the matrix *rsan(imxi,imxsi2,4,imxmw)* whose elements are equal to (particular cases that will be considered later):

(3) $$rsan(jsam,i,n,imw) = \sum_{j=igridc(i)-(jsam-1)\cdot nrd,k=0}^{igridc(i+1)-1-(jsam-1)\cdot nrd,k=igridc(i+1)-1-igridc(i)} rils(nils - j + 1) \cdot k^{n-1},$$

where $1 \le n \le 4$.

The computation of matrix *rsan* is computed as follows:

5. Mirroring of *rils* and initialisation of *rsan*

In order to simplify and save time in the subsequent computations, the local vector *rils1(imxils*) is computed just mirroring with respect to the central element the vector containing the AILS function *rils(j=1→nils,imw).*

```
do 35 j=1,nils
    rils1(j)=rils(nils-j+1,imw)
35  continue
```

Matrix *rsan* is initialised

```
do jsam=1,nsam(imw)
   do i=1,nused1(imw)
    do k=1,4
     rsan(jsam,i,k,imw)=0.0d0
    end do
   end do
 end do
```

Begin loop 2 over the points of the local coarse grid
jsam= 1, nsam(imw)

The variable *ilim(2,jsam,imw),* that indicates the number of points of the compressed grid that have to be taken into account for the computation of the value of the low resolution spectrum at *jsam* for microwindow *imw*, is initialised to 0.

ilim(2, jsam, imw)=0

Begin loop 3 over the points of the compressed grid

i=iin,ifin

iin and *ifin* are respectively 1 and *nused1(imw)-1* in the case of linear interpolation and are 2 and *nused1(imw)-2* in the case of cubic interpolation:

```
if (cint(imw).eq. 'cub')then
   iin=2
   ifin=nused1(imw)-2
 else
   iin=1
   ifin=nused1(imw)-1
 end if
```

For each *i*, a check is done for evaluating if the elements of the matrix *rsan(jsam, i, n,imw), n=1→4* have to be computed.
The logical variable *li* indicates whether this test was succesful (li equal to true) or not.
At the beginning *li* is set to false.

Since the convolution of the spectrum with the AILS function in correspondence of the coarse grid point *jsam* has to be performed considering the points of the spectrum on the regular fine grid between $\left(\left(jsam-1\right)\cdot nrd+1\right)$ and $\left(\left(jsam-1\right)\cdot nrd+nils\right)$,
the point *i* of the compressed grid has to be taken into account if the following condition is verified:

(*) $$\left(\left(jsam-1\right)\cdot nrd+1\right)\le igridc\left(i,imw\right)\le\left(\left(jsam-1\right)\cdot nrd+nils\right)$$

When this condition is verified, the variable *k*, which is the second factor in the expression of rsan, is set to 0 and *li* is set to true.

If condition (*) is not verified, but the following condition si verified:

(**) $$igridc\left(i+1,imw\right)>\left(\left(jsam-1\right)\cdot nrd+1\right)$$

it means that the interval between $\left(\left(jsam-1\right)\cdot nrd+1\right)$ and $\left(\left(jsam-1\right)\cdot nrd+nils\right)$ includes only a portion of the interval of the fine grid from *igridc(i,imw)* to *igridc(i+1,imw).*
In this case the point *i* has to be taken into account, but the starting value of k is equal to:
$\left(\left(jsam-1\right)\cdot nrd+1\right)-igridc(i,imw)$.

```
      li=.false.

      if (igridc(i,imw) .ge. ((jsam-1)*nrd +1) .and.
     &    igridc(i,imw) .le. ((jsam-1)*nrd +nils)) then

      k=0
      li=.true.
     else
      if(igridc(i+1,imw) .gt. (jsam-1)*nrd+1)then
        k=((jsam-1)*nrd+1)-igridc(i,imw)
        li=.true.
      end if
     end if
```

Operations 6. and 7. are performed only if *li* is true.

6. Setting of some variables for the computation of *rsan*

The limits within which the index of the *rils1* vector has to vary in the summations which define the matrix *rsan* are computed.
These limits are computed taking into account also the possibility that only a portion of the regular grid interval between two near points of the compressed grid is contained in the interval between $\left(\left(jsam-1\right)\cdot nrd+1\right)$ and $\left(\left(jsam-1\right)\cdot nrd+nils\right)$.

In order to avoid considering the same element of *rils* and *rsp* twice, we consider, for each *i*, all the points between *igridc(i)* and *(igridc(i+1)-1).* The only exception is when *jjf* is equal to *(nils-1),* or in the case of cubic interpolation, when *jjf* is equal to *(nils-2)* and *i* is equal to *(nused1(imw)-2).*
In the case of cubic interpolation, the first and the last point of the spectrum on the regular fine grid is not taken into account. They will be taken into account in routine spectrum_pt.

```
      jji=max(((jsam-1)*nrd+1),igridc(i,imw))-
&          (jsam-1)*nrd
      jjf=min(igridc(i+1,imw)-1,nils-1+(jsam-1)*nrd)-
&          (jsam-1)*nrd
       if(jjf .eq. (nils-1)) jjf=jjf+1
       if(jjf .eq. (nils-2) .and. i .eq.
&          (nused1(imw)-2) .and. cint(imw) .eq. 'cub')
&           jjf=jjf+1
```

The total number of points of the compressed grid to be taken in account for *jsam,* i.e. *ilim(2,jsam,imw*), is increased of one unit each time condition either (*) or condition (**) is verified.

```
      ilim(2,jsam,imw)=ilim(2,jsam,imw)+1
```

Besides, also the first point of the compressed grid to be taken into account for *jsam* (*ilim(1,jsam,imw))* is calculated.

```
      if(ilim(2,jsam,imw) .eq. 1) ilim(1,jsam,imw)=i
```

7. Computation of *rsan(imxi,imxsi2,4,imxmw)*

The matrix *rsan* is computed performing the following summations:

$$rsan(jsam,i,n,imw) = \sum_{\substack{j=jji,\,k=\max(0,\,-igridc(i,imw)+((jsam-1)\cdot nrd+1)) \\ j=jjf,\,k=\min(igridc(i+1)-1-igridc(i),((jsam-1)\cdot nrd+nils-igridc(i)))}} rils(nils-j+1)\cdot k^{n-1},$$

where $1 \le n \le 4$.

```
      do j=jji,jjf
       rsan(jsam,i,1,imw)=rsan(jsam,i,1,imw)+rils1(j)
       rsan(jsam,i,2,imw)=rsan(jsam,i,2,imw)+rils1(j)*
&             dble(k)
      if(cint(imw) .eq. 'cub')then
       rsan(jsam,i,3,imw)=rsan(jsam,i,3,imw)+rils1(j)*
&             dble(k*k)
       rsan(jsam,i,4,imw)=rsan(jsam,i,4,imw)+rils1(j)*
&             dble(k*k*k)
      end if
```

```
 k=k+1
end do
if(jjf .eq. nils ) goto [end do over jsam]
end if
```

2.2.31 READ_LOOKUP_PT

Description

This routine is called by retr_pt.f module only if *lookupc* = .true.
It reads the cross-section look-up tables.

Variables exchanged with external modules

Name	Description
ilookupmw	integer*4 ilookupmw(imxmw) ilookupmw(imw)=0 no look-up tables for mw imw ilookupmw(imw)=1 look-up tables for all the absorbers of the mw ilookupmw(imw)=2 look-up tables for not all the absorbers of the mw
lmgas	logical lmgas(imxgmw,imxmw): lmgas(mgas,imw)=.true. : calculation of cross-sections without look-up tables lmgas(mgas,imw)=.false. :calculation of cross-sections by means of look-up tables
smw	character*6 smw(imxmw) : microwindow identifier
nselmw	total number of selected microwindows
igasmw	I*4 igasmw(imxmw): number of gases to be considered in each Mw
igashi	I*4: igashi(imxgas) HITRAN code number for each global gas number
igasnr	I*4: igasnr(imxgas,imxmw): global gas number for the local gas number of each Mw
dsigma	R*8 dsigma(imxsig,imxmw): general wavenumber fine grid
isigma	I*4: isigma(imxmw) : number of general wavenumber fine grid points
nll	I*4 nll(imxgmw,imxmw): number of basis vectors
npl	I*4 npl(imxgmw,imxmw): number of -log(pressure) tabulation points
rp1l	R*4 rp1l(imxgmw,imxmw): lowest -log(pressure) [-ln(p), p in mb]
rdpl	R*4 rdpl(imxgmw,imxmw): spacing of -log(pressure) tabulation
ntl	I*4 ntl(imxgmw,imxmw): number of temperature tabulation points
rt1l	R*4 rt1l(imxgmw,imxmw): lowest tabulated temperature [k]
rdtl	R*4 rdtl(imxgmw,imxmw): spacing of temperature tabulation [k]
ru	R*4 ru(imxsi2,imxbv,imxgmw,imxmw): U-matrix
rkl	R*4 rkl(imxbv,imxnx,imxgmw,imxmw): K-matrix
iigrid	I*4 iigrid(imxsig,imxmw): *iigrid (1→ isigma(imw),imw):* irregular grid in the '0' and '1' representation for all the fine grid points of the extended microwindow *imw*

lirrgridmw	logical: lirrgridmw(imxmw): logical variable indicating, for each microwindow, if the irregular grid is available.
tab	character*3 tab(imxgmw,imxmw): tabulation code for cross-section look-up tables

Detailed description

The location of the directory containing LUTs is read from the environment variable ORM_LUT.

```
call getenv ("ORM_LUT",siod)
    iodl = index(siod," ")-1   ! computes the length of siod
    if (iodl.gt.80) then
      write(*,*)' --- FATAL ERROR in main orm ---'
      write(*,*)'Path of the I/O directories too long !'
      stop
    end if
    if (siod(iodl:iodl).ne.'/') then
      write(*,*)' --- FATAL ERROR in main orm ---'
      write(*,*)'The environment variable ORM_LUT'
      write(*,*)'must end by / '
      stop
    end if
```

1. Initialisation to 0 of matrix *ru*:

```
     do i=1,nselmw
       do j=1,igasmw(imw)
         do k=1,imxnx
           do ki=1,imxsi2
            ru(ki,k,j,i)=0.
           end do
         end do
        end do
     end do
```

2. Opening of the file containing the look-up table

For each microwindow, if at least the cross-section look-up table relative to one gas is available (*ilookupmw(imw)* ≠ 0), a do-loop over the gases in the considered microwindow is performed.
If the look-up table relative to the considered gas is available (*lmgas* = .false.), the hitran code of this gas *ihit* is computed and the file containing the look-up table (lookup_//smw(imw)//'_'//num//'.dat') is read.
Please note that the look-up tables are contained in binary files.
smw(imw) represents the identification label of the microwindow, *num* is the character containing *ihit.*

```
    do imw=1,nselmw
      if(ilookupmw(imw).ne.0) then
```

```
      do mgas=1,igasmw(imw)

        if(.not. lmgas(mgas,imw)) then

         ihit=igashi(igasnr(mgas,imw))

         write(num,'(i2.2)')ihit

          open(99,file=siod(1:iodl)//
     &          'lookup_'//smw(imw)//'_'//num//'.dat',
     &           form= ‘unformatted’ ,status='old')

         3. Reading of the file

        end if
      end do
    end if
   end do
```

3. Reading of the file

For the format of the binary files containing the look-up tables, please refer to [RD4].

An arbitrary number of initial comments records, starting with ‘!’ or ‘ ‘, can be present at the beginning of the file.

```
100       continue
          read(99) header
      if ( index ( header, '!' ) .gt. 0 .or. header .eq. ' ') goto 100
```

Reading of the tabulation code *tab* of the LUT and check on its value: if *tab* is not equal to one of the following strings:

log, lin, 4rt,

a message is written and this look-up table is not taken into account (lmgas(mgas,imw) is set to true).

```
          tab(mgas,imw)=header(11:13)

          if (tab(mgas,imw) .ne. 'log' .and.
     &        tab(mgas,imw) .ne. 'LOG' .and.
     &        tab(mgas,imw) .ne. 'lin' .and.
     &        tab(mgas,imw) .ne. 'LIN' .and.
     &        tab(mgas,imw) .ne. '4rt' .and.
     &        tab(mgas,imw) .ne. '4RT') then
          write(*,*) 'Problems in the tabulation of'
          write(*,*) 'cross-section look-up tables:'
```

```
        write(*,*) 'the tabulation does not correspond'
        write(*,*) 'to one of the functions coded'
        write(*,*) 'in the decompression stage'
        lmgas(mgas,imw)=.true.
        goto { end do over gases}
       end if
```

Reading of the dimension record

```
     read(99)   nll(mgas,imw),nvl,rv1l,rdvl,npl(mgas,imw),
    &           rp1l(mgas,imw),rdpl(mgas,imw),
    &           ntl(mgas,imw),rt1l(mgas,imw),
    &           rdtl(mgas,imw)
```

Check on the starting frequency of the look-up table:
calculation of the points *istart* of the look-up table to be skipped (this number will be different of 0 only if the look-up table is built for a wider interval than the extended microwindow). On the contrary, if the look-up table is build for a smaller interval than the extended microwindow, the logical variable *lmgas(mgas,imw)* will be set to true and the program will skip to consider next gas.

```
       xistart=(dsigma(1,imw)-dble(rv1l))/dble(rdvl)  !!!!!!!!!!!!!
       istart=nint(xistart)
        if(istart .lt.0) then
         lmgas(mgas,imw)= .true.
         go to { end do over gases}
        end if
```

Calculation of *nx*, the total number of tabulated (p,T) values.

```
       nx=npl*ntl
```

The first *istart* rows of matrix *ru* are read, but not stored :

```
      do i=1,istart
        read(99)(rummy(j),j=1,nll(mgas,imw))
       end do
```

Reading of the rows of *ru*-matrix corresponding to each frequency point of the extended microwindow.
If an irregular grid is available for microwindow *imw*, only the rows of the *ru*-matrix corresponding to the frequency points *i* with *iigrid(i,imw) =1* are stored and they are stored on a *'compressed grid'*.
The *'compressed grid'* is made of all and only the frequency points *i* with *iigrid(i,imw) =1*.

```
      ii=0
      do i=1,isigma(imw)
       if ((iigrid(i,imw) .eq.1 .and.
&       lirrgridmw(imw)) .or. .not. lirrgridmw(imw))then
        ii=ii+1
       read (99) (ru(ii,j,mgas,imw),
&                    j=1,nll(mgas,imw))

       else
        read(99)(rummy(j),j=1,nll(mgas,imw))
       end if
      end do
```

The last *istart* rows of matrix *ru* are read, but not stored :

```
      do i=istart+isigma(imw)+1,
&         (nvl-istart-isigma(imw))
         read(99)(rummy(j),j=1,nll(mgas,imw))
       end do
```

Reading of the matrix *rkl*

```
      do j=1,nx
       read(99)(rkl(i,j,mgas,imw),
&                  i=1,nll(mgas,imw))
       end do
```

2.2.32 DECOMPR_PT

Description:
It returns the absorption coefficient vector *rcross1(1:isigma(imw))* across the whole microwindow for path conditions (*rp*,*rt*) (where *rp* is -log(pressure/mb), *rt* is the temperature) in units of cm^2/molecules.

Variables exchanged with external modules:

Name:	Dimension:	Description:
rp	R*4	-log(pressure) of the path we are considering
rt	R*4	temperature of the path we are considering
mgas	I*4	local index of the gas
imw	I*4	index of the mw we are considering
ru	R*4 *ru(imxsi2,imxbv, imxgmw,imxmw)*	U-matrix

rkl	R*4 *rkl(imxbv,imxnx, imxgmw,imxmw)*	K-matrix
nll	I*4 *nll(imxgmw, imxmw)*	number of basis vector
npl	I*4 *npl(imxgmw, imxmw)*	number of -log(pressure) tabulation points
rp1l	R*4 *rp1l(imxgmw, imxmw)*	lowest -log(pressure) value
rdpl	R*4 *rdpl(imxgmw, imxmw)*	spacing of -log(pressure) tabulation
ntl	I*4 *ntl(imxgmw, imxmw)*	number of temperature tabulation points
rt1l	R*4 *rt1l(imxgmw, imxmw)*	lowest tabulated temperature
rdpl	R*4 *rdpl(imxgmw, imxmw)*	spacing for U-matrix tabulation
ntl	I*4 *ntl(imxgmw, imxmw)*	number of temperature tabulation points
rt1l	R*4 *rt1l(imxgmw, imxmw)*	lowest tabulated temperature
rdtl	R*4 *rdtl(imxgmw, imxmw)*	spacing of temperature tabulation
isigma1	I*4	total number of frequency points in extended mw
tab	C*3	tabulation code of cross-section look-up tables
rcross1	R*4 *rcross1(imxsi2)*	returned cross-section vector for the whole microwindow (cm^2/molecules)

Detailed description:

The actual compressed/reconstructed tables can either represent the absorption coefficient *rcross* directly (tab='lin') or some functions:
tab='log' implies tabulation is of ln(k);
tab='4rt' implies tabulation is of sqrt(sqrt(k)).
This routine is able to handle 'lin', 'log' and '4rt' tabulation.

1. Setting of the parameter *rkmin*. This is necessary to ensure that ln(k) returns a reasonable value when the reconstructed absorption coefficients are close to zero or negative (possible with k or sqrt(sqrt(k))).

rkmin=1.0 e-38

2. Interpolation points and weights in -log(pressure) axis
Calculation of the nearest (left) point *ip* to *rp* on tabulated pressure grid, difference between *rp* and -log(pressure) in *ip, rdpl1*.
The variable *rxp* (*rxp= (rp-rp1l)/rdpl+1)* is limited to the range 1: *npl* to ensure there is no extrapolation in cases where the required *rp, rt* are outside the tabulated range (in this case the edge values are used).
The variable *ip* is limited to the range 1: (*npl*-1) to ensure that when *rxp=npl*, the interpolation does not attempt to access undefined memory elements.

```
rxp=(rp-rp1l(mgas,imw))/rdpl(mgas,imw)+1.
rxp=min( max (1.0,rxp),float(npl(mgas,imw)))
ip=min (int(rxp),npl(mgas,imw)-1)
rdpl1=rxp - float(ip)
```

3. Interpolation points and weights in temperature axis
Calculation of the nearest (left) point *it* to *rt* on tabulated pressure grid, difference between *rt* and temperature in *it, rdtl1*.
The variable *rxt (rxt=(rt-rt1l)/rdtl+1)* is limited to the range 1: *ntl* to ensure there is no extrapolation in cases where the required *rp, rt* are outside the tabulated range (in this case the edge values are used).
The variable *it* is limited to the range 1: (*ntl*-1) to ensure that when *rxt=ntl*, the interpolation does not attempt to access undefined memory elements.

```
rxt=(rt-rt1l(mgas,imw))/rdtl(mgas,imw)+1.
rxt=min( max (1.0,rxt),float(ntl(mgas,imw)))
it=min (int(rxt),ntl(mgas,imw)-1)
rdtl1=rxt - float(it)
```

4. Calculation of indices and weights in 1: *nx* dimension for (*rp*,*rt*) interpolation of *rkl*

```
II=ip + npl(mgas,imw) * (it-1)          ! low  -lnp, low  temperature
JI=II+1                                 ! high -lnp, low  temperature
```

```
      IJ=II+npl(mgas,imw)                    ! low  -lnp, high temperature
      JJ=IJ+1                                ! high -lnp, high temperature

      WII=( 1.0 -rdpl1) * (1.0 -rdtl1)
      WJI=rdpl1 * (1.0 - rdtl1)
      WIJ=(1.0 - rdpl1) * rdtl1
      WJJ= rdpl1 *rdtl1
```

5. Expansion from basis vectors

Calculation of cross-section corresponding to *rp* and *rt*.

```
      do iv=1,isigma1                                  ! loop over frequency grid points
                                                       ! (the grid is compressed if
                                                       ! irregular grid is available)
        rkkii=0.0
        rkkij=0.0
        rkkji=0.0
        rkkjj=0.0
        do il=1,nll(mgas,imw)                          ! loop over all basis vectors
         rkkii=rkkii+ru(iv,il,mgas,imw)*rkl(il,II,mgas,imw)   ! multiply ru * rkl(rp,rt)
         rkkij=rkkij+ru(iv,il,mgas,imw)*rkl(il,IJ,mgas,imw)
         rkkji=rkkji+ru(iv,il,mgas,imw)*rkl(il,JI,mgas,imw)
         rkkjj=rkkjj+ru(iv,il,mgas,imw)*rkl(il,JJ,mgas,imw)
        end do
         if (tab(mgas,imw) .eq. 'log' .or.             ! tabulation of ln(k)
     &       tab(mgas,imw) .eq. 'LOG') then
          rcross1(iv)=exp(WII*rkkii+WIJ*rkkij+
     &                WJI*rkkji+WJJ*rkkjj)
         else
          rcross1(iv)=exp(WII*log(max(rkkii,rkmin))+   ! tabulation of k or sqrt(sqrt(k))
     &                WIJ*log(max(rkkij,rkmin))+
     &                WJI*log(max(rkkji,rkmin))+
     &                WJJ*log(max(rkkjj,rkmin)))
          if (tab(mgas,imw) .eq. '4rt' .or.            ! tabulation of sqrt(sqrt(k))
     &        tab(mgas,imw) .eq. '4RT' )
     &         rcross1(iv)=rcross1(iv)**4
         end if
        rcross1(iv)=rcross1(iv)*10000./6.0221367e+23 !!! ! cross-section in cm^2/atoms
      end do
```

Notes

The above assumes that the absorption coefficients are required on the same wavenumber grid as the *ru* tabulation, so no interpolation is performed in the wavenumber dimension.

The interpolation in (p,T) is always carried out in ln(k) since ln(k) is generally a linear function of -ln(p) (in the Lorentz limit).

Since ln(k**0.25) = 0.25*ln(k), interpolation in (p,T) domain for 'LIN' and

'4RT' reconstructions is the same, with the expansion from k**0.25 to k left until the last step when KABS is calculated.

2.2.33 HEX_BIN

Description

This module makes the conversion between a hexadecimal character to a binary number of four digit.

Variables exchanged with external modules

Name	Description
c	character*1: hexadecimal character (0,1,2...,9,A,B...F)
n1	integer*2 : see below
n2	integer*2 : see below
n3	integer*2 : see below
n4	integer*2 : see below

Detailed description

According to the particular value of *c*, the module returns four integers that represent the binary representation of the hexadecimal character *c*.

c	*n1*	*n2*	*n3*	*n4*
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
A	1	0	1	0
B	1	0	1	1
C	1	1	0	0
D	1	1	0	1
E	1	1	1	0
F	1	1	1	1

2.2.34 CONT_CHAR_PT

Description

This subroutine evaluates the qualifiers that characterise continuum retrieved parameters. The routine is called by the output_pt module. In that occasion the qualifiers are calculated and directly written into the main output file of pT retrieval (pt_out.dat).

Variables exchanged with external modules

Name	Description
rxpar	rxpar(imxtop) = vector of the fitted parameters
rainv	rainv(imxtop,imxtop)= VCM of the retrieved parameters
ilimb	ilimb = number of measured geometries
ipar	ipar = number of parameter-levels (i.e. N. of elements of rzpar vector)
nselmw	nselmw = total number of selected microwindows for the retrieval
nucl	nucl = number of limb geometries to be skipped before starting continuum fit; numbering starts from top.
lokku	lokku(imxgeo,imxmw) = occupation matrix used for the selection of operational MW's for each observation geometry
lcfit	lcfit(imxgeo,imxmw) = continuum occupation matrix
lccmat	lccmat(imxgeo,imxmw) = logical matrix which identifies altitudes & MWs where the continuum is set equal to the continuum of a nearby MW (close-close MWs).

Detailed description

```
* This module contains the algorithm for deriving the quantities to be reported in Level 2 products
* for characterisation of continuum fitted parameters, starting from the ORM variables.
*
* The matrix lccmat(imxgeo,imxmw) identifies among the MWs/altitudes of the
* occupation matrix 'lokku',
* the MWs/altitudes which are tightly grouped with the next (leftwards) MW/altitude where
* continuum is fitted. This matrix is computed in the modules 'mwcont_pt(vmr)'
*
      subroutine cont_char_pt(rxpar,rainv,ilimb,ipar,nselmw,nucl,
     &           lokku,lcfit,lccmat)

      implicit none
      include 'parameters_pt.inc'
* Declaration of variables is omitted here

* Initialisation of computed variables:
      do i=1,ilimb
        do j=1,nselmw
          igroup_type(i,j) = 0          ! continuum parameter group type
          xsect(i,j) = 0.d0             ! continuum cross-section at sweep i, and MW j
          var(i,j) = 0.d0               ! variance of retrieved continuum at sweep i, MW j
          covp(i,j) = 0.d0              ! covariance between xsect(i,j) and p(i)
          covt(i,j) = 0.d0              ! covariance between xsect(i,j) and T(i)
        end do
      end do

      icpar = 0  ! initialisation of a counter
*
* Start of loop over sweeps where continuum is considered
* and loop over microwindows of the current retrieval
*
```

```
      do i=nucl+1,ilimb                ! loop on sweeps (altitudes)
        do j=1,nselmw          ! loop on microwindows
          if (lokku(i,j)) then    ! if the current mw 'j' is used at sweep 'i'
            if (lcfit(i,j)) then    ! if continuum is fitted at this sweep/mw
              icpar = icpar + 1  ! icpar counts continuum parameters

* We have to setup the group_type(i,j) for the current continuum parameter:
* 1 - this mw is isolated
* 2 - this mw is an edge of a loose group
* 3 - this mw is a leftmost  edge of a tight group
* 4 - this mw is a leftmost edge of a tight group AND an edge of a loose group
* 5 - this mw belongs to a tight group (but is not an edge of the group)
* 6 - this mw belongs to a loose group (but is not an edge of the group)
*
              loose = .FALSE.
              ltight = .FALSE.
* Look right:
              do k=j+1,nselmw
                if (.not.lcfit(i,k)) then
                 if (.not.lccmat(i,k).and.lokku(i,k)) loose=.TRUE.
                 if (lccmat(i,k).and.lokku(i,k)) ltight=.TRUE.
                else
                 goto 12
                end if
              end do
12              continue
* Look left:
              do k=j-1,1,-1
                if (.not.lcfit(i,k)) then
                 if ((.not.lccmat(i,k)).and.lokku(i,k)) loose=.TRUE.
                else
                 goto 13
                end if
              end do
13              continue
* Take a decision:
              if (.not.(loose.or.ltight)) igroup_type(i,j)=1
              if (loose.and.(.not.ltight)) igroup_type(i,j)=2
              if (ltight.and.(.not.loose)) igroup_type(i,j)=3
              if (ltight.and.loose) igroup_type(i,j)=4

********
* In p,T retrieval the following correspondences are valid:
              xsect(i,j) = rxpar(ilimb+ipar+icpar)
* var(xsect(i,j)) =
              var(i,j) = rainv(ilimb+ipar+icpar,ilimb+ipar+icpar)
* cov(xsect(i,j),p(i)) =
              covp(i,j)= rainv(i,ilimb+ipar+icpar)
* cov(xsect(i,j),T(i)) =
              covt(i,j)= rainv(ilimb+i,ilimb+ipar+icpar)
********

            else                ! if continuum is not fitted at this sweep/mw

              if (lccmat(i,j)) then
                igroup_type(i,j)=5   ! the mw belongs to a tight group
              else
                igroup_type(i,j)=6   ! the mw belongs to a loose group
              end if
```

```
        end if              ! end if cont. is fitted at this sweep/mw
      end if                ! end if mw is used at sweep 'i'
    end do                  ! end loop on microwindows
  end do                    ! end loop on sweeps
```

* writes the results into the main output file of the retrieval (pt_out.dat) :

```
  do j=1,nselmw
    do i=1,ilimb
      write(29,'(a5,i2,a9,i2)')'mw = ',j,',  lmb = ',i
      write(29,*)'group_type(lmb,mw) = ', igroup_type(i,j)
      write(29,*)'xsect(lmb,mw) = ',xsect(i,j)
      write(29,*)'var(xsect(lmb,mw)) = ', var(i,j)
      write(29,*)'cov(xsect(lmb,mw),p(lmb)) = ', covp(i,j)
      write(29,*)'cov(xsect(lmb,mw),t(lmb)) = ', covt(i,j)
    end do
  end do
  end
```

2.3 Variables and parameters used in the p,T retrieval program

The parameters used in the calculation are listed in the table below

Name	Description	Value
dcdop	used in Doppler broadening: sqrt(2 ln2 k avog / c^2)	3.5811737d-7
dext	extension of the (already with iadd*delta extended) microwindow where ioutin is set to 1 [cm^{-1}]	0.4
dinvpi	1/pi	0.318309886
dsqln2	sqrt(ln2)	0.832554611
dsqpi	sqrt(pi)	1.772453851
dtineig	minimim permitted value for the eigenvalues of A	1.0d-40
iqlclf	the quotient between coarse and fine wavenumber grid intervals	5
imxapo	maximum number of points in the apodisation function (path difference domain)	513
imxbv	max. number of base vectors of compressed look-up tables	10
imxcof	max number of coefficients for the calculation of the quotient of the partition sum (=4)	4
imxcop	max. number of continuum parameters	180
imxcta	max number of elements in the correction table of tangent altitudes due to refraction index	50
imxept	max number of extra paths	1
imxfcs	max number of frequencies to which cross sections are provided in the look-up tables	1
imxfpg	max number of elements in the fixed P grid imposed to the retrieval	50
imxgas	max number of gas in the retrieval	10
imxgeo	max number of simulated observations	18
imxgmw	max number of gases per MW	4
imxhit	number of gases in the HITRAN 96 data base	36
imxhol	max. number of holes between true elements in the columns of the occupation matrix	100
imxi	maximum number of sampling points in the synthetic spectra computed at the observed frequencies	100
imxsi2	max number of '1' points in the irregular grids of the considered microwindows	400
imxilc	max number of sampling point in the instrument line-shape function (course grid!)	1000
imxils	maximum number of sampling points in the instrument line-shape function (fine-grid!)	2400
imxism	max number of isotopes in HITRAN data base per molecule (=8)	8
imxiso	number of total isotopes in the HITRAN database	85
imxite	maximum number of macro-iterations in retrieval procedure	15
imxj	maximum dimension of J matrix (VCMobs = $J \cdot J^T$)	imxilc+imxi
imxlay	max number of layers for modelling the atmosphere (=imxlev-1)	imxlev-1

imxlev	max number of levels used for modelling the atmosphere	70
imxlin	max number of lines per microwindow	300
imxlmb	max number of parameters to be retrieved for each set of parameters (p,T,C,vmr)	18
imxmw	max number of microwindows	20
imxnx	max. number of p and T points considered in cross-section look-up tables (nx=np*nt)	1000
imxobs	max number of observational point (for Jacobian matrix)	2700
imxpat	max number of possible paths (be careful: imxpat* imxsig*4*imxgas is the number of bytes needed for the biggest field (variable rcross) in the program!)	imxlay+imxept* (imxgeo-1)
imxpcs	max number of P to which cross sections are provided in the look-up tables	1
imxpro	max number of elements in p, t profiles	100
imxpun	max. dimension of a pointer in ficarra_pt	100
imxri	max number of refraction indices provided in the corresponding file	50
imxsav	max. dimension of the saved vector used dy mwcont_pt and ficarra_pt	3000
imxsig	max number of wavenumber grid-points for a microwindow	5500
imxsl	max number of sub-levels between the pointings of the simulations	20
imxsnc	max number of sampling point for the sinc function used to interpolate the instrument line-shape function	4800
imxtcs	max number of T to which cross sections are provided in the look-up tables	1
imxtop	max number of parameters to be fitted	60
imxvt	max number of vibrational T provided in the corresponding file	20
rairmass	average molec. weigth of the air (kg/kmol) (US STD)	28.9644
rbc	Boltzmann constant (for density in mol/cm-3)	1.380658e-19
rc1	constant in the Planck-function (2 h c^2)	1.191043934e-3
rcn	constant in the refraction index expression (n=1.+(rcn*rt0n/rp0n)*p/T)	.000272632
rdmult	the number of Doppler half-widths from the line-centre from which the Lorentz function instead of the Voigt-function is used Error: rdmult=10 -> 1.5% ; rdmult=20 -> 0.4% ; rdmult=30 -> 0.18%	30.
refind	multiplicative constant in the expression of refraction index n: refind= rcn*rt0n/rp0n	rt0n*rcn/ rp0n
rg0	acceleration of gravity (m/s**2)	9.80665
rhck	h*c/k [K/cm-1]	1.4387687
rk	10^{-5}/rbc	10^{-5}/rbc
rmovr	1000 * rairmass / R(=8314.32[N.m/(kmol.K)])	3.483676
rp0h	reference pressure for pressure broadening	1013.25
rp0n	pressure on level sea for refraction index calculation	1013.25
rt0h	reference temperature for pressure broadening	296.
rt0int	reference temperature for the line intensity	296.

rt0n	temperature on level sea for refraction index calculation	288.16
rvlf	multiplier for (Doppler+Lorentz=~Voigt) half-width to determine the local fine grid	0.1
rvmult	rvmult is the number of (Doppler+Lorentz=~Voigt) half-widths from the line-centre where the transition between local coarse and local fine grid occurs (rvmult >= rdmult !!	50

The variables used in the calculation and exchanged by modules are listed in the table below

Name	**Dim-ensions**	**Description**	**Modified in**
cint	imxmw	character*3: it indicates, for each microwindow, what kind of interpolation has to be performed between the spectral points of the irregular grid.	read_irrgrid_pt
delta		distance between fine-wavenumber grid points [cm^{-1}]	finput_pt
deps		maximum relative variation for each iteration in calculation of curtis-godson variables	finput_pt
dsigm0		central frequency of the line used for testing P levels [cm^{-1}]	finput_pt
dsigma	imxsig, imxmw	wavenumber fine grid for each microwindow [cm^{-1}]	grid_pt
dsilin	imxlin, imxmw	central wavenumber for each line of each Mw [cm^{-1}]	finput_pt
dstep		distance between coarse-wavenumber grid points [cm-1]	finput_pt
iadd		number of fine-wavenumber grid points to be added on both sides of each microwindow (due to the ils-convolution)	ails_pt
ibase		number of base-levels	chbase_pt
icode	imxlin, imxmw	HITRAN molecular code for each line of each Mw	finput_pt
icontpar		total number of continuum parameters to be fitted	guesspar_pt
iderlay	imxlmb, 3	highest (x,1), lowest (x,3) and middle (x,2) (the one directly above the perturbed layer) which is affected by each derivative (imxlmb refers to the parameter-levels)	mkplev_pt
iept		actual number of extra paths	finput_pt
ifco		three positions switch used for enabling / disabling offset or offset and continuum fitting.	finput_pt
ifspmw	imxmw	index of the first sampling point of each MW * NOTE: the sampling point at frequency=0 has index=1	finput_pt
igas		number of different gases for actual retrieval	inigas_pt
igashi	imxgas	HITRAN code number for each global gas number	inigas_pt
igasmw	imxmw	number of gases to be considered for each mw	inigas_pt
igasnr	imxgas, imxmw	global gas number for the local gas number of each Mw	inigas_pt
igeo		number of simulated geometries	occusim_pt

igeocder	imxgeo,2	for each simulated geometry *j* the highest (*igeocder*(*j*,1)) and the lowest (*igeocder*(*j*,2)) parameter level which has to be considered for the continuum-derivatives	tcgeo_pt
igridc	imxsi2, imxmw	matrix which associates to each microwindow and each point of the compressed grid, the corresponding index on the regular fine grid.	read_irrgrid_pt
iigrid	imxsig, imxgeo, imxmw	irregular grid in the '0' and '1' representation for all the fine grid points of the extended microwindow *imw*.	read_irrgrid_pt
iiso	imxlin, imxmw	isotope number for each line of each Mw.	finput_pt
ilev		number of levels for simulations	mkplev_pt
ilim	2, imxi, imxmw	variable used for making the direct interpolation/convolution : *ilim(1,jsam,imw):* first point of the compressed grid to be considered for the computation of the low resolution spectral point at *jsam* for microwindow *imw*; *ilim(2,jsam,imw):* total number of points of the compressed grid to be considered for the computation of the low resolution spectral point at *jsam* for microwindow *imw*.	read_irrgrid_pt
ilimb		number of measured geometries	finput_pt
ilimbmw	imxmw	number of valid measured geometries per microwindow (number of 2 in each column of iocsim)	occusim_pt
iline	imxmw	number of lines in each microwindow	finput_pt
imaingas		HITRAN code of the main gas of the retrieval (=2 for CO_2 in the case of p-T-retrieval)	finput_pt
imw		number of the actual microwindow	fwdmdl_pt
iobs		total number of observations to be fitted	occusim_pt
iocsim	imxgeo, imxmw	occupation matrix for the simulations to performed = 0 no simulation required, = 1 simulation required without FOV = 2 simulation required with FOV	occusim_pt
ioutin	imxlin, imxmw	flag for each line =1: line-shape has to be calculated at each wavenumber inside the microwindow =2: line is considered as nearby continuum	finput_pt
ipar		number of parameter-levels	occusim_pt
ipath		number of different IAPT numbers in ipoint	point_pt
ipoint	imxlay, imxgeo	matrix, which attaches to each pair of layer/geometry the IAPT number	point_pt
ipro		number of elements contained in P, T and VMR profiles initial guess	finput_pt
irowmw	imxmw	the row of the Jacobian matrix where the actual	occusim_pt

		microwindow starts	
isigma	imxmw	number of general wavenumber fine grid points in each microwindow	grid_pt
iterg		macro - iteration index (Gauss)	retr_pt
iterm		micro - iteration index (Marquardt)	retr_pt
itglev	imxgeo	number of the tangent-level for each geometry	mkplev_pt
itop		total number of parameters to be fitted	guesspar_pt
lccmat	imxgeo, imxmw	This matrix identifies among the MWs/altitudes of the occupation matrix 'lokku', the MWs & altitudes which are tightly grouped with the next (leftwards) MW/altitude where continuum is fitted.	mwcont_pt
lconverg		logical variable which is true if convergence is reached	convchk_pt
lextinf1		switch for enabling the use of LOS info at each iteration	finput_pt
lfit	imxlmb	logical vector that identify the levels where the profiles are fitted: referred to rztang (to the mearsured geometries)	finput_pt
lfitgeo	imxgeo	logical vector that identify the levels where the profiles are fitted: referred to rzsi (to the simulated geometries)	occusim_pt
lifend		switch for enabling the use of LOS info only at the end of the iterations	finput_pt
lifwasucc		logical variable which is if FALSE only in case p,T retrieval was unsuccessful (too many micro-iterations)	retr_pt
lirrgridmw	imxmw	logical vector that, for each selected microwindow in the actual retrieval, indicates whether the irregular grid is available.	read_irrgrid_pt
lokku	imxgeo,imxmw	occupation matrix used for the selection of operational MW's for each observation geometry	finput_pt
lparbase	imxpro	logical vector that identify the levels where the profiles are fitted: referred to rzbase (to the base-levels)	chbase_pt
nailsdp		number of AILS data points	finput_pt
napod	imxapo	number of points of rapod , the apodisation function in interferogram domain. IT HAS TO BE (2**n+1), WITH n INTEGER.	finput_pt
nils		number of elements of rils	ails_pt
ninterpol		switch for the decision of interpolation of the absorption cross-sections for the geometries above the lowest geometry (only if the IAPT number of the path is increasing, which was decided during the calculation of ipoint) =-1: no interpolation, all cross-sections recalculated =0: all cross-sections above the lowest geometry are interpolated =1: new calculation only of the tangent-layer, all other	finput_pt

		layers interpolated =2: new calculation of the tangent-layer and the layer above, all others interpolated =3: ...	
nll	imxgw, imxmw	number of basis vectors in cross-section look-up tables	read_lookup_pt
npl	imxgw, imxmw	number of -log(pressure) tabulation points	read_lookup_pt
nrd		Ratio between general coarse grid step and general fine grid step	finput_pt
nsam	imxmw	number of sampling points in each MW (general coarse grid)	finput_pt
nselmw		total number of selected microwindows for the retrieval	finput_pt
ntl	imxgw, imxmw	number of temperature tabulation points	read_lookup_pt
nucl		nucl+1 = upper parameter level for continuum fit	retr_pt
nused1	imxmw	total number of points of the compressed grid for each microwindow	read_irrgrid_pt
ra	imxtop, imxtop	matrix defined as (transpose of rjacob) * rvcmobinv * rjacob	abcalc_pt, amodif_pt
rails	imxilc, imxmw	apodised instrument line shape for all selected MWs	finput_pt
rainv	imxtop, imxtop	matrix inverse of ra	ainvcal_pt
raircol	imxlay, imxgeo	air-column for each layer and each geometry [moec/cm^{-2}]	curgod_pt
rapod **real*4**	imxapo	apodisation function in path difference domain	finput_pt
rapod_sigma **real*4**	imxilc	apodisation function in spectral domain	finput_pt
rb **real*4**	imxtop, imxobs	matrix defined as (transpose of rjacob) * rvcmobinv	abcalc_pt
rbase		greater base of trapezium of Field of View function [km]	finput_pt
rblos	imxtop, imxlmb	matrix defined as the result of the matrix product (rjaclos)^{T} * rinvclos	abcalc_pt
rcbase	imxpro, imxmw	continuum on the base-levels for each MW [cm^2/molec]	chbase_pt
rcderfov	imxi, imxgeo,	derivate with respect to continuum after fov convolution [r.u./(cm^2/molec)]	fov_pt

	imxlmb		
rchisq	0:imxite	total chi square in the different iterations	difchi_pt
rchisqp	imxlmb,imxmw	chi-square for each observation geometry and each microwindow temperature profiles	difchi_pt
rclay	imxlay, imxmw	model-layer values of the continuum [cm^2/molec]	conlay_pt
rcol	imxlay, imxgeo, imxgas	column amounts for each layer, each geometry and each gas [molec/cm^2]	curgod_pt
rcolpert	imxlay, imxgeo, 2	columns of the main gas for the perturbed temperature profiles [molec/cm^2]	curgod_pt
rconint	imxlmb, imxmw	frequency range around each MW,in which the continuum can be considered as varying linearly. [cm^{-1}]	finput_pt
rconvc	3	thresholds used to check convergence criteria (see convchk-description)	finput_pt
rcprof	imxpro,imxmw	array containing continuum cross section as a function of altitude and microwindow [cm^2/molec]	finput_pt
rcross **real*4**	imxsi2, imxpat, imxgmw	absorption cross sections for each irregular grid point (1st index), each IAPT number (2nd index) and each gas (3rd index) for the actual Mw [cm^2/molec]	cross_pt
rcrossper t **real*4**	imxsi2, imxpat, 2	absorption cross sections for the main gas for each irregular grid point (1st index), each IAPT number (2nd index) and for the two equivalent temperature profiles (3rd index). *rcrosspert(i,j,1)* are the cross-sections calculated using the temperatures *rteqpert(j,1)* and *rcrosspert(i,j,2)* using *rteqpert(j,2).*	cross_pt
rearad		local radius of curvature of the earth [km]	finput_pt
redfact		reduction factor applied to 'rincz' when it produces not acceptable P levels	finput_pt
rdpl **real*4**	imxgw, imxmw	spacing of -log(pressure) tabulation in cross-section look-up table	read_lookup_pt
rdtl **real*4**	imxgw, imxmw	spacing of temperature tabulation in cross-section look-up table	read_lookup_pt
relow	imxlin, imxmw	lower state energy for each line of each Mw [cm^{-1}]	finput_pt
rexph	imxlin, imxmw	exponent for temp. dependence of air-broadenedhalf width	finput_pt
rexphref		exponent for the calculation of Lorentz h-w in mkplev	finput_pt
rhcorr	imxlmb	vector of the tangent height corrections	vc_heightcorr
rhw0	imxlin, imxmw	air broadened half width [cm^{-1}/atm] at 296 K	finput_pt

rhw0ref		half-width of the line used for testing P levels [cm^{-1}/atm] at 296 K	finput_pt
rhwvar		relative max. half-width variation allowed between two neighbouring P levels	finput_pt
rils	imxils, imxmw	instrument-line-shape function in the frequency fine grid	ails_pt
rincz		trial increment given to altitude for building P levels	finput_pt
rint0	imxlin, imxmw	line intensity for each line of each Mw [cm^{-1}/(molec*cm^{-2}]	finput_pt
rintils		ratio between the frequency step approximating infinitesimal spectral resolution and the integral of the ILS function	ails_pt
rinvclos	imxlmb, imxlmb	inverse of the VC matrix of the LOS engineering data	ffinput_pt
rjaccon	imxlmb* imxmw, imxtop	jacobian matrix for the derivatives of the continuum parameter-level values with respect to the continuum parameters	ficarra_pt
rjaclos	imxlmb, imxtop	jacobian matrix which links LOS engineering data with the unknowns of the retrieval	jacloscalc
rjacob **real*4**	imxobs, imxtop	Jacobian Matrix 1st index: observations 2nd index: parameters	jacsetmw_pt
rkl **real*4**	imxbv, imxnx, imxgmw, imxmw	K-matrix	read_lookup_pt
rlambda		Marquardt damping factor	retr_pt
rlambdadiv		coefficient used to decrease rlambda at each macro-iteration	finput_pt
rlambdain		initial value of rlambda	finput_pt
rlambdamul		coefficient used to increase rlambda at each micro-iteration	finput_pt
rlat		latitude of the actual limb-scan (deg.)	finput_pt
rlinchisq		χ^2 calculated in the linear approximation	newparest_pt
rlolin	imxlin, imxmw	lower limit where the line has to be considered [km]	finput_pt
rmaxtv1		max. allowed temp. variation between levels,when: 0 < altitude of level < rzt12 [K]	finput_pt
rmaxtv2		max. allowed temp. vatiation between levels, when: rzt12 < altitude of level < rulatm [K]	finput_pt
rmrmod	imxlev, imxgas	volume mixing ratio for each gas considered in actual retrieval on levels used for rad. tra. calc.	mkplev_pt
rnoise	imxmw,imxgeo	NESR dependent on geometry and microwindow	finput_pt
rnres	imxobs	vector of the differences between the observed spectra and the calculated ones; first all the geometries of the first microwindow starting from the first geometry,	difchi_pt

		then all the other microwindowsvector of the differences between the observed spectra and the calculated ones	
rnreslos	imxlmb	vector of residuals of LOS data	difchi_pt
robs	imxi, imxgeo, imxmw	observed spectra corresponding to the different tangent pressures and different microwindows (on the general wavenumber coarse grid) [r.u.]	finput_pt
roffs	imxmw	fitted instrumental offset for each mw [r.u.]	updprof_pt
ropath	imxlay, imxgeo	optical path length for each layer, each geometry [km]	curgod_pt
rp1l **real*4**	imxgw, imxmw	lowest -log(pressure) value in cross-section look-up table	read_lookup_pt
rpartcder	imxlay, imxlmb,imxmw	partial derivatives of the continuum layer values with respect to the parameter-level values	conlay_pt
rpbase	imxpro	pressure on the base-levels [hPa]	chbase_pt, updprof_pt
rpeq	imxpat, imxgas	equivalent pressures [hPa]	curgod_pt
rperc		maximum relative (with respect to rconint) distance between central frequencies of two microwindows which are defined as close-close ones for the definition of continuum emission	finput_pt
rpmod	imxlev	pressure on levels used for the radiat. transf. calc. [hPa]	mkplev_pt
rpprof	imxpro	vector of pressure profile as a function of altitude Z. [hPa]	finput_pt
rsan	imxi, imxsi2,4, imxmw	variable used for making the direct interpolation/convolution of the spectrum.	read_irrgrid_pt
rsl		half-difference between the bases of the trapezium (1/rsl gives the slope) [km]	finput_pt
rspct	imxi, imxgeo	spectrum for each geometry on the general coarse grid [r.u.] 1st index: general wavenumber coarse grid 2nd index: geometries to be simulated for the actual Mw	spectrum_pt
rspctcder	imxi, imxgeo,imxlmb	continuum derivative spectra on the general coarse grid for each geometry and each parameter level [r.u./(cm^2/molec)] 1st index: general wavenumber coarse grid 2nd index: geometries to be simulated for the actual Mw 3rd index: levels where the parameters are retrieved	spectrum_pt
rspfov	imxi, imxgeo,	simulated spectra corresponding to the different tangent pressures and different microwindows on the	fov_pt, addoff_pt

	imxmw	general wavenumber coarse grid: (rspct * FOV) [r.u.]	
rt1l **real*4**	imxgw, imxmw	lowest tabulated temperature in cross-section look-up table	read_lookup_pt
rtbase	imxpro	temperature of the base levels [K]	chbase_pt, updprof_pt
rteq	imxpat, imxgas	equivalent temperatures [K]	curgod_pt
rthres1		thresholds used to check convergence criteria (see convchk-description)	finput_pt
rthres2		thresholds used to check convergence criteria	finput_pt
rthres3		thresholds used to check convergence criteria	finput_pt
rtmain	imxpat	Curtis-Godson temperature of the main gas [K]	curgod_pt
rtmod	imxlev	temperature on levels used for the radiat. transf. calc. [K]	mkplev_pt
rtprof	imxpro	vector of temperature as a function of altitude Z. [km]	finput_pt
ru **real*4**	imxsi2, imxbv, imxgmw, imxmw	U-matrix	read_lookup_pt
rucl		upper continuum limit (km) above this altitude the continuum is not any more fitted	
rulatm		upper limit of the atmosphere [km]	finput_pt
ruplin	imxlin, imxmw	upper limit where the line has to be considered [km]	finput_pt
rvchcorr	imxlmb, imxlmb	VC matrix of the tangent height corrections	vc_heightcorr
rvcmobinv **real*4**	imxi, imxi, imxmw	blocks of the inverse of the variance covariance matrix of the observations for each selected microwindow of the actual retrieval.	sinvcal_mw_pt
rvcmobinvopt **real*4**	imxi, imxi	optimised block of inverse of the variance covariance matrix of the observations	sinvcal_pt
rvmrbase	imxpro,imxgas	volume mixing ratio of the gases on the base levels [ppm]	chbase_pt, updprof_pt
rvmrprof	imxpro,imxgas	matrix of VMR profiles [ppm]	finput_pt
rwmol	imxhit, imxism	molecular weight for each HITRAN molecular code and isotope number [g/mol]	wmol_pt
rwmolref		molecular weight of the gas used for testing P levels [g/mol]	finput_pt
rxpar	imxtop	vector of the fitted parameters	guesspar_pt, newparest_pt
rxparold	imxtop	vector of the fitted parameters at the previous iteration	newparest_pt
rzbase	imxpro	altitude of the base-levels [km]	chbase_pt,

			updprof_pt
rzc0		altitude (km) above which the initial guess continuum profiles are forced to be zero	finput_pt
rzerof		zero-filling expressed as the ratio between measured and transformed interferogram	finput_pt
rzmod	imxlev	heights of levels used for the radiat. tranf. calc. [km]	mkplev_pt
rzmodpert	imxlev, imxlmb	perturbed altitude grids after the perturbation of temp. profiles. [km]	mkplev_pt
rzpar	imxlmb	vector of the altitudes where the temperature profile is fitted [km]	updprof_pt, guesspar_pt
rzprof	imxpro	vector of altitudes Z to which rtprof, rpprof and rvmrprof are referred [km]	finput_pt
rzsi	imxgeo	tangent altitudes of the geometries to be simulated [km]	occusim_pt, updprof_pt
rzt12		altitude where the temperature threshold changes from rmaxtv1 to rmaxtv2 [km]	finput_pt
rztang	imxgeo	vector containing the engineering values of tangent altitudes [km]	finput_pt
smw	imxmw	character*6: vector containing the identifying label of the selected microwindows	input_pt
tab	imxgmw, imxmw	character*3: tabulation code of cross-section look-up tables	read_lookup_pt

3. Software architecture and algorithms of level 2 VMR retrieval scientific code

In this section the software architecture and the algorithms used in VMR retrieval scientific code are specified. In Section 3.1 the high level flow diagram of the calls between main modules and the detailed calling tree are described. The tree of calls of each module, its I/O data and the algorithms are described in section 3.2.

3.1 High level flow diagram of calls

Fig. 9 shows the high level flow diagram of calls of the VMR retrieval module. Each box corresponds to a single main module of the program. The FWDMDL_VMR module is however an exception and it contains more than one main module. The operations described in the flow diagram of Fig. 9 are carried-out by the program module named 'RETR_VMR'.

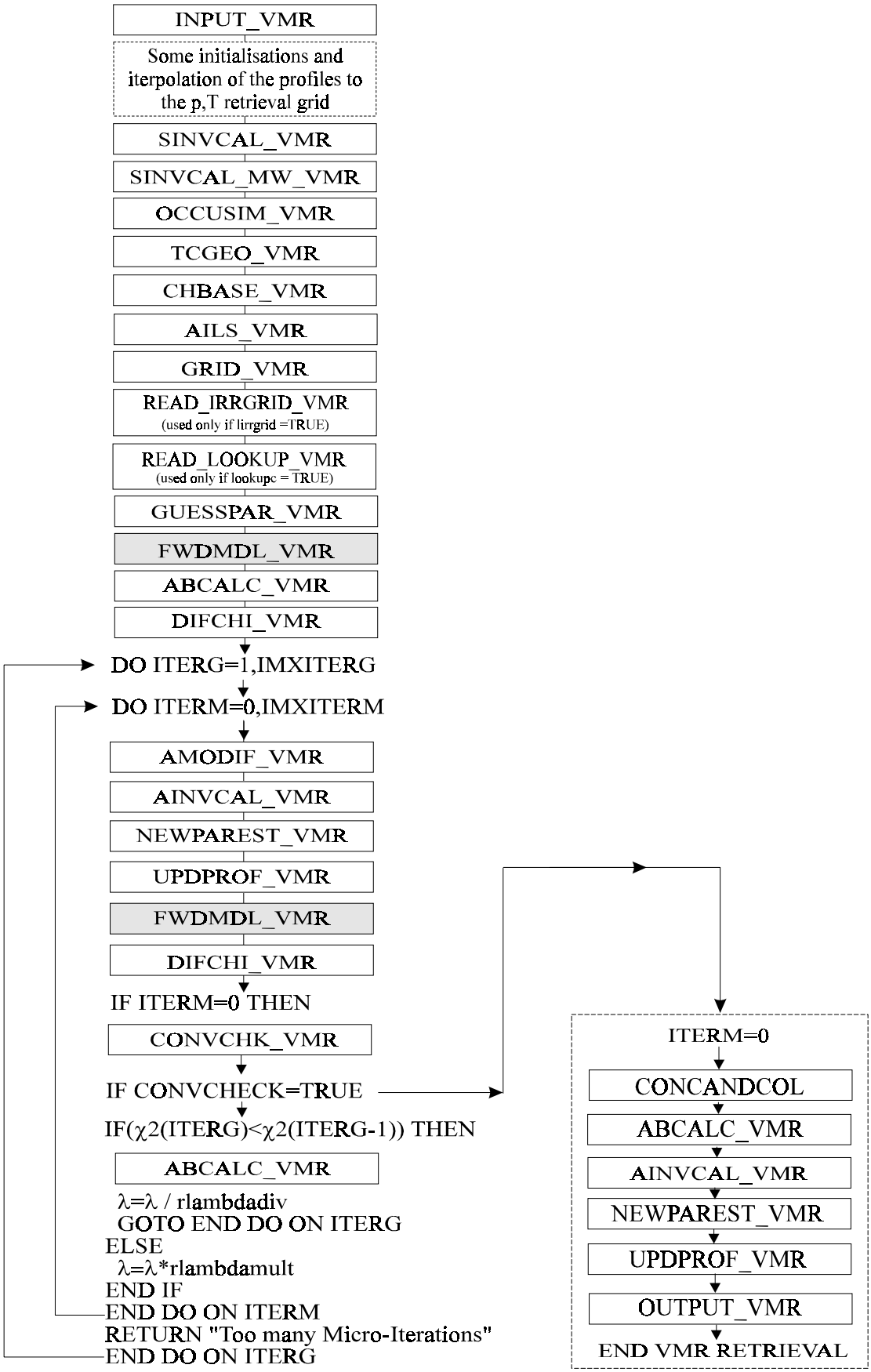


Fig. 9 Flow diagram of VMR retrieval module

Below the calling tree RETR_VMR module is described.

```
RETR_VMR ]
        |-----INPUT_VMR *
        |??---LOGINT_VMR *
        |??---LINP_VMR *
        |???--SINVCAL_VMR *
        |??---SINVCAL_MW_VMR *
        |??---OCCUSIM_VMR *
        |-----GCGEO_VMR *
        |-----CHBASE_VMR *
        |-----FAILS_VMR *
        |-----GRID_VMR *
        |?----READ_IRRGRID*
        |?----READ_LOOKUPC*
        |-----GUESSPAR_VMR *
        |-----FWDMDL_VMR *
        |-----ABCALC_VMR *
        |-----DIFCHI_VMR *
        |-----AMODIF_VMR *
        |-----AINVCAL_VMR *
        |-----NEWPAREST_VMR *
        |((---UPDPROF_VMR *
        |((---FWDMDL_VMR *
        |((---DIFCHI_VMR *
        |((?--CONVCHK_VMR *
        |((?--ABCALC_VMR *
        |((---OUTPUT_VMR *
        |((---CONCANDCOL *
        |((?--ABCALC_VMR *
        |((?--AINVCAL_VMR *
        |((---NEWPAREST_VMR *
        |((---UPDPROF_VMR *
        |((---OUTPUT_VMR *
```

The complete structure of this program is described in Fig. 10. Frames drawn with a continuous line are main modules, i.e are source (.f) or object (.o) files. Outlined frames refer to submodules contained in one of the main modules. For a detailed description of program modules see section 3.2.

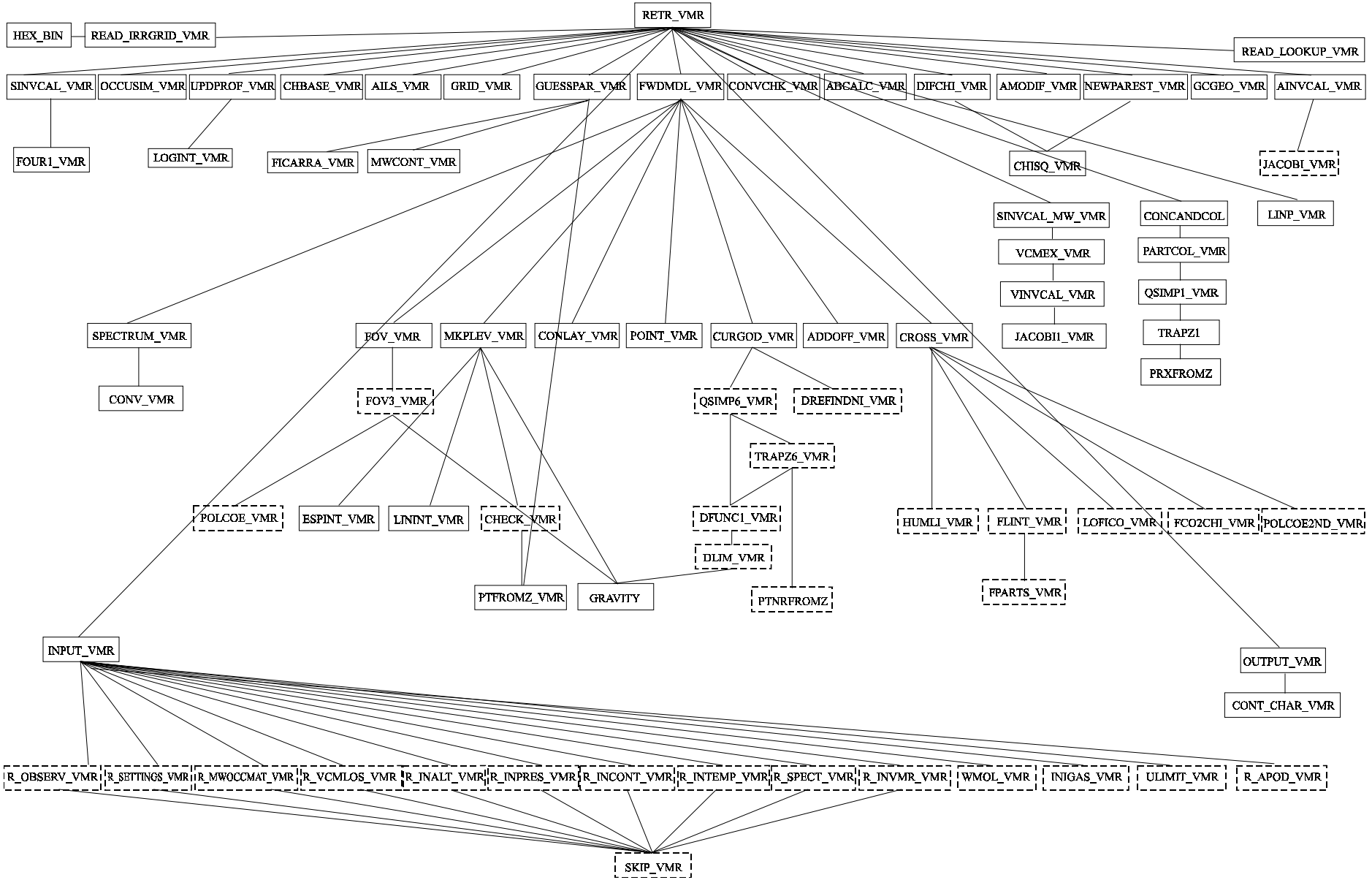

RETR_VMR
HEX_BIN
READ_IRRGRID_VMR
READ_LOOKUP_VMR
SINVCAL_VMR
OCCUSIM_VMR
UPDPROF_VMR
CHBASE_VMR
AILS_VMR
GRID_VMR
GUESSPAR_VMR
FWDMDL_VMR
CONVCHK_VMR
ABCALC_VMR
DIFCHI_VMR
AMODIF_VMR
NEWPAREST_VMR
GCGEO_VMR
AINVCAL_VMR
FOUR1_VMR
LOGINT_VMR
FICARRA_VMR
MWCONT_VMR
CHISQ_VMR
JACOBI_VMR
SINVCAL_MW_VMR
CONCANDCOL
LINP_VMR
VCMEX_VMR
PARTCOL_VMR
VINVCAL_VMR
QSIMP1_VMR
JACOBI1_VMR
TRAPZ1
PRXFROMZ
SPECTRUM_VMR
FOV_VMR
MKPLEV_VMR
CONLAY_VMR
POINT_VMR
CURGOD_VMR
ADDOFF_VMR
CROSS_VMR
CONV_VMR
FOV3_VMR
QSIMP6_VMR
DREFINDNI_VMR
TRAPZ6_VMR
POLCOE_VMR
ESPINT_VMR
LININT_VMR
CHECK_VMR
DFUNC1_VMR
HUMLI_VMR
FLINT_VMR
LOFICO_VMR
FCO2CHI_VMR
POLCOE2ND_VMR
DLIM_VMR
PTFROMZ_VMR
GRAVITY
PTNRFROMZ
FPARTS_VMR
INPUT_VMR
OUTPUT_VMR
CONT_CHAR_VMR
R_OBSERV_VMR
R_SETTINGS_VMR
R_MWOCCMAT_VMR
R_VCMLOS_VMR
R_INALT_VMR
R_INPRES_VMR
R_INCONT_VMR
R_INTEMP_VMR
R_SPECT_VMR
R_INVMR_VMR
WMOL_VMR
INIGAS_VMR
ULIMIT_VMR
R_APOD_VMR
SKIP_VMR

Fig. 10 Complete calling tree of VMR retrieval module.

Since in the 'RETR_VMR' module performs also some initialisations and other operations which are marginal with respect to the flow of the calls, but fundamental for the successful completion of the retrieval, we report here the details of the algorithms contained in RETR_VMR module.

Variables echanged with external modules

The INPUT_VMR routine which is called by RETR_VMR at the beginning of the flow, reads and sets-up all the program variables used by the different program modules. The variables are passed from INPUT_VMR to RETR_VMR through common statements.

Detailed description

We report hereafter an extract of the FORTRAN source code of RETR_VMR module, where the performed operations are explained in detail. Please note that the program lines included whithin the 'special' comment lines '* +++++++++++++++++' are not to be included in the level 2 prototype code because are used only for debugging purposes.

```
      subroutine retr_vmr(im)
      implicit none
      include 'parameters_vmr.inc'
```

Variables declarations and common statements are not reported here, please check the source code of retr_vmr.f

```
****
* Reads input files:
      rdtime = etime(rtar)
      write(*,*)'before input'
      call input_vmr(im)
      rdtime = etime(rtar)
      write(*,*)'E_Time after input_vmr (s) = ',rtar(1)+rtar(2)
*
* Reads the environment variable TEP which establishes
* whether this is a test run for writing-out variables at the TEPs:
      call getenv('TEP',step)

      write(*,*)'lfit = ',(lfit(j),j=1,ilimb)
*
      write(*,'(/a)')'lokku = '
      do j=1,ilimb
        write(*,*) (lokku(j,k),k=1,nselmw)
      end do
****
* checks  whether p,T retrieved profiles have to be used:
*
      if(lifptret) then
        write(*,*)
        write(*,*)' Using p,T retrieved profiles for VMR retrieval !!'
        write(*,*)
****
* Inserted by Dornier:
* Reading of p,T retrieved data from a dump file:
*
      open(34,file=sidir(1:iodl)//'pt_dump.dat',
     &       form='unformatted',status='unknown')
      write(*,*)'Retrieving from dump file:  ibaseret'
      read(34) ibaseret
      write(*,*)'Value(s) found:'
```

```
      write(*,*) ibaseret
      write(*,*)'  --------'

      write(*,*)'Retrieving from dump file: imxpropt'
      read(34) imxpropt
      write(*,*)'Value(s) found:'
      write(*,*) imxpropt
      write(*,*)'  --------'
      if (imxpropt.ne.imxpro) then
       write(*,*)'Fatal error in retr_vmr: '//
     &          'imxpro in p,T retrieval is NOT'
       write(*,*)'equal to imxpro in VMR retrieval. '//
     &          'Please check and correct'
       write(*,*)'files parameters_pt.inc and parameter_vmr.inc !!'
       stop
      end if

      write(*,*)'Retrieving from dump file: imxgeopt'
      read(34) imxgeopt
      write(*,*)'Value(s) found:'
      write(*,*) imxgeopt
      write(*,*)'  --------'
*
      if (imxgeopt.ne.imxgeo) then
       write(*,*)'Fatal error in retr_vmr: imxgeo '//
     &          'in p,T retrieval is NOT'
       write(*,*)'equal to imxgeo in VMR retrieval. '//
     &          'Please check and correct'
       write(*,*)'files parameters_pt.inc and parameter_vmr.inc !!'
       stop
      end if

      write(*,*)'Retrieving from dump file: ilimbpt'
      read(34) ilimbpt
      write(*,*)'Value(s) found:'
      write(*,*) ilimbpt
      write(*,*)'  --------'

      if (ilimbpt.ne.ilimb) then
       write(*,*)'You are trying to prform VMR retrieval '//
     &          'from a scan which'
       write(*,*)'has a different N. of sweeps compared to '//
     &          'the scan used in'
       write(*,*)'p,T retrieval.'
       write(*,*)'PROGRAM STOPPED !!'
       stop
      end if

      write(*,*)'Retrieving from dump file: rpbaseret'
      read(34) rpbaseret
      write(*,*)'Value(s) found:'
      write(*,'(8f10.5)')(rpbaseret(j),j=1,ibaseret)
      write(*,*)'  --------'

      write(*,*)'Retrieving from dump file: rtbaseret'
      read(34) rtbaseret
      write(*,*)'Value(s) found:'
      write(*,'(8f10.5)')(rtbaseret(j),j=1,ibaseret)
      write(*,*)'  --------'
```

```
      write(*,*)'Retrieving from dump file: rzbaseret'
      read(34) rzbaseret
      write(*,*)'Value(s) found:'
      write(*,'(8f10.5)')(rzbaseret(j),j=1,ibaseret)
      write(*,*)'  --------'

      write(*,*)'Retrieving from dump file: rztanret'
      read(34) rztanret
      write(*,*)'Value(s) found:'
      write(*,'(8f10.5)')(rztanret(j),j=1,imxgeopt)
      write(*,*)'  --------'
      write(*,*)'Data retrieval finished.'
      close (34)
*
****
* Interpolation of VMR and continuum profiles to the grid of retrieved p,T
* profiles
*
* Interpolations:
      do j=1,ibaseret
      if (rpbaseret(j).ge.rpprof(1)
     &    .and.rpbaseret(j).le.rpprof(ipro)) then
*
* Log interpolation in pressure for VMR profiles:
         do k=1,igas
          call logint_vmr(rpprof,rvmrprof(1,k),ipro,rpbaseret(j),
     &                    rv1(j,k))
         end do
* Linear interpolation in pressure for continuum profiles:
         do k=1,nselmw
          call linp_vmr(rpprof,rcprof(1,k),ipro,rpbaseret(j),
     &                    rcn1(j,k))
         end do
*
      else
* Extrapolations:
*
       if (rpbaseret(j).lt.rpprof(1)) then
         do k=1,igas
          rv1(j,k)=rvmrprof(1,k)+((rvmrprof(2,k)-rvmrprof(1,k))/
     &       log(rpprof(2)/rpprof(1)))*log(rpbaseret(j)/rpprof(1))
         end do
         do k=1,nselmw
          rcn1(j,k)=rcprof(1,k)+((rcprof(2,k)-rcprof(1,k))/
     &       (rpprof(2)-rpprof(1)))*(rpbaseret(j)-rpprof(1))
         end do
       else
         do k=1,igas
          rv1(j,k)=rvmrprof(ipro,k)+
     &       ((rvmrprof(ipro-1,k)-rvmrprof(ipro,k))/
     &       log(rpprof(ipro-1)/rpprof(ipro)))*
     &       log(rpbaseret(j)/rpprof(ipro))
         end do
         do k=1,nselmw
          rcn1(j,k)=rcprof(ipro,k)+((rcprof(ipro-1,k)-rcprof(ipro,k))/
     &       (rpprof(ipro-1)-rpprof(ipro)))*(rpbaseret(j)-rpprof(ipro))
         end do
       end if
```

```
      end if

      end do
*
* Update profiles:
      ipro = ibaseret
      do j=1,ipro
        rzprof(j)=rzbaseret(j)       ! Altitudes
        rpprof(j)=rpbaseret(j)       ! pressures
        rtprof(j)=rtbaseret(j)       ! temperatures
        do k=1,igas
           rvmrprof(j,k)=rv1(j,k)   ! VMR profiles for all the gases
        end do
        do k=1,nselmw
          rcprof(j,k)=rcn1(j,k)      ! Continuum profiles for all the MWs
        end do
      end do
*
* Update tangent altitudes:
      do j=1,ilimb
       rztang(j)=rztanret(j)         ! Tangent altitudes
      end do

****
* Checks whether also previously retrieved VMR profiles have to be used:
*
      if (lifvmret) then
*
* We check which are the retrievals that have already been performed:
*
      call system('ls '//sidir(1:iodl)//'???_dump.dat > dumplist')
      open(37,file='dumplist',status='old')
       j=0
16      read(37,'(a80)',END=18) sr3
       j=j+1
       k = index(sr3,"_dump")-3
       cname(j)=sr3(k:k+11)
       go to 16
18     close(37)
      call system('rm dumplist')
      inpr = j
      write(*,*)'The following retrievals have already'//
     &          ' been performed:'
      write(*,'(6(a3,2x))')(cname(j)(1:3),j=1,inpr)
      write(*,*)'The corresponding VMR profiles will be '//
     &    'used for the current retrieval.'
      write(*,'(a/)')'(if required !!)'

*
* Reading VMR profiles of previous retrievals:
*
      do j=1,inpr
        open(34,file=sidir(1:iodl)//cname(j),
     &       form='unformatted',status='old')
        read(34) ibase_prv(j)
        if(ibase_prv(j).ne.ibaseret) then
          write(*,*)'FATAL ERROR: '
          write(*,*)'You are trying to use a previously '//
```

```
     &    'retrieved VMR profile which is not '
          write(*,*) 'in the same grid of p,T retrieved profiles'
          write(*,*)' --- PROGRAM STOPPED --- '
          stop
         end if
        read(34) igashi_prv(j)
        do k=1, ibase_prv(j)
         read(34) rvmrbase_prv(k,j)
        end do
        close (34)
      end do
*
* for each of the gases of the current retrieval checks whether
* the profile of the current gas has already been retrieved.
* If yes, the previously retrieved profile is used:
*
      do j=1,igas
       do k=1,inpr
        if (igashi_prv(k).eq.igashi(j)) then
          write(*,*)'Using previously retrieved profile'//
     &             ' in '//cname(k)
          do m=1,ipro
            rvmrprof(m,j)=rvmrbase_prv(m,k)
          end do
        end if
       end do
      end do

      end if        ! end if lifvmret
*
      else          ! else lifptret
        write(*,*)
        write(*,'(a)')
     &  'Using templates p,T and VMR profiles for VMR retrieval !!'
        write(*,*)
      end if        !  end if lifptret
****
      sr1=sgas(im)(1:3)
      if(sr1(3:3).eq.' ') sr1(3:3)='_'
*
      rlambda=rlambdain     ! initialisation of Marquardt damping factor
      iterg=0                           ! intialisation of the Gauss-iterations index

**********************************************************************
* Atmospheric continuum profiles are scaled and set to zero where
* necessary:
      do k=1,ipro
       if (rzprof(k).gt.rzc0) then
         do j=1,nselmw
          rcprof(k,j) = 0.D0
         end do
       else
         do j=1,nselmw
          rcprof(k,j)=rcprof(k,j)*1.D30
         end do
       end if
      end do
* Setup of upper continuum limit (nucl) and umbrella radius:
      nucl = 0
```

```
      do j=1,ilimb
        if(rztang(j).gt.rucl.and.rucl.ge.rztang(j+1)) nucl=j
      end do
      write(*,'(a,f5.2,2x,i4,2x,f5.2)')
     &         'rucl, nucl, rperc = ',rucl,nucl,rperc

* +++++++
* Please note that this is only a re-initialisation which is not
* to be performed in the operational code:
      do j=1,nselmw
       do k=1,ilimb
         rconint(k,j) = 10.D0
       end do
      end do
* +++++++

* +++++++
      open(50,file=sodir(1:iodl)//sr1//'_rcprof_ref.dat'
     &     ,status='unknown')
      do j=1,ipro
      write(50,'(15e20.5)')rpprof(j),(rcprof(j,k)
     &                    ,k=1,nselmw)
      end do
      close(50)
* +++++++
*
* Instrumental offset is initialized to 0.
*
      do k=1,nselmw
        roffs(k)=0.D0
      end do
*
      write(*,*)'before sinvcal_vmr'
      call sinvcal_vmr(rapod,napod,rzerof,rapod_sigma,nailsdp,
     &                 rvcmobinvopt)
      call sinvcal_mw_vmr(rapod_sigma,nailsdp,rvcmobinvopt,nselmw,nsam,
     &                 rzerof,rvcmobinv)
*
      write(*,*)'before occusim'
      call occusim_vmr(rztang,ilimb,imaingas,lokku,nselmw,lfit,
     &          rbase,rzsi,nsam,igeo,lfitgeo,ipar,iocsim,ilimbmw,
     &          irowmw,iobs,rzpar)
*
      write(*,*)'Before gcgeo'
      call gcgeo_vmr(lfitgeo,ipar,igeo,lfit,ilimb,nucl,
     &               igeogder,igeocder)
*
      write(*,*)'before chbase'
      call chbase_vmr(rzprof,rtprof,rpprof,rvmrprof,rcprof,ipro,igas,
     &          nselmw,rztang,ilimb,rzpar,ipar,rlat,lfit,
     &          rzbase,rtbase,rpbase,rvmrbase,rcbase,ibase,lparbase)
*
* ++++++
* Writing out of initial guess VMR and CONTINUUM profiles:
*
      open(43,file=sodir(1:iodl)//sr1//'_inguessb_zpv.dat',
     &        status='unknown')
      do j = 1,ibase
        write(43,'(3(1pe12.4))')rzbase(j),rpbase(j),rvmrbase(j,1)
```

```
      end do
      close(43)
      open(50,file=sodir(1:iodl)//sr1//
     &            '_rcbase_ing.dat',status='unknown')
      do j=1,ibase
       write(50,'(15e20.5)')rpbase(j),(rcbase(j,k),k=1,nselmw)
      end do
      close(50)
* ++++++
*
      write(*,*)'before fails'
      call fails_vmr(nselmw,nailsdp,nrd,rails,delta,dstep,rils,iadd,
     &               nils,rintils)
*
      write(*,*)'before grid'
      call grid_vmr(nselmw,nsam,nrd,dstep,ifspmw,
     &         iadd,delta,iline,dsilin,ioutin,isigma,dsigma)
*
      if (lirrgrid) then
        write(*,*)'before read_irrgrid_vmr'
        call read_irrgrid_vmr(lirrgridmw,smw,nselmw,
     &                dsigma,isigma,delta,nrd,igeo,iigrid,
     &                cint,igridc,nused1,rsan,ilim,rils,
     &                nils,nsam)
      else
       do imw=1,nselmw
        lirrgridmw(imw)=.false.
       end do
      end if
*
      rdtime = etime(rtar)
      r1 = rtar(1)+rtar(2)
      write(*,*)'E_Time before read_lookup (s): ',r1
      write(*,*)'before read_lookup_vmr'
      if (lookupc) call read_lookup_vmr(ilookupmw,lmgas,smw,
     &                nselmw,
     &                igasmw,igashi,igasnr,dsigma,isigma,
     &                nll,npl,rp1l,
     &                rdpl,ntl,rt1l,
     &                rdtl,ru,rkl,iigrid,lirrgridmw,tab)
      rdtime = etime(rtar)
      r1 = rtar(1)+rtar(2) - r1
      write(*,*)'E_Time required for read_lookup (s): ',r1
*
      write(*,*)'before guesspar'
      call guesspar_vmr(rzbase,rvmrbase,rcbase,ibase,nselmw,
     &      rzpar,ipar,rztang,ilimb,lfit,lokku,lparbase,rperc,rconint,
     &      rxpar,itop,icontpar,rjaccon,isaved,dstep,nsam,
     &      ifspmw,nucl,lcfit,lccmat,ifco,rpbase)
*
* TEP_01_VMR:
      if(step.eq.'test')
     &   call tep_01_vmr(ibase,igeo,ipar,igas,ilimb,itop,
     &   nselmw,igasmw,isigma,igeocder,igeogder,rzbase,
     &   rtbase, rpbase,rcbase,rvmrbase,rztang,rzsi,lfit,
     &   lfitgeo,lokku,iocsim,rxpar)
*
      rdtime = etime(rtar)
      r1 = rtar(1)+rtar(2)
```

```
      write(*,*)'E_Time before first call to fwdmdl_vmr (s): ',r1
      write(*,*)'before fwdmdl'
      call fwdmdl_vmr(rzsi,igeo,rzbase,rtbase,rpbase,rvmrbase,
     &                rcbase,ibase,rulatm,rwmolref,dsigm0,
     &                rhw0ref,rmaxtv1,rmaxtv2,rzt12,rhwvar,
     &                igas,rexphref,rincz,redfact,rlat,
     &                lfitgeo,lparbase,nselmw,iept,rearad,
     &                deps,isigma,dsigma,delta,iocsim,igasmw,
     &                ruplin,rlolin,iline,icode,rint0,relow,rhw0,
     &                dsilin,ioutin,igasnr,rexph,rwmol,igashi,
     &                iiso,ninterpol,nsam,nils,rils,rintils,nrd,
     &                iadd,ilimbmw,lokku,nucl,ilimb,igeocder,
     &                igeogder,rjaccon,roffs,rbase,rsl,icontpar,
     &                itglev,rzmod,rtmod,rpmod,rxmod,
     &                ipar,irowmw,
     &                ilookupmw,lmgas,smw,nll,npl,
     &                rp1l,rdpl,ntl,rt1l,rdtl,ru,rkl,tab,
     &                rjacob,rspfov,iigrid,cint,lirrgridmw,
     &                igridc,nused1,rsan,ilim)
      rdtime = etime(rtar)
      r1 = rtar(1)+rtar(2)-r1
      write(*,*)'E_Time required for fwdmdl_vmr (s): ',r1

     rdtime = etime(rtar)
      r1 = rtar(1)+rtar(2)
      write(*,*)'before abcalc'
      call abcalc_vmr(rjacob,rvcmobinv,ra,rbt,iobs,
     &     itop,nselmw,ilimbmw,nsam,rnoise,ilimb,lokku)
      rdtime = etime(rtar)
      r1 = rtar(1)+rtar(2)-r1
      write(*,*)'E_Time required for abcalc_vmr (s): ',r1
*
      rdtime = etime(rtar)
      r1 = rtar(1)+rtar(2)
      write(*,*)'before difchi'
      call difchi_vmr(iobs,itop,robs,rspfov,rvcmobinv,rnoise,
     &           nsam,nselmw,ilimb,lokku,
     &           ilimbmw,iterg,rnres,rchisq,rchisqp)
      rdtime = etime(rtar)
      r1 = rtar(1)+rtar(2)-r1
      write(*,*)'E_Time required for difchi_vmr (s): ',r1

      write(*,*)'Retrieval of gas: ',sgas(im)
      write(*,*)'iterg,rchisq=',iterg,rchisq(iterg)
      rdtime = etime(rtar)
      write(*,*)'E_Time before starting iterations (s): ',
     &          rtar(1)+rtar(2)
*
*  Begin of the do-loop on macro-iterations
*
      do 10 iterg=1,imxiterg

      rdtime = etime(rtar)
      rtit = rtar(1)+rtar(2)      ! initialisation of E_Time of the current iteration

      write(*,'(//a,/a/)')c1,c1
      write(*,'(a,i2/)')'Starting GAUSS macro-iteration N. ',iterg
      write(*,*)' iterg=',iterg
*
```

```
*  Begin of the do-loop on micro-iterations
*
      do 20 iterm=0,imxiterm

     write(*,'(a,i3)')'Macro-iteration N. ',iterg
     write(*,'(a,i2)')'Marquardt micro-iteration index: '
    &                ,iterm

     write(*,*) 'iterm=',iterm

     write(*,'(a,e10.3)')'Lambda =  ',rlambda
*
        rdtime = etime(rtar)
        r1 = rtar(1)+rtar(2)
        write(*,*)'before amodif'
        call amodif_vmr(ra,rlambda,itop,ipar,icontpar)
        rdtime = etime(rtar)
        r1 = rtar(1)+rtar(2) - r1
        write(*,*)'E_Time required for amodif_vmr (s) = ',r1
*
        rdtime = etime(rtar)
        r1 = rtar(1)+rtar(2)
        write(*,*)'before ainvcal'
        call ainvcal_vmr (ra,itop,rainv)
        rdtime = etime(rtar)
        r1 = rtar(1)+rtar(2) - r1
        write(*,*)'E_Time required for ainvcal_vmr (s) = ',r1
*
        rdtime = etime(rtar)
        r1 = rtar(1)+rtar(2)
        write(*,*)'before newparest'
        call newparest_vmr(rainv,rbt,rnres,rxparold,itop,iobs,
    &                   iterm,rjacob,rxpar,rlinchisq,
    &                   rvcmobinv,rnoise,nsam,nselmw,ilimbmw,
    &                   ilimb,lokku)
        rdtime = etime(rtar)
        r1 = rtar(1)+rtar(2) - r1
        write(*,*)'E_Time required for newparest_vmr (s) = ',r1
*
* Constraining continuum parameters in physically meaningul ranges:
*
     do j=1,ipar
      if (rxpar(j).lt.0.0d0) then
        rxpar(j) = 1.0d-10
        write(*,*)'WARNING: VMR constrained at sweep: ',j
      end if
     end do
     do j=ipar+1, ipar+icontpar
      if (rxpar(j).lt.1.D-20) rxpar(j)=1.D-20
      if (rxpar(j).gt.1.D+20) rxpar(j)=1.D+20
     end do
*
     print*,'CHISQ in linear approx. rlinchisq = ',rlinchisq
        write(*,*)'before updprof'
     write(*,'(a)')' Actual values of rxpar :'
     write(*,'(6(1pe12.4))')(rxpar(j),j=1,itop)
     call updprof_vmr(rxpar,itop,ipar,rzpar,rzbase,
    &                ibase,rcbase,nselmw,rvmrbase,igas,roffs,
    &                lparbase,lokku,ilimb,ilimbmw,icontpar,
```

```
     &                isaved,nsam,ifspmw,dstep,rjaccon,
     &                nucl,rpbase)
*
      if(step.eq.'test')
     & call tep_06_vmr(ibase,icontpar,igas,nselmw,
     &          rzbase,lparbase,rcbase,roffs,isaved,
     &          rjaccon,rvmrbase)
*
* ++++++
        write(*,*)' rzbase, rtbase, rpbase, ibase = ',ibase
        do j=1,ibase
         write(*,*)rzbase(j),rtbase(j),rpbase(j)
        end do
        write(*,*)' rzsi,  igeo = ',igeo
        write(*,'(6f10.4)')(rzsi(j),j=1,igeo)
*
* Writing out VMR and CONTINUUM profiles:
*
      open(84,file=sodir(1:iodl)//sr1//
     &       '_retrb_zpv.dat',
     &       status='unknown')
      k=0
      do m=1,ibase
      if (lparbase(m)) then
        k=k+1
        r1i=rvmrbase(m,1)-sqrt(rainv(k,k))
        r2=rvmrbase(m,1)+sqrt(rainv(k,k))
      else
        r1i=0.D0
        r2=0.D0
      end if
        write(84,'(5(1pe12.4))')rzbase(m),rpbase(m),rvmrbase(m,1),
     &   r1i,r2
      end do
      close(84)
*
      open(50,file=sodir(1:iodl)//sr1//
     &            '_rcbase_ret.dat',status='unknown')
      do j=1,ibase
      write(50,'(15e20.5)')rpbase(j),(rcbase(j,k),k=1,nselmw)
      end do
      close(50)
* ++++++

        rdtime = etime(rtar)
        r1 = rtar(1)+rtar(2)
*
        write(*,*)'before fwdmdl'
        call fwdmdl_vmr(rzsi,igeo,rzbase,rtbase,rpbase,rvmrbase,
     &                rcbase,ibase,rulatm,rwmolref,dsigm0,
     &                rhw0ref,rmaxtv1,rmaxtv2,rzt12,rhwvar,
     &                igas,rexphref,rincz,redfact,rlat,
     &                lfitgeo,lparbase,nselmw,iept,rearad,
     &                deps,isigma,dsigma,delta,iocsim,igasmw,
     &                ruplin,rlolin,iline,icode,rint0,relow,rhw0,
     &                dsilin,ioutin,igasnr,rexph,rwmol,igashi,
     &                iiso,ninterpol,nsam,nils,rils,rintils,nrd,
     &                iadd,ilimbmw,lokku,nucl,ilimb,igeocder,
     &                igeogder,rjaccon,roffs,rbase,rsl,icontpar,
```

```
     &              itglev,rzmod,rtmod,rpmod,rxmod,
     &              ipar,irowmw,
     &              ilookupmw,lmgas,smw,nll,npl,
     &              rp1l,rdpl,ntl,rt1l,rdtl,ru,rkl,tab,
     &              rjacob,rspfov,iigrid,cint,lirrgridmw,
     &              igridc,nused1,rsan,ilim)
*
       rdtime = etime(rtar)
       r1 = rtar(1)+rtar(2) - r1
       write(*,*)'E_Time required fwdmdl_vmr (s) = ',r1
*
* We save the old residuals:
        do j=1,iobs
          rnresold(j) = rnres(j)
        end do

        rdtime = etime(rtar)
        r1 = rtar(1)+rtar(2)
        write(*,*)'before difchi'
        call difchi_vmr(iobs,itop,robs,rspfov,rvcmobinv,rnoise,
     &           nsam,nselmw,ilimb,lokku,
     &           ilimbmw,iterg,rnres,rchisq,rchisqp)
        rdtime = etime(rtar)
        r1 = rtar(1)+rtar(2) - r1
        write(*,*)'E_Time required difchi_vmr (s) = ',r1
*
        if(step.eq.'test')
     &   call tep_07_vmr(iobs,iterg,rnres,rchisq,
     &          ilimb,nselmw,rchisqp)
*
        write(*,*)'Retrieval of: ',sgas(im)
        write(*,*)'iterg,rchisq=',iterg,rchisq(iterg)
*
        if (iterm.eq.0) then
*
          write(*,*)'before convchk'
          call convchk_vmr(rchisq,iterg,rlinchisq,rxpar,rxparold,
     &      ipar,itop,iobs,rlambda,rconvc(1),rconvc(2),rconvc(3),
     &      lconverg)
*
          if (lconverg) then
          write(*,'(a)')'The convergence criteria are now verified,'
     &      //' exiting from iterations :-) '
          goto 30
          end if
        end if
*
        if (rchisq(iterg).le.rchisq(iterg-1).or.
     &       rchisq(iterg).lt.1.0) then
*
          rdtime = etime(rtar)
          r1 = rtar(1)+rtar(2)
          write(*,*)'before abcalc'
          call abcalc_vmr(rjacob,rvcmobinv,ra,rbt,iobs,
     &      itop,nselmw,ilimbmw,nsam,rnoise,ilimb,lokku)
          rdtime = etime(rtar)
          r1 = rtar(1)+rtar(2) - r1
          write(*,*)'E_Time required abcalc_vmr (s) = ',r1
          rlambda=rlambda/rlambdadiv
```

```
          goto 15
        else
          rlambda=rlambda*(rlambdamul-1.D0)/(1.D0+rlambda)
          do j=1,itop
            rxpar(j) = rxparold(j)
          end do
          do j=1,iobs
            rnres(j) = rnresold(j)
          end do

        end if
20    continue
      write(*,*)'Too many MARQUARDT micro-iterations'
      write(*,*)'Jumping to next VMR retrieval'
      go to 177
15      call output_vmr (rxpar,ipar,icontpar,rainv,
     &   nsam,robs,rspfov,rchisq,iobs,
     &   itop,iterg,iterm,rlambda,rlinchisq,
     &   ilimb,igeo,nselmw,rchisqp,slab,lokku,.true.,
     &    lcfit,lccmat,nucl,rvcol,rconc,rvcmcol,rvcmconc)

      rdtime = etime(rtar)
      rtit = rtar(1)+rtar(2) - rtit
      write(*,*)'E_Time spent in G. it. ',iterg,' was (s): ',rtit

10    continue
*
      write(*,'(//a,i3/)')'Maximum N. of allowed macro-iterations'
     & //' has been reached, iterg = ',iterg-1
30    continue
*
      write(*,'(a)')'Exited from iterations, producing now the output.'
*
      call concandcol(rzmod,rtmod,rpmod,rxmod,itglev,igeo,
     &    lfitgeo,rzbase,rvmrbase(1,1),ibase,lparbase,rainv,
     &    rvcmcol,rvcmconc,rvcol,rconc)

        rdtime = etime(rtar)
        r1 = rtar(1)+rtar(2)
        write(*,*)'before abcalc'
        call abcalc_vmr(rjacob,rvcmobinv,ra,rbt,iobs,
     &       itop,nselmw,ilimbmw,nsam,rnoise,ilimb,lokku)
        rdtime = etime(rtar)
        r1 = rtar(1)+rtar(2) - r1
        write(*,*)'E_Time required abcalc_vmr (s) = ',r1
*
        rdtime = etime(rtar)
        r1 = rtar(1)+rtar(2)
        write(*,*)'before ainvcal'
        call ainvcal_vmr (ra,itop,rainv)
        rdtime = etime(rtar)
        r1 = rtar(1)+rtar(2) - r1
        write(*,*)'E_Time required ainvcal_vmr (s) = ',r1
*
        rdtime = etime(rtar)
        r1 = rtar(1)+rtar(2)
        write(*,*)'before newparest'
        call newparest_vmr(rainv,rbt,rnres,rxparold,itop,iobs,
     &                   iterm,rjacob,rxpar,rlinchisq,
```

```
     &                rvcmobinv,rnoise,nsam,nselmw,ilimbmw,
     &                ilimb,lokku)
         rdtime = etime(rtar)
         r1 = rtar(1)+rtar(2) - r1
         write(*,*)'E_Time required for newparest_vmr (s) = ',r1
*
* Constraining continuum parameters in physically meaningul ranges:
*
      do j=1,ipar
       if (rxpar(j).lt.0.0d0) then
         rxpar(j) = 1.0d-10
         write(*,*)'WARNING: VMR constrained at sweep: ',j
       end if
      end do
      do j=ipar+1, ipar+icontpar
       if (rxpar(j).lt.1.D-20) rxpar(j)=1.D-20
       if (rxpar(j).gt.1.e+20) rxpar(j)=1.D+20
      end do
*
      write(*,'(/a)')'Final estimate of the vector rxpar:'
      write(*,'(6(1pe12.4))')(rxpar(j),j=1,itop)
      write(*,'(/a)')'Final estimate of the square errors on rxpar:'
      write(*,'(6(1pe12.4))')
     &        (rainv(j,j),j=1,itop)
*
         write(*,*)'before updprof'
      call updprof_vmr(rxpar,itop,ipar,rzpar,rzbase,
     &                ibase,rcbase,nselmw,rvmrbase,igas,roffs,
     &                lparbase,lokku,ilimb,ilimbmw,icontpar,
     &                isaved,nsam,ifspmw,dstep,rjaccon,
     &                nucl,rpbase)
*
* ++++++
      open(50,file=sodir(1:iodl)//sr1//
     &            '_rcbase_ret.dat',status='unknown')
      do j=1,ibase
      write(50,'(15e20.5)')rpbase(j),(rcbase(j,k),k=1,nselmw)
      end do
      close(50)
*
      open(84,file=sodir(1:iodl)//sr1//
     &          '_retrb_zpv.dat',
     &          status='unknown')
      k=0
      do m=1,ibase
      if (lparbase(m)) then
        k=k+1
        r1i=rvmrbase(m,1)-sqrt(rainv(k,k))
        r2=rvmrbase(m,1)+sqrt(rainv(k,k))
      else
        r1i=0.D0
        r2=0.D0
      end if
        write(84,'(5(1pe12.4))')rzbase(m),rpbase(m),rvmrbase(m,1),
     &   r1i,r2
      end do
      close(84)
* ++++++
*
```

```
      call output_vmr (rxpar,ipar,icontpar,rainv,
     &  nsam,robs,rspfov,rchisq,iobs,
     &  itop,iterg,iterm,rlambda,rlinchisq,
     &  ilimb,igeo,nselmw,rchisqp,slab,lokku,.false.,
     &  lcfit,lccmat,nucl,rvcol,rconc,rvcmcol,rvcmconc)
*
* Writing of vmr retrieved profile into a dump file:
*
      open(34,file=sodir(1:iodl)//sr1//'_dump.dat',
     &       form='unformatted',status='unknown')
      write(34) ibase
      write(34) igashi(1)
      do j=1,ibase
        write(34) rvmrbase(j,1)
      end do
      close (34)
*
177   continue
*
      close files associated with units form 10 to 30 and unit 40
*
* Closing tep files:
      if(step.eq.'test')then
        close files associated with units from 71 to 77
      end if
*
      return
      end
```

3.2 VMR retrieval modules architecture and algorithms

In this section the architecture and the algorithms of the VMR retrieval modules are described. The descriptions follow the guidelines explained in Sect. 2.2.

3.2.1 INPUT_VMR

Description: This is the subroutine which reads the input files used by VMR retrieval module. This subroutine makes also congruity checks on data and used parameters, builds memory structures that will be used later in the program and performs some initialisations. It makes available to the other routines all the variables read from input files.
The structure of this module is straightforward and it is not worth to describe in detail the operations therein performed. The used FORTRAN code, plenty of comments and self explanatory is reported in AD7. For completeness, we just list in the following the various small sub-modules used by the 'input_vmr' subroutine.

3.2.1.1 R_OBSERV_VMR

Description: subroutine used to read file of observations.
See the FORTRAN source code in [AD7]

3.2.1.2 R_SETTINGS_VMR

Description: subroutine used to read file of settings.
See the FORTRAN source code in [AD7]

3.2.1.3 R_MWOCCMAT_VMR

Description: subroutine used to read file of MW occupation matrix.
See the FORTRAN source code in [AD7]

3.2.1.4 R_INALT_VMR

Description: subroutine used to read file of altitudes to which the initial guess profiles are referred.
See the FORTRAN source code in [AD7]

3.2.1.5 R_INPRES_VMR

Description: subroutine used to read file of initial guess pressure profile.
See the FORTRAN source code in [AD7]

3.2.1.6 R_INTEMP_VMR

Description: subroutine used to read file of initial guess temperature.
See the FORTRAN source code in [AD7]

3.2.1.7 R_INCONT_VMR

Description: subroutine used to read file of observations.
See the FORTRAN source code in [AD7]

3.2.1.8 R_SPECT_VMR

Description: subroutine used to read file of spectroscopic database.
See the FORTRAN source code in [AD7]

3.2.1.9 WMOL_VMR

Description: Initialisation of the molecular isotope weights.
See fortran source code in [AD7].

3.2.1.10 INIGAS_VMR

Description: Initialisation of the variables *'igas, igashi,igasmw,igasnr'* that define the two internal gas codes. See par. 2.2.1.10 and the fortran source code in [AD7].

3.2.1.11 R_INVMR_VMR

Description: subroutine used to read file of initial guess VMR profiles.
See the FORTRAN source code in [AD7]

3.2.1.12 UPLIMIT_VMR

Description: It controls whether the variable ‘rulatm’, that represents the upper limit of the atmosphere, is both greater than the highest simulated geometry and less than the highest point of the initial profiles. See the fortran source code in [AD7]

3.2.1.13 R_APOD_VMR

Description: subroutine to read file of apodisation function in the interferogram domain.
See the fortran source code in [AD7]

3.2.1.14 SKIP_VMR

Description: subroutine to skip comment lines on read files convention is that: at least one comment line appears before a read statement last comment line starts with a character '#' in column 1. See the fortran source code in [AD7]

3.2.2 SINVCAL_VMR

For the description of this module see section 2.2.2.

3.2.3 SINVCAL_MW_VMR

For the description of this module see section 2.2.3.

3.2.3.1 VCMEX_VMR

For the description of this module see section 2.2.3.1.

3.2.4 VINVCAL_VMR

For the description of this module see section 2.2.4.

3.2.5 OCCUSIM_VMR

For the description of this module see section 2.2.5.

3.2.6 GCGEO

Description: Determination of the vectors *igeogder* and *igeocder*, that relate to each simulated geometry the parameter levels which have to be considered for the derivatives (i.e. the parameter levels where a change of the parameter influences the spectrum).

Variables exchanged with external modules:

Name:	Description:
lfitgeo	logical vector that is true if a simulated geometry is also a parameter-level
ipar	number of parameter levels
igeo	number of simulated geometries
lfit	logical vector that is true if an observational level is also a parameter level
ilimb	number of measured geometries
nucl	nucl+1 = upper parameter level for continuum fit
igeogder	for each simulated geometry *j* the highest (*igeotder*(*j*,1)) and lowest (*igeotder*(*j*,2)) parameter level which has to be considered for the vmr-derivatives
igeocder	for each simulated geometry *j* the highest (*igeocder*(*j*,1)) and the lowest (*igeocder*(*j*,2)) parameter level which has to be considered for the continuum-derivatives

Module structure
1.Calculation of *igeogder*
2.Calculation of *igeocder*

Detailed description

1.Calculation of *igeogder:*
The highest parameter level influences all simulated geometries:
For $1 \le jgeo \le igeo$:
$igeogder(jgeo,1) = 1$

The lowest parameter level that influences the simulated geometry is the one of the geometry itself, if the geometry is also a parameter level. If the geometry is no parameter level, the parameter level below is used (if it exists).

For $1 \le jgeo \le igeo$:

Count the parameter levels up to *jgeo*:
    mpar=0
    For $1 \le kgeo \le jgeo$:
        if [*lfitgeo(kgeo)*]: *mpar*=*mpar*+1

If the geometry *jgeo* is a parameter level:
    if [*lfitgeo(jgeo)*]: *igeogder(jgeo,2)=mpar*
if *jgeo* is no parameter level:

else:
if *mpar* is not equal to the total number of parameter levels:
if [*mpar* ≠ *ipar*]: *igeogder(jgeo,2)=mpar*+1
if *mpar* is equal to the total number of parameter levels:
if [*mpar = ipar*]: *igeogder(jgeo,2)=mpar*

2.Calculation of *igeocder:*

2.1 Determination of the highest derivative to be calculated for continuum
kcl=0
Begin loop I on limb observations *jcl=1, ..., ilimb*
Begin condition I: if *lfit(jcl)* = TRUE then
kcl=kcl+1
Begin condition II if *jcl > nucl* then
Begin loop II on simulated geometries *kgeo=1, ..., igeo*
igeocder(kgeo,1) = kcl
End loop II on simulated geometries
go to (**)
End if condition II
End if condition I
End loop I on limb observations
(**) continue

2.2 Determination of the lowest derivative to be calculated for continuum
Begin loop I on the simulated geometries: *jgeo = 1, ..., igeo-1*
mpar = 0
Begin loop II on the simulated geometries: *kgeo = 1, ..., jgeo+1*
if *lfitgeo(kgeo)* = TRUE *mpar=mpar+1*
End loop II on the simulated geometries
igeocder(jgeo,2) = mpar
End loop I on the simulated geometries

3.2.7 CHBASE_VMR

Description: Is the same as module CHBASE_PT, but here we set *istart = 0* at step 2 (see Sect.2.2.7).

3.2.8 FAILS_VMR

For the description of this module see section 2.2.8.

3.2.9 GRID_VMR

For the description of this module see section 2.2.9.

3.2.10 GUESSPAR_VMR

GUESSPAR_VMR]
|(----LININT_VMR *
|(----MWCONT_VMR *
|(----FICARRA_VMR *
|(((--BLIND_VMR *

Description: This module builds the initial guess of the vector *rxpar* which contains the parameters that are going to be fitted in VMR retrievals.

Variables exchanged with external modules:

Name	Description
rzbase	rzbase(imxpro) = altitude of the base-levels
rvmrbase	rvmrbase(imxpro,imxgas) = VMR profiles of the different gases
rcbase	rcbase(imxpro,imxmw) = continuum on the base-levels for each MW
ibase	ibase = number of base-levels
nselmw	nselmw = total number of selected microwindows for the retrieval
rzpar	rzpar(imxlmb) = vector of the altitudes where the temperature profile is fitted
ipar	ipar = number of parameter-levels (i.e. N. of elements of rzpar vector)
rztang	rztang(imxgeo) = vector containing the engineering values of tangent altitudes.
ilimb	ilimb = number of measured geometries
lfit	lfit(imxlmb) = These are logical vectors that identify the levels
lokku	lokku(imxgeo,imxmw) = occupation matrix used for the selection of operational MW's for each observation geometry
lparbase	lparbase(imxpro) = logical vector which identifies the altitudes where the T profile is fitted, among the altitudes rzbase.
rperc	rperc = maximum relative (with respect to rconint) distance between central frequencies of two microwindows which are defined as close-close ones for the definition of continuum emission
rconint	rconint(imxlmb,imxmw) = frequency range around each MW, for each sweep tangent altitude, in which the continuum can be considered as varying linearly.
rxpar	rxpar(imxtop) = vector of the fitted parameters
itop	itop = total number of parameters to be fitted
icontpar	icontpar = total number of continuum parameters to be fitted
rjaccon	rjaccon(imxpro*imxmw,imxcop) = jacobian matrix for the derivative of the continuum base-level values with respect to the continuum parameters
isaved	isaved(imxsav) = vector containing all the necessary quantities for the reconstruction of continuum profiles performed by *ficarra* subroutine
dstep	dstep = distance between coarse-wavenumber grid points [cm-1]
nsam	nsam(imxmw) = number of sampling points in each MW (general coarse grid)
ifspmw	ifspmw(imxmw) = index of the first sampling point of each MW * NOTE: the sampling point at frequency=0 has index=1
nucl	nucl = number of limb geometries to be skipped before starting continuum fit; numbering starts from top.
lcfit	lcfit(imxgeo,imxmw) = continuum occupation matrix
lccmat	lccmat(imxgeo,imxmw)= logical matrix identifying altitudes/MWs where the continuum is assumed to be equal to its value at a neighbouring MW,
ifco	Switch for enabling offset or offset and continuum fit.
rpbase	rpbase(imxpro) = pressure at the base-levels

Algorithm Description

Starting from the initial guess of VMR and continuum profiles, the initial guess of the vector *rxpar* of the fitted parameters is evaluated. The total number of fitted parameters (*itop*) and the total number of continuum fitted parameters (*icontpar*) are evaluated as well.

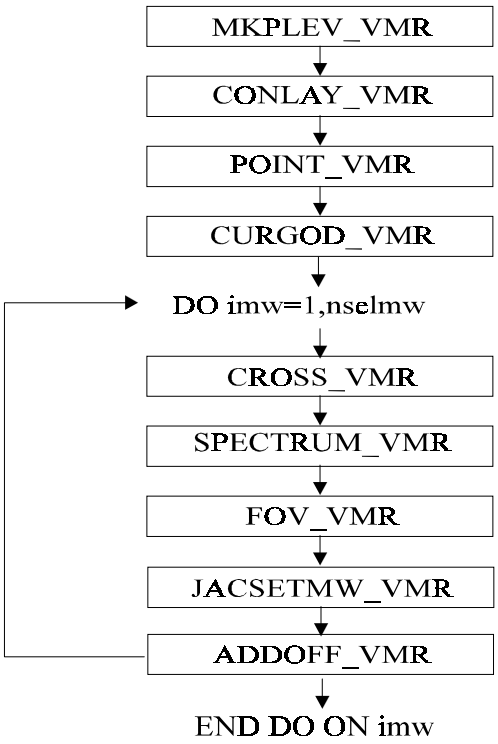
Detailed description

The module proceeds in the following steps:

- For *j=1,..., ipar, rxpar(j)* is set equal to the VMR of the main gas *rvmrbase(k,1)* at the tangent altitudes of the sweeps which correspond to fitted points in the VMR profile. These sweeps are identified, among all the fitted sweeps, by a TRUE element in the vector *lfit.* Sorting is always from top.
- subroutine **MWCONT_VMR** is called: the vector *rcpar* of the continuum parameters to be fitted is coputed as well as *icontpar* that is the total number of continuum fitted parameters. The integer vector *isaved* is also computed.
- subroutine **FICARRA_VMR** is called: it rebuilds continuum profiles starting from the vector *rcpar* and the integer vector *isaved*, it computes also the jacobian matrix *rjaccon* which contains the derivatives of the different points in the continuum base-profiles with respect to the continuum fitted parameters.
- For *j=ipar+1, ..., ipar+icontpar, rxpar(j)* is set equal to *rcpar(j-ipar)*
- For *j=ipar+icontpar+1, ..., ipar+icontpar+nselmw, rxpar(j)* is set equal to zero. Such elements of *rxpar* refer indeed to the instrumental offsets whose initial guess is supposed to be equal to zero.
- *itop* is then computed as:
  if *ifco* = 0 ---> *itop = ipar*
  if *ifco* = 1 ---> *itop = ipar + icontpar*
  if *ifco* = 2 ---> *itop = ipar + icont[ar + nselmw*
- *itop* is then checked and if *itop > imxtop* (*imxtop* is a parameter contained in 'parameters_vmr.inc') a fatal error is produced and the program is stopped.

3.2.11 FWDMDL_VMR

Description
This program module is used to carry-out the folw of the calls reported in the following diagram:



- At the end of this flow the 'model' VMR profile of the main gas of the retrieval is copied in the *rxmod* vector:

  do *j*=1,*ilev*
  rxmod(j)=rmrmod(j,1)
  end do

Variables exchanged with external modules:
Please refer to the calling instruction reported in Sect. 3.1 of the present document.

3.2.11.1 MKPLEV_VMR

MKPLEV_VMR
|(----CHECK]
|((?--CHECK]
|((?--LININT *
|((((-LININT *
|((((+ESPINT *
|((((+GRAVITY *
|((((+GRAVITY *

Description: builds the layering of the atmosphere that allows the calculation of the radiative transfer integral.

Variables exchanged with external modules:

Name	Description
rzsi	rzsi(imxgeo) = tangent altitudes of the geometries to be simulated
igeo	igeo = number of simulated geometries
rzbase	rzbase(imxpro) = altitude of the base-levels
rtbase	rtbase(imxpro) = temperature of the base levels
rpbase	rpbase(imxpro) = pressure on the base-levels
rvmrbase	rvmrbase(imxpro,imxgas) = volume mixing ratio of the gases on the base levels
ibase	ibase = number of base-levels
rulatm	rulatm = upper limit of the atmosphere
rwmolref	rwmolref = molecular weigth of the gas that has been selected as a reference for building the levels.
dsigm0	dsigm0 = Centre frequency of the line selected as a reference for building the levels.
rhw0ref	rhw0ref = half-width of the line selected as a reference for building the levels.
rmaxtv1	rmaxtv1 = max. allowed temperature variation (K) between two neighbouring levels, when the lower level is located below rzt12.
rmaxtv2	rmaxtv2 = max. allowed temperature variation (K) between two neighbouring levels, when the lower level is located above rzt12.
rzt12	rzt12 = altitude (km) where the temperature thresholds rmaxtv1 and rmaxtv2 are exchanged.
rhwvar	rhwvar = max. allowed half-width variation of the selected reference line between two neighbouring levels.
igas	igas = total number of different gases
rexphref	rexphref = exponent for the calculation of Lorentz h-w for the line selected as a reference for building the levels.
rincz	rincz = guess altitude increment (km) used for building the levels above the highest simulated geometry.
redfact	redfact = reduction factor applied to 'rincz' when it produces not

	acceptable P levels above the highest simulated geometry.
rlat	rlat = actual latitude (degrees)
lfitgeo	lfitgeo(imxgeo) = logical vector which identifies the simulations which correspond to a fitted point in the T profile, among all the simulations to be performed.
rzmod	rzmod(imxlev) = heights of model levels used for the radiat. transf. calc.
rpmod	rpmod(imxlev) = pressure on model levels used for the radiat. transf. calc.
rtmod	rtmod(imxlev) = temperature on model levels used for the radiat. transf. calc.
rmrmod	rmrmod(imxlev,imxgas) = volume mixing ratio for each gas considered in actual retrieval on model levels used for rad. tra. calc.
ilev	ilev = number of model levels (for rad. trans. calculation)
itglev	itglev(imxgeo) number of the tangent-level for each geometry
iderlay	iderlay(imxlmb,3) highest (x,1), lowest (x,3) and middle (x,2) (the one directly above the 'derivated' layer) which is affected by each derivative (imxlmb refers to the parameter-levels)
ipar	ipar = number of altitudes where the temperature profile is fitted.

Algorithm Description

Module structure

The module proceeds along the following steps:

1. Building of the levels located between the lowest and the highest simulated geometries
2. building of the levels located above the highest simulated geometry,
3. interpolation of temperature and VMR profiles to the altitude levels generated in steps 1. and 2., determination of pressure at the generated levels,
4. calculation of itglev,
5. calculation of iderlay.

Detailed description

Step 1: Building of the altitude levels located between the lowest and the highest simulated geometries.

First of all the altitudes rzsi (tangent altitudes of the simulated geometries) are taken as levels (main-levels). Then, starting from low altitudes (from rzsi(1)) the couples of neighbouring main-levels are processed by CHECK subroutine which establishes, by checking pre-defined criteria, whether the two considered levels can be accepted. If the two levels are accepted, then the next couple of levels is checked, otherwise sub-levels are generated using the following procedure:

Let's call rz1 and rz2 the altitudes of the two considered main-levels that cannot be accepted,

isublev = 1 (sub-level index)

(**) logic = .TRUE.

start loop on sub-layers lying between the two considered main-levels: j=1, ...,isublev+1

rz1n = rz1 + (j-1)*[(rz1-rz2)/(isublev+1)]

rz2n = rz1 + j*[(rz1-rz2)/(isublev+1)] (Generated sub-level)

CHECK the couple of levels rz1n, rz2n: the result of CHECK module is stored in the logical variable lcheck

logic = logic .and. lcheck
end loop
if logic = .TRUE. then accept the generated sub-levels and proceed to consider the next couple of main-levels.
if logic = .FALSE. then set isublev = isublev + 1 and proceed to (**).

After this step, each tangent altitude of the simulated geometries has associated its own level; furthermore, equispaced levels can exist between the tangent altitudes. The generated sub-levels allow to properly model the atmosphere also in the regions where the atmospheric properties have a large variation in the scale of the distance between the tangent altitudes of two neighbouring simulated geometries.

Step 2: building of the levels located above the highest simulated geometry.
Let's start from the tangent altitude of the highest simulated geometry rzsi(igeo). New levels located above rzsi(igeo)are generated using the following algorithm:

rz1 = rzsi(igeo)
(+) rz2 = rz1 + rincz
rz1 and rz2 are then processed by CHECK:
if the check is successful then:
rz2 is accepted as a new level,
rz1 = rz2,
rincz is set equal to its initial value,
ifails = 0,
proceed to (+).
else:
ifails = ifails +1
rincz = rincz / (redfact * ifails)
proceed to (+)
end if

The above procedure is stopped when the new generated level is higher than the upper limit of the atmosphere increased of the value of rincz (rulatm+rincz); the level higher than (rulatm+rincz) is not included in the generated set of levels, but the check with (rulatm+rincz) instead of rulatm assures that the layering is built on the overall range of the atmosphere to be considered.
All the obtained levels are then sorted starting from high altitudes and recorded in the vector rzmod(imxlev); the total number of levels is recorded in the variable ilev.

General remark: during steps 1 and 2, whenever a new level is built, the total number of generated levels is checked and if this number is greater than the parameter imxlev then, the thresholds rmaxtv1, rmaxtv2 and rhwvar are multiplied by a factor 1.1 and the procedure is restarted from step 1.. Before exiting, the procedure restores the initial values of the thresholds, so that next time the module is called, the right value of the thresholds is used. At the last call of the forward model, if different thresholds, with respect to the user-defined ones are used, a warning is produced by the main program. This feature avoids the production of a large number of levels during the iterations of the retrieval, when the temperature profile can be really distorted with respect to its real shape, and no very accurate simulations are required.

Step 3: interpolation of temperature and VMR profiles to the altitude levels generated in steps 1. and 2. Computation of pressure at the altitudes rzmod.
The following operations are performed at this step:

- For each altitude of the vector rzmod, the corresponding temperature rtmod is obtained by linear interpolation (in the altitude domain, using LININT) between the elements of temperature profile rtbase(imxpro) which are referred to the altitudes rzbase(imxpro).
- For each altitude of the vector rzmod, the corresponding VMR of all the considered gases rvmrmod(imxlev,imxgas) is obtained by linear interpolation (in the altitude domain, using LININT) between the elements of VMR profiles rvmrbase(imxpro,imxgas) which are referred to the altitudes rzbase(imxpro).
- Rebuilding of pressure profile i.e. calculation of pressure at the rzmod altitudes.

The pressure corresponding to the lowest level rzmod(ilev) is computed using exponential interpolation (ESPINT) from the pressure profile rpbase(imxpro) which is referred to the altitudes rzbase(imxpro).
The subsequent elements of rpmod are then computed using hydrostatic equilibrium law, as follows:

$$rpmod(i-1) = rpmod(i) * \exp\left[- \frac{rmovr * gravity(\bar{z}_i, rlat) * \Delta z_i}{\bar{t}_i} \right]$$

where:

$\bar{z}_i = [rzmod(i-1) + rzmod(i)] / 2$
$\Delta z_i = rzmod(i-1) - rzmod(i)$
$\bar{t}_i = [rtmod(i-1) + rtmod(i)] / 2$
rmovr is a parameter, (see description of parameters.inc)
$gravity(\bar{z}_i, rlat)$ is computed by 'GRAVITY' function,
i ranges from ilev, ..to..., 2.

Step 4: calculation of itglev.
itglev is an integer vector which indicates, for each simulated geometry, the index of the tangent level. itglev(i) = j means that the tangent level of the i-th simulated geometry is the level number j; remember that the numbering of the levels starts from high altitudes.

Step 5: calculation of iderlay.
For each fitted parameter we calculated first iderlev which is defined as:
iderlev(j,1) is the first (highest) level where the VMR profile of the main gas is modified due to the variation of the j-th fitted point z
iderlev(j,3) is the last (lowest) level where the VMR profile of the main gas is modified due to the variation of the j-th fitted point
iderlev(j,2) is the central level where the VMR profile of the main gas is modified due to the variation of the j-th fitted point; i.e. iderlev(j,2) = itglev(I(j)).
Afterwards iderlay is computed as herewith described:

```
begin loop on perturbations j=1,ipar
        iderlay(j,2)=iderlev(j,2)-1
end loop on perturbations

iderlay(1,1) = 1
iderlay(1,3) = iderlay(2,2)

loop on perturbations j=2,ipar-1
```

```
        iderlay(j,1) = iderlay(j-1,2)+1
        iderlay(j,3) = iderlay(j+1,2)
end loop on perturbations

iderlay(ipar,1) = iderlay(ipar-1,2) + 1
iderlay(ipar,3) = ilev - 1          .
```

3.2.11.2 CHECK_VMR

CHECK_VMR]
|?----PTFROMZ_VMR *

Description: This module is used by **MKPLEV_VMR** to check whether two neighbouring levels can be accepted. This is done by evaluating the temperature and the Voigt line-width variation for a selected reference line, going from one level to the other.

Variables exchanged with external modules:

Name:	Description:
rz1	rz1 = altitude (km) of the first considered level
rz2	rz2 = altitude (km) of the second considered level
rtprof	rtprof(imxpro) = actual temperature profile
rpprof	rpprof(imxpro) = actual pressure profile
rzprof	rzprof(imxpro) = altitudes to which rtprof and rpprof are referred.
ipro	number of elements in the profiles rpprof, rtprof, rzprof
rwmolref	rwmolref = molecular weigth of the gas that has been selected as a reference for building the levels.
dsigm0	dsigm0 = Centre frequency of the line selected as a reference for building the levels.
rhw0ref	rhw0ref = half-width of the line selected as a reference for building the levels.
rmaxtv1	rmaxtv1 = max. allowed temperature variation (K) between two neighbouring levels, when the lower level is located below rzt12.
rmaxtv2	rmaxtv2 = max. allowed temperature variation (K) between two neighbouring levels, when the lower level is located above rzt12.
rzt12	rzt12 = altitude (km) where the temperature thresholds rmaxtv1 and rmaxtv2 are exchanged.
rhwvar	rhwvar = max. allowed half-width variation of the selected reference line between two neighbouring levels.
lcheck	lcheck = logical variable output of the module. If lcheck is returned TRUE the checks have been successful
rexphref	rexphref = exponent for the calculation of Lorentz h-w for the line selected as a reference for building the levels.
rlat	rlat = actual latitude (degrees)
lfirstcall	lfirstcall = logical variable that indicates whether this is the first time that CHECK module is called in the current run of MKPLEV_PT.

Detailed description

The checks proceed along the following steps:

- The internal variable *rmaxtv* is set equal to *rmaxtv1* if *rz1* < *rtz12* otherwise *rmaxtv* = *rmaxtv2*.

- The variables *lcheck, lcheck1* and *lcheck2* are initialised to TRUE.

- Temperature and pressure corresponding to the altitudes *rz1* and *rz2* are evaluated using **PTFROMZ_VMR** module. The temperatures of the two levels *rz1* and *rz2* are stored respectively in the variables *rtemp1* and *rtemp2*, while the pressures are stored respectively in the variables *rpres1* and *rpres2*.

- The temperature variation between the two levels is checked:
  if $|rtemp1 - rtemp2| < rmaxtv$ then set *lcheck1* = FALSE

- Doppler and Lorentz half-widths *rhwd, rhwl* of the seleced reference line are then evaluated at the two levels:
  *rhwd1=dsigm0*3.581047d-7**sqrt*(rtemp1/rwmolref)*
  *rhwd2=dsigm0*3.581047d-7**sqrt*(rtemp2/rwmolref)*
  rhwl1=rhw0ref(rpres1/rp0h)*(rt0h/rtemp1)**rexphref*
  rhwl2=rhw0ref(rpres2/rp0h)*(rt0h/rtemp2)**rexphref*

- The Voigt half-widths *rhwv* are then given by:
  *rhwv1 = 0.5 * rhwl1 * 1.0686215708754 +* sqrt*(rhwl1*rhwl1*0.216866444 + rhwd1*rhwd1)*
  *rhwv2 = 0.5 * rhwl2 * 1.0686215708754 +* sqrt*(rhwl2*rhwl2*0.216866444 + rhwd2*rhwd2)*

- The ratio *rhwrat* between the two half widths is then:
  rhwrat = abs(rhwv2 / rhwv1)

- The check on the half-widths is then performed:
  if *rhwrat* < 1 then set *rhwrat*=1./*rhwrat*
  if *rhwrat* > *rhwvar* then *lcheck1* = FALSE

- The result of the check is then stored in *lcheck*:
  lcheck = lcheck1 .and. *lcheck2*.

3.2.11.3 JACSETMW_VMR

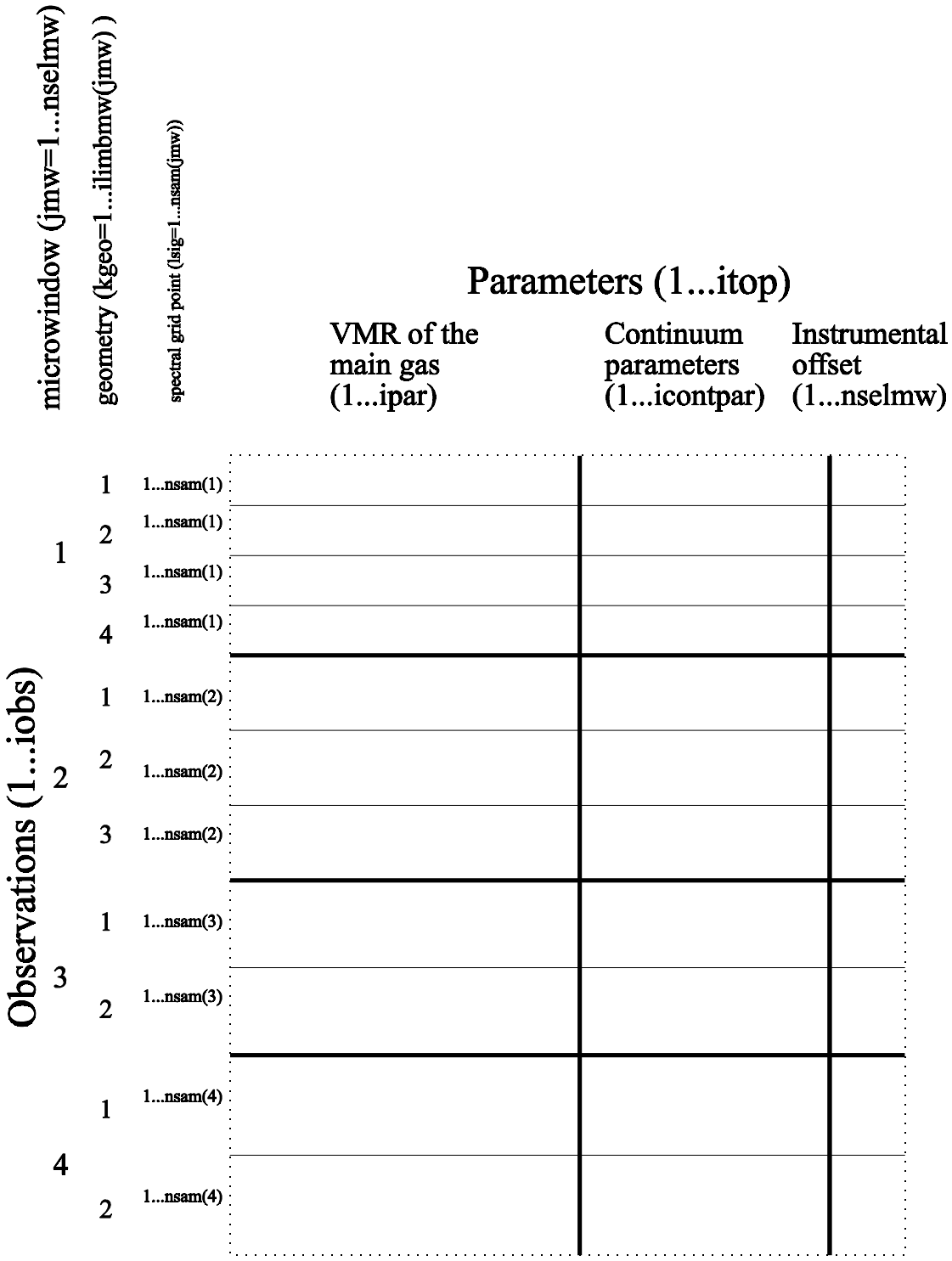
Description
For the actual microwindow the VMR derivatives are written into the jacobian matrix.
The derivatives of the specta with respect to the fitted continuum parameters are calculated by multiplication of the derivatives 'rcderfov' (with respect to the parameter-levels) with the derivatives 'rjaccon' of the continuum on the parameter-levels with respect to the fitted continuum parameters

Variables exchanged with external modules:

Name	Description
imw	number of the actual microwindow
ilimbmw	number of valid measured geometries per microwindow
ipar	number of parameter-levels
nsam	number of sampling points in each Mw (general coarse grid)
lokku	occupation matrix used for the selection of operational Mw's for each observation geometry
nucl	nucl+1 = upper parameter level for continuum fit
ilimb	number of measured geometries
rgderfov	derivate with respect to vmr after fov convolution
rcderfov	derivate with respect to continuum after fov convolution
icontpar	total number of continuum parameters to be fitted
rjaccon	jacobian matrix for the derivative of the continuum parameter-level values with respect to the continuum parameters
irowmw	the row of the Jacobian matrix where the actual mirowindow starts
rjacob	Jacobian Matrix 1st index: observations 2nd index: parameters
lparbase	lparbase(imxpro) = logical vector which identifies the altitudes where the T profile is fitted, among the altitudes rzbase.
ibase	ibase = number of base-levels

Module structure
1. Writing the VMR derivatives into the Jacobian matrix
2. Multiplication of the 'local' continuum derivatives with the continuum jacobian matrix and writing the result into the Jacobian matrix
3. Writing the instrumental offset derivatives into the Jacobian matrix

Detailed description
Before describing the single steps of the code we give an overview of the structure of the Jacobioan matrix which is the matrix of the derivatives of all observations with respect to all parameters:



1. Writing the vmr derivatives into the Jacobian matrix:

For all parameter levels $1 \leq jpar \leq ipar$:

The actual column of the parameters for the temperature derivatives is:
lcol=jpar
and the starting row is:
lrow=irowmw(imw)

Then for all geometries $1 \leq kgeo \leq ilimbmw(imw)$ and all frequency grid points $1 \leq lsig \leq nsam(imw)$:

rjacob(lrow,lcol)=rgderfov(lsig,kgeo,jpar)
lrow=lrow+1

3. Multiplication of the 'local' continuum derivatives with the continuum jacobian matrix and writing the result into the Jacobian matrix:

Begin loop I on continuum parameters: *jpar=1,...,icontpar*
lcol=ipar+jpar
lrow=irowmw(imw)
Begin loop II on the goeometries of the current MW: *kgeo=1,..,ilimbmw(imw)*
Begin loop III on frequency: *lsig=1, ..., nsam(imw)*
m3=0.
r1=0.
Begin loop IV on the 'base' levels: *m1=1, ..., ibase*
if (*lparbase(m1)*)then
m3 = m3 + 1
*m2=(imw-1)*ibase+m1*
*r1=r1+rcderfov(lsig,kgeo,m3)*rjaccon(m2,jpar)*
end if
End loop IV on the 'base' levels
rjacob(lrow,lcol)=r1
lrow=lrow+1
End loop III on frequency
End loop II on geometries of the current MW
End loop I on continuum parameters

4. Writing the instrumental offset derivatives into the Jacobian matrix:
The derivatives with respect to the instrumental continuum are equal to 1.
The column where the derivatives are written for the actual Mw is:
lcol=ipar+icontpar+imw
The starting row is: *lrow=irowmw(imw)*
Then, for all geometries $1 \leq kgeo \leq ilimbmw(imw)$ and all frequency grid points $1 \leq lsig \leq nsam(imw)$:

rjacob(lrow,lcol)=1
lrow=lrow+1

3.2.11.4 CURGOD_VMR

```
CURGOD_VMR
   |-----DREFIND_VMR >
   |((---QSIMP6
         |-----DREFIND_VMR
         |----DFUNC1_VMR
               |?-----DLIM_VMR
               |?-----GRAVITY
         |-----TRAPZ6_VMR]
               |(----PTNMRFROMZ_VMR *
                     |-----DREFIND_VMR
               |----DFUNC1_VMR
                     |?-----DLIM_VMR
                     |?-----GRAVITY
```

Description

This subroutine performs the ray-tracing for the different observation geometries (sweep) and calculates:

1. for all the pairs geometry-layer:
- the column of all the gases (*rcol*) that have to be taken in account in the actual retrieval
- the air column (*raircol*)
- the length of the optical path (*ropath*) in the layer
- the derivative of the main gas column with respect to VMR (in ppm);

2. and, only for a sub-set of the possible 'paths', the IAPT-numbers (see subroutine point):
- the equivalent pressure (in Curtis-Godson meaning) (*rpeq*) for all the gases (IAP)
- the equivalent temperature (*rteq*) for all the gases (IAT)

For some explanations of the reasons of the choices implemented in this module, refer to *T.N. on 'High Level algorithm definition and physical and mathematical optimisations' (TN-IROE-RSA9601),* sect. 6.1 and 6.2.

Variables exchanged with external modules

Name	Description
igeo	Total number of simulated geometries
ipar	Total number of altitudes where the temperature profil is fitted
ilev	Total number of atmospheric levels
itglev	Vector that associates to each geometry, the corresponding number of the tangent level.
iderlay	iderlay(imxlmb,3): highest (x,1), lowest (x,3) and middle (x,2) (the one directly above the 'derivated' layer) layer which is affected by each derivative (imxlmb refers to the parameter levels)
ipoint	Matrix of IAPT-number

igas	Total number of gases in the selected MW
rpmod	rpmod(imxlev): pressure on levels used for the radiat. transf. calc.
rtmod	rtmod(imxlev): temperature on levels used for the radiat. transf. calc.
rzmod	rzmod(imxlev): heights of levels used for the radiat. tranf. calc.
rmrmod	rmrmod(imxlev,imxgas): volume mixing ratio for each gas considered in actual retrieval on levels used for rad. transf. calc.
rearad	earth radius
rlat	latitude of the actual limb-scan [deg.]
deps	degree of accuracy required for the calculation of Curtis-Godson integrals
rpeq	rpeq(imxpat,imxgas) implemented atmospheric (equivalent) pressures (IAPs)
rteq	rteq(imxpat,imxgas) implemented atmospheric (equivalent) temperatures (IATs)
rcol	rcol(imxlay,imxgeo,imxgas) columns for each layer, each geometry and each gas
raircol	raircol(imxlay,imxgeo) air-column for each layer and each geometry
ropath	ropath(imxlay,imxgeo) optical path lenght for each layer, each geometry
rpartgder	rpartgder(imxlay,imxgeo,2) derivatives of the main gas column with respect to VMR [molec/(cm^2*ppm)] _rpartgder(x,y,1): derivative with respect to the value of VMR on the parameter-level below the layer; _rpartgder(x,y,2): derivative with respect to the value of VMR on the parameter-level above the layer
rtmain	rtmain(imxpat) Curtis-Godson equivalent temperature (IAT) of the main gas

Module structure:

1. Initialisation of some variables.

Begin loop 1 over all the simulated geometries

2. Calculation of the number of layers corresponding to the considered geometry, tangent altitude and Snell's law constant.

Begin loop 2 over layers of the actual geometry

3. Check if the actual combination geometry-layer ('path') corresponds to a new IAPT number.

4. Preparation of the inputs for subsequent calculation of the integrals

Begin loop 3 over retrieval parameters

5. Determination of the parameters that affect the column of the actual 'path' and calculation of the heights of corresponding levels.

End loop 3

Begin loop 4 over all the gases in selected microwindows

6. Definition of VMR of the actual gas on the boundaries of the layer

7. Calculation of the following integrals:

- gas column for all the 'paths' (combination of layer and geometry);
- air column and path length for all the paths (this calculation is independent on the gas, so it is performed only once in the do-loop on the gases);
- IAPTs.
- derivative of the main gas column with respect to VMR on the parameter-level above the layer

8. Calculation and storage of required quantities and determination of the main gas equivalent temperature

End loop 4

End loop 2

End loop 1

Begin loop 5 on simulated geometries

Begin loop 6 on the layers above the hightest retrieval parameter

9. Calculation of the derivatives of the columns with respect to VMR for these particular layers

End loop 6

Begin loop 7 on the layers below the lowest retrieval parameter

10. Calculation of the derivatives of the columns with respect to VMR for these particular layers

End loop 7

End loop 5

Detailed description:

1. Initialisation of variables

The pointer of matrix *ipoint, iponew,* is set equal to 0 (see 3.).

loop 1 over all the simulated geometries

kgeo=igeo→1

We start from the lowest geometry (igeo) in order to read matrix ipoint in the appropriate way.

2. Calculation of some variables.

_Determination of the number of layers for the actual geometry (*ilayge=itglev(kgeo)-1*).

_Determination of the tangent altitude referred to the surface of the earth r_T (*dtanal=rzmod(ilayge+1)*) and to the centre of the earth R_T (*dtan_0=rearad+dtanal*).

(In the following *R* will indicate a particular altitude referred to the centre of the earth, *r* the same altitude referred to the surface of the earth.)

_Calculation of the constant in Snell's law:

$$dsnellc = R_T \cdot n(r_T).$$

The refractive index at altitude *r* is a function of pressure *p* and temperature *T* at that altitude.

It is calculated by the function **drefind_vmr(***T,p).*

loop 2 over all the layers to be considered for each geometry

lay=1→*itglev(kgeo)-1*

3. Check if the actual combination geometry-layer ('path') corresponds to a new IAPT-number

For the actual combination geometry-layer, a check is performed in order to establish whether equivalent pressure and temperature have to be calculated or not.

Only if the IAPT-number *ipoint (lay,kgeo)* is greater than *iponew*, the pointer *iponew* is updated to the value of *ipoint (lay,kgeo)* and the logical variable *lflag* passes from false to true.

4. Preparation of the inputs for subsequent integration

This section prepares the inputs to module **qsimp6_vmr** that will compute the integrals.

dalay=rzmod(lay+1), dta=rtmod(lay+1), dpa=rpmod(lay+1) and *dblay=rzmod(lay), dtb=rtmod(lay), dpb=rpmod(lay)* are respectively the heights (referred to the surface of the earth), the temperatures and the pressures on the lower and higher boundary of the layer.

The integration variable *x* used for subsequent integrals is: $x = \sqrt{R^2 - R_T^2}$.

So, the heights referred to the centre of the earth of the boundaries of the layer (*da_0* and *db_0*) are used for calculating the limits of integration for the actual layer:

$dxa = \sqrt{da\_0^2 - dtan\_0^2}$ and $dxb = \sqrt{db\_0^2 - dtan\_0^2}$.

loop 3 over retrieval parameters

jpar=2→ipar

5. Determination of the parameters that affect the column of the actual 'path' and calculation of the heights of corresponding levels.

Since VMR behaviour inside each layer is obtained by making an interpolation from the values of the VMR on the two parameter levels between which the layer is located, the column corresponding to a particular 'path' is affected by the change of VMR only on the two near parameter levels. Besides, since VMR is interpolated linearly, so that the main gas column is given by the following expression:

$$column = \int_{dalay}^{dblay} \left(X_1(dzdown) + \frac{X_1(dzup) + X_1(dzdown)}{dzup - dzdown} \cdot (z - dzdown) \right) \cdot \frac{P(z)}{T(z)} \cdot \frac{ds}{dz} \cdot dz$$

(for clearness we have assumed that the integration variable is the altitude z instead of the real integration variable x),

the derivatives of the column with respect to VMR are given by:

$$\frac{\partial column}{\partial X_1(dzup)} = \int_{dalay}^{dblay} \left(\frac{z - dzdown}{dzup - dzdown} \right) \cdot \frac{P(z)}{T(z)} \cdot \frac{ds}{dz} \cdot dz$$

$$\frac{\partial column}{\partial X_1(dzdown)} = \int_{dalay}^{dblay} \left(1 - \frac{z - dzdown}{dzup - dzdown} \right) \cdot \frac{P(z)}{T(z)} \cdot \frac{ds}{dz} \cdot dz = daircoll - \frac{\partial column}{\partial X_1(dzup)}$$

(for the meaning of daircoll see below).

We have obtained that, since *daircoll* is computed, only the calculation of the derivative of the column with respect to the VMR on the parameter-level just above the layer is necessary, the other being derivated by this one.

Given the layer *lay*, the parameter-level *jpar* is determined so that:

$$iderlay(jpar - 1,\ 2) < lay \le iderlay(jpar,\ 2).$$

When the previous inequality is verified, the logical variable *lder* passes from false to true and the heights corresponding to the parameter-levels between which the layer is located are determined:

$$dzdown = rzmod\left(iderlay\left(jpar,\ 2\right)+1\right),\quad dzup = rzmod\left(iderlay\left(jpar-1,\ 2\right)+1\right).$$

For some layers it can happen that this inequality is never verified, in this case customised calculations are performed in sections 9. and 10.

loop 3 over the all the gases in the selected microwindows
jgas=1→*igas*

6. Definition of the VMR of the gas on the boundaries of the layer
dmra=rmrmod(lay+1,jgas) and *dmrb=rmrmod(lay,jgas)* are the VMRs of the actual gas on the boundaries of the layer.

7. Calculation of the equivalent values by means of integration along the line of sight.
The module **qsimp6_vmr (***dalay, dxa, dblay, dxb, dta, dtb, dpa, dpb, dmra, dmrb, dsnellc, dtan_0, rearad, rlat, deps, dcoll, daircoll, dopathl, dtl, dpl, ddercol, jgas, lflag, lder, dzdown***)** performs the calculation of all the required integrals.

8. Calculation and storage of required quantities and set-up of the main gas equivalent temperatures
The column (in number of molecules per square centimeter) of the actual path is finally calculated and stored:

$$rcol\left(lay, kgeo, jgas\right) = dcoll \cdot rk \cdot 10^{-6},$$

rk is a parameter contained in the file 'parameters.inc' and the factor 10^{-6} is due to the fact that the VMRs are read from input in parts per million (ppm).
If logical variable *lflag* is true, the equivalent pressure (in mbar) and temperature (°K) are normalised and stored:

$$rpeq\left(iponew, jgas\right) = \frac{dpl}{dcoll}\ ,$$

$$rteq\left(iponew, jgas\right) = \frac{dtl}{dcoll}$$

If *jgas*=1, then

the air column, the path-length, the temperature of the main gas, the derivative of the main gas column with respect to VMR are calculated.

In fact, being the air column (in number of molecules per square centimeter) and the path length (in km) independent on the gas, inside the loop on the gases they have to be calculated only once (*jgas=1*).

$$raircol = daircoll \cdot rk\ ,$$
$$ropath(lay, kgeo) = dopathl\ ,$$
$$rtmain(iponew) = rteq(iponew, jgas)$$

Besides, since the main gas has always local code equal 1 in each retrieval, the temperature of the main gas and the derivatives of the main gas column with respect to VMR are calculated only in this case.

$$rtmain(iponew) = rteq(iponew, jgas)$$

If *lder* is true, then
the derivative of the column with respect to VMR on the
parameter-level just above the layer and to the VMR on the
parameter-level just below the layer are calculated (in molecules per square centimeter and per ppm) and stored respectively in *rpartger(lay,kgeo,2)* and *rpartger(lay,kgeo,1):*

$$rpartger(lay,kgeo,2) = \frac{ddercol \cdot rk \cdot 10^{-6}}{dzup - dzdown},$$

$$rpartger(lay,kgeo,1) = raircol \cdot 10^{-6} - rpartger(lay,kgeo,2)$$

Loop 5 on simulated geometries
kgeo=1 $\rightarrow$ *igeo*

Loop 6 on the layers above the hightest retrieval parameter
lay=1 $\rightarrow$ *iderlay(1,2)*

9. Calculation of the derivatives of the columns with respect to VMR in this particular case
The derivatives of the column corresponding to layers above the highest parameter-level (*jpar*=1) and below the lowest parameter-level (*jpar=ipar*) require customised computations, because on the levels above the highest parameter-level and below the lowest parameter-level the value of VMR is scaled according to the variation-factor of the VMR respectively on the highest parameter-level and the lowest parameter-level.
The derivative of the column with respect to the highest parameter level is simply given by:

$$rpartgder(lay, kgeo, 1) = \frac{rcol(lay,\ kgeo,\ 1)}{rmrmod\left(iderlay(1,\ 2)+1,\ 1\right)},$$

while *rpartger(lay,kgeo,2)* is set to 0.

Loop 7 on the layers below the lowest retrieval parameter
lay=iderlay(ipar,2)+1 $\rightarrow$ *itglev(kgeo)-1*

10. Calculation of the derivatives of the columns with respect to VMR in this particular case
The derivative of the column corresponding to the layers below the lowest parameter-level with respect to the lowest parameter level is simply given by:

$$rpartgder(lay, kgeo, 2) = \frac{rcol(lay,\ kgeo,\ 1)}{rmrmod\left(iderlay(ipar,\ 2)+1,\ 1\right)},$$

while *rpartger(lay,kgeo,1)* is set to 0.

3.2.11.5 QSIMP6_VMR & TRAPZ6_VMR

|((---QSIMP6_VMR
|-----DREFIND_VMR +
|-----DFUNC1_VMR >

```
|-----TRAPZ6_VMR
|       |(----SQRT *
|       |(----PTNMRFROMZ_VMR
|       |       |-----DREFIND_VMR +
|       |(----DFUNC1_VMR >
```

Description

Starting from:

- the limits of integration *dxa* and *dxb*,
- the value of temperature, pressure (and consequently of refractive index) and VMR of the actual gas on the boundaries of the layer,
- the interpolation law in altitude of all these quantities inside the layer,

these two modules can calculate six different numerical integrals: *dcoll, dpl, dtl, daircoll, dopathl* and *ddercol*. According to the value of the logical variables *lflag* and *lder*, some of them are not calculated.

Variables exchanged with external modules

Name:	**Description:**
dalay	altitude of the lower boundary of the layer
dxa	lower limit of integration
dblay	altitude of the higher boundary of the layer
dxb	higher limit of integration
dta	temperature corresponding to the lower boundary of the layer
dtb	temperature corresponding to the higher boundary of the layer
dpa	pressure corresponding to the lower boundary of the layer
dpb	pressure corresponding to the higher boundary of the layer
dmra	VMR corresponding to the lower boundary of the layer
dmrb	VMR corresponding to the higher boundary of the layer
dsnellc	Snell's law constant
dtan_0	tangent altitude referred to centre of the earth
rearad	earth radius
rlat	latitude
deps	required accuracy for the integrals calculation
dcoll	returned column of this path (to be moved to the choisen measurement units)
daircoll	returned air density (to be multiplied by parameter rk)
dopathl	returned path lenght (in km)
dtl	returned equivalent temperature (to be normalised)
dpl	returned equivalent pressure (to be normalised)
ddercol	returned derivative of the main gas column with respect to VMR
jgas	actual gas number (local code)
lflag	logical variable: only when it is true, the equivalent pressure and temperature have to be calculated
lder	logical variable: it is true when the not-perturbed profils are considered. Only when it is true the air column and the path lenght

	have to be calculated.
dzdown	height of the parameter level just below the considered layer

Module structure:
See 'Numerical Recipes in FORTRAN' [RD2], pag. 130-133.

Detailed description:
The structure of this module is exactly the same of the one reported on 'Numerical Recipes in FORTRAN', pag. 130-133, with the exception that more than one integral (a maximum of six integrals) are calculated at the same time.
In particular, the integrals are computed in numerical way using Simpson rule: in the implemented method, the trapezoidal rule is refined until a specified degree of accuracy *deps* has been achieved.

The integrals calculated by **qsimp6_vmr** and **trapz6_vmr** modules are the following:

1. $$dcoll = \int_{dxa}^{dxb} X_{gas}\left(r(x)\right) \cdot \frac{p\left(r(x)\right)}{T\left(r(x)\right)} \cdot \frac{ds}{dx} \cdot dx$$

2. $$dpl = \int_{dxa}^{dxb} X_{gas}\left(r(x)\right) \cdot \frac{p^2\left(r(x)\right)}{T\left(r(x)\right)} \cdot \frac{ds}{dx} \cdot dx$$

3. $$dtl = \int_{dxa}^{dxb} X_{gas}\left(r(x)\right) \cdot p\left(r(x)\right) \cdot \frac{ds}{dx} \cdot dx$$

4. $$daircoll = \int_{dxa}^{dxb} \frac{p\left(r(x)\right)}{T\left(r(x)\right)} \cdot \frac{ds}{dx} \cdot dx$$

5. $$dopathl = \int_{dxa}^{dxb} \frac{ds}{dx} \cdot dx$$

6. $$ddercol = \int_{dxa}^{dxb} \left(r(x) - dzdown\right) \cdot \frac{p\left(r(x)\right)}{T\left(r(x)\right)} \cdot \frac{ds}{dx} \cdot dx$$

In these integrals the integration variable *x*, is given by:

$$x = \sqrt{R^2 - R_T^2},$$

$\frac{ds}{dx} \cdot dx$ represents the increment along the line of sight *s*, refractive index dependent, $X_{gas}\left(r(x)\right)$, $p\left(r(x)\right)$, $T\left(r(x)\right)$ represent respectively the gas VMR, pressure and temperature behaviour as a function of the integration variable *x*.

The values of pressure, temperature, VMR, refractive index at a particular height *r* corresponding to the particular *x* is computed by the module
ptnmrfromz_vmr (*dz1, dalay, dta, dpa, dmra, dblay, dtb, dpb, dmrb,* <u>*dt1*</u>*,* <u>*dp1*</u>*,* <u>*dmr1*</u>*,* <u>*drefind1*</u>**)**.

The function **dfunc1_*vmr*(***dx*,*drefr*,*dsnellc* ,*dtan_0*,*dalay*,*dta*,*dpa*,*dblay*,*dtb*,*rlat***)** calculates the value of $\frac{ds}{dx}$ at the altitude r.

The calculation of some of the integrals is bound to the value of the logical variables *lflag* and *lder*. In particular, integral no.1 is always calculated, integrals no. 2 and 3 are calculated only if logical variable *lflag* is true, integrals no. 4 and 5 are calculated only if *jgas* is equal to 1 and integral no. 6 is calculated only if *jgas*=1 and *lder* is true, so the relative outputs of this module have meaning only when these conditions are verified.
Actually, in order to reduce the number of 'if-conditions', the calculation are performed for all the integrals, while the accuracy criteria are checked only for the required integrals.

3.2.11.6 DFUNC1_VMR

For the description of this function, see 2.2.11.7.

3.2.11.7 DLIM_VMR

For the description of this function, see 2.2.11.8.

3.2.11.8 DREFIND_VMR

For the description of this function, see 2.2.11.9.

3.2.11.9 PTNMRFROMZ_VMR

For the description of this function, see 2.2.11.10.

3.2.11.10 FOV_VMR

```
FOV_VMR
   |((((+MAX0 *
   |((((+MIN0 *
   |((((+ABS *
   |((((+FOV3_VMR
   |       |(----INTCON >
   |((((+FOV5_VMR
   |       |(----INTCON >
   |-----FOV4_VMR
           |(----INTCON >
```

Description

Field of View convolution is performed in analytical way by determining first the spectrum, as a function of altitude, interpolated between a given number of spectra simulated at different discrete tangent heights, then convolving this function with the antenna pattern of the field of view FOV(z) (a symmetrical trapezium-shape function with greater base equal to *rbase* and the half-difference between the two bases equal to *rsl*):

$$S^F(\sigma, z) = S\left(\sigma,\ p(z)\right) * FOV(z).$$

Since interpolation is performed by determining the polynomial that passes through a given number of points, the order of the polynomial is dependent on the number of spectra at discrete tangent altitudes that are considered.
In general, spectrum interpolation around a particular tangent altitude is performed using a second order polynomial passing through the three spectra corresponding to that altitude and the two real contiguous tangent altitudes (that, nominally, are 3 km distant). But in some particular cases, in presence of sharp variations in VMR, additional simulated spectra between the nominal tangent altitudes are needed to perform a good interpolation, hence interpolation is performed using more than 3 spectra.
All these decisions have been taken in **occusim_vmr** module, the aim of this module is just to define, starting from the matrix *iocsim*, for each tangent altitude in correspondence of which an observation exists, which and how many spectra have to be taken in account for building the interpolation around that altitude.
The procedure used, for each tangent altitude whose spectrum corresponds to an observation, is the following:

1. to see whether the two adiacent geometries are distant more than *rbase* (the interval in which convolution is performed)
2. if 1. is verified, to see if these two tangent adiacent tangent altitudes are symmetrical with respect the central tangent altitude where FOV convolution is required.
3. if 2. is verified, interpolation is performed with those 3 spectra
4. if 2. is not verified, tangent altitudes contiguous to the previous ones are checked, and spectra corresponding to tangent altitudes as symmetrical as possible are choisen: also in this case, interpolation is however performed with 3 spectra.
5. if 1. is not true, the tangent levels next to the ones adiacent to the central altitude are considered: if we are near the borders, an interpolation with a third order polynomial is built (4 spectra), otherwise 5 spectra are considered and a forth order polynomial is determined.

When this choice has been taken, the contribution of the field of view to spectra (*rspct*) corresponding to the observed ones, to the derivatives of the spectrum with respect to continuum (*rspctcder*) and to the derivatives of the spectrum with respect to VMR is computed by calling modules **fov3_vmr, fov4_vmr, fov5_vmr**.

For some explanations of the reasons of the choices implemented in this module, refer to *T.N. on 'High Level algorithm definition and physical and mathematical optimisations', (TN-IROE-RSA9601),* par. 6.6.

Variables exchanged with external modules:

Name	Description
igeo	total number of simulated geometries
iocsim	iocsim(imxgeo,imxmw): occupation matrix for the simulations to be performed = 0 no simulation required, = 1 simulation required without FOV = 2 simulation required with FOV

itglev	itglev(imxgeo): number of the tangent-level for each geometry
rspct	rspct(imxi,imxgeo): low-resolution spectrum without FOV
rspctcder	rspctcder(imxi,imxgeo,imxlmb) convolved continuum derivative spectra for each geometry and each parameter level
rspctgder	rspctgder(imxi,imxgeo,imxlmb): derivatives of the spectrum with respect to VMR without FOV
nsam	nsam(imxmw): n. of sampling points in each MW (coarse grid)
imw	number corresponding to the actual microwindow
rzmod	rzmod(imxlev): heights of levels used for the radiat. tranf. calc.
rbase	greater base of the trapezium that approximates the antenna pattern of FOV
rsl	half-difference between the two bases of the trapezium
igeocder	igeocder(imxgeo,2): for each geometry the highest (x,1) and lowest (x,2) continuum derivative (in the parameter-grid) which has to be calculated
igeogder	igeogder(imxgeo,2): for each simulated geometry the highest (x,1) and lowest (x,2) parameter for which the derivatives of the spectra with respect to VMR are calculated
rspfov	rspfov(imxi,imxgeo,imxmw): simulated spectra corresponding to the different tangent pressures and different microwindows on the frequency coarse grid: (rspct * FOV)
rcderfov	rcderfov(imxi,imxgeo,imxlmb) derivate with respect to continuum after fov convolution
rgpertfov	rgpertfov(imxi,imxgeo,imxlmb) derivatives of the spectrum with respect to VMR after fov-convolution

Module structure:

1. Inizialisation of the output variables and calculation of the area of the trapezium that approximates FOV antenna pattern.

Begin loop 1 on simulated geometries

Begin condition 1: the spectrum simulated at the actual geometry corresponds to an observation for the actual microwindow

2. Definition of the tangent altitude of the actual spectrum and of the four contiguous spectra; evaluation of the distance between upper and lower tangent heights.

Begin condition 2: the distance between two contiguous tangent heights is ≥ rbase

3. Evaluation of the distance between central and upper tangent height (*rdiff1*) and between central and lower tangent height (*rdiff2*); setting of the input variables for fov3_vmr.

Begin condition 3: the upper and lower levels are not symmetrical with respect to the center

Begin condition 4: the upper tangent level is more distant than the lower one with respect to central height

4. Evaluation of the distance between central level and the second tangent level below it.

Begin condition 5: the second tangent level below the central level (if it exists) and the level above it are not symmetrical

5. Warning message

Else condition 5

6. Setting of some of the input variables for fov3_vmr.
End condition 5
Else condition 4
7. Evaluation of the distance between central level and the second tangent level above it.
Begin condition 6: the second tangent level above the central level (if it exists) and the level above it are not symmetrical
8. Warning message
Else condition 6
9. Setting of some of the input variables for fov3_vmr.
End condition 6
End condition 4
End condition 3
10. Interpolation of the spectrum and spectrum derivatives using the three previously determined spectra and convolution with the FOV function.
Else condition 2
Begin condition 7: if both the second tangent levels below the central level and above exist
11. Interpolation of the spectrum and spectrum derivatives using the five previously determined spectra and convolution with the FOV function.
Else condition 7
Begin condition 8: the second tangent levels below the central level does not exist
12. The interpolation is performed with 4 spectra: setting of some of the input variables for fov4_vmr
Else condition 8
13. The interpolation is performed with 4 spectra: setting of some of the input variables for fov4_vmr
End condition 8
14. Interpolation of the spectrum and spectrum derivatives using the four previously determined spectra and convolution with the FOV function.
End condition 7
15. Warning message if a particular condition is verified.
End condition 2
End condition 1
End loop 1

Detailed description:

1. Inizialisation of the output variables and calculation of the area under the trapezium that approximates FOV.

All the elements of matrices *rspfov*, *rcderfov*, *rgderfov* are set equal to 0.
Besides, the variable *iactugeo*, that counts the number of the geometries corresponding to observations in the actual MW, is set equal to 0.

The FOV function is represented by a symmetrical trapezium-shape function whose greater base is *rbase* and the half-difference between the two bases is *rsl*.

Hence, the area under this trapezium function is given by:
rarea=rbase-rsl.

Loop 1 over the simulated geometries
jgeo=2→ igeo -1
Geometry no.1 and no. igeo do not surely correspond to observations.

Condition 1
This condition checks the value of the iocsim matrix for the given MW and the given geometry (remember that this module is located in a loop on all the selected MWs).
If iocsim(jgeo,imw) is equal to 2, it means that the simulated spectrum we are considering corresponds to an observation, hence it will be convolved with FOV function.
If that condition is not verified, all the calculations for taking in account FOV are skipped and the subsequent spectrum is analysed.
Let's consider the case in which iocsim(jgeo,imw) = 2; then:

2. Definition of the tangent altitudes of the actual spectrum and of the four contiguous spectra, above and below; evaluation of the distance between upper and lower tangent heights.
The counter of the observed geometries in the actual MW *iactugeo* is increased of 1 unit.
Determination of the tangent altitude (*rztan=rzmod(itglev(jgeo))*) of the spectrum we are considering.
Determination of the tangent altitudes of the two contiguous spectra: the above spectrum, characterised by the geometry *jgeoup=jgeo-1*, has tangent altitude *rztanup=rzmod(itglev(jgeoup))* and the one below, characterised by the geometry *jgeodown=jgeo+1*, has tangent altitude *rztandown=rzmod(itglev(jgeodown)).*
Determination of the tangent altitudes of the second tangent level, both above and below the considered one. Note that the tangent altitudes of these spectra cannot be defined when we are near the borders: in this case we force their value to the last possible one in order to avoid some if's, but when this condition is not verified they will however never be used as the code will check for their significant.
The second spectrum above the central one, characterised by the geometry *jgeoup2=jgeo-2*, has tangent altitude *rztanup2=rzmod(itglev(max(1,jgeoup2)))* and the second below, characterised by the geometry *jgeodown2=jgeo+2*, has tangent altitude

rztandown2=rzmod(itglev(min(jgeodown2,igeo))).

Calculation of the distance between the tangent altitudes corresponding to geometries *jgeoup* and *jgeodown*: *rdiff=rztanup-rztandown*.

Condition 2:
$rdiff \geq rbase$
In this case the convolution range is all contained between jgeodown and jgeoup: the only problem is that jgeodown and jgeoup can be asymmetrical with respect to jgeo, and this is not good because interpolation is worse in this case.

3. Evaluation of the distance between central and upper tangent height (*rdiff1*) and between central and lower tangent height (*rdiff2*); setting of the input variables for fov3_vmr.
We calculate:
rdiff1=rztanup1-rztan
rdiff2=rdiff-rdiff1

We also set the input variables for fov3_vmr that will be used if condition 3 will result not verified:
iargum4=jgeoup1
rargum5= rztanup1
iargum6=jgeodown1
rargum7= rztandown1

Condition 3:
rdiff2 < 2· rpar or rdiff1 < 2· rpar
and
|rdiff2-rdiff1| > rpar

rpar is a parameter contained in the file 'parameters.inc'

If this condition is verified, either the upper or the lower level represents a tangent altitude corresponding to an observation, the other is an additional geometry. A symmetrical interpolation is preferred.

Condition 4:
rdiff1> rdiff2

4. Evaluation of the distance between central level and the second tangent level below it.
We set:
rdiff2= rztan-rztandown

Condition 5:
$jgeodown2 > igeo \;\; or \;\; |rdiff2 - rdiff1| \geq rpar$

5. Warning message
It is not possible to perform symmetrical interpolation: the interpolation will be performed using the three spectra defined in 3.
Message: 'Asymmetrical interpolation for the lowest geometry'

6. Setting of some of the input variables for fov3_vmr.
Spectrum corresponding to geometry *jgeodown2* is used: the inputs for fov3_vmr different from the ones in 3. are set:
iargum6=jgeodown2
rargum7=rztandown2

7. Evaluation of the distance between central level and the second tangent level above it.
We set:
rdiff1= rztanup2-rztan

Condition 6:
$jgeoup2 < 1 \;\; or \;\; |rdiff2 - rdiff1| \geq rpar$

8. Warning message
It is not possible to perform symmetrical interpolation: the interpolation will be performed using the three spectra defined in 3.

Message: 'Asymmetrical interpolation for the highest geometry'

9. Setting of some of the input variables for fov3_vmr.
Spectrum corresponding to geometry *jgeoup2* is used: the inputs for fov3_vmr different from the ones in 3. are set:
iargum4=jgeoup2
rargum5=rztanup2

10. Interpolation of the spectrum and spectrum derivatives using the three previously determined spectra and convolution with the FOV function.
FOV convolution is performed calling module **fov3_vmr** (*iactugeo, imw, jgeo, rztan, iargum4, rargum5, iargum6, rargum7, rspct, rspctcder, rspctgder, nsam, rbase, rsl, rspfov, rcderfov, rgderfov, igeocder, igeogder, rzc, rarea*)

Condition 7:
$jgeodown2 \le igeo$ and $jgeoup2 \ge 1$
11. Interpolation of the spectrum and spectrum derivatives using the five previously determined spectra and convolution with the FOV function.

FOV convolution is performed calling module **fov5_vmr** (*iactugeo, imw, jgeo, rztan, jgeoup1, rztanup1, jgeodown1, rztandown1, jgeoup2, rztanup2, jgeodown2, rztandown2, rspct, rspctcder, rspctgder, nsam, rbase, rsl, rspfov, rcderfov, rgderfov, igeocder, igeogder, rzc, rarea*)

The distance *rdist* between the tangent altitudes corresponding to *jgeoup2* and *jgeodown2* is then calculated.

Condition 8:
jgeodown2 > igeo
We cannot use the second spectrum below the central one, so we shall use four sets of data.

12. The interpolation is performed with 4 spectra: setting of some of the input variables for **fov4_vmr**
We set the inputs to module **fov4_vmr**
iargum2=jgeodown1
rargum3=rztandown1
iargum4=jgeoup1
rargum5=rztanup1
iargum6=jgeoup2
rargum7=rztanup2
The distance *rdist* between the tangent altitudes corresponding to *jgeoup2* and *jgeodown1* is then calculated.

13. The interpolation is performed with 4 spectra: setting of some of the input variables for **fov4_vmr**
In this case we cannot use the second spectrum above the central one, so the inputs to module **fov4_vmr** are:
iargum2=jgeodown2
rargum3=rztandown2
iargum4=jgeodown1

rargum5=rztandown1
iargum6=jgeoup1
rargum7=rztanup1
The distance *rdist* between the tangent altitudes corresponding to *jgeoup1* and *jgeodown2* is then calculated.

14. Interpolation of the spectrum and spectrum derivatives using the four previously determined spectra and convolution with the FOV function.
FOV convolution is performed calling module **fov4_vmr** (*iactugeo, imw, jgeo, rztan, iargum2, rargum3, iargum4, rargum5, iargum6, rargum7, rspct, rspctcder, rspctgder, nsam, rbase, rsl, rspfov, rcderfov, rgderfov, igeocder, igeogder, rzc, rarea*)

15. Warning message if a particular condition is verified.
If *rdist* < *rbase*, a message is written: 'Warning: extrapolation has been performed'

3.2.11.11 FOV3_VMR

Description: After performing the interpolation in altitude between the spectra at three contiguous tangent altitudes, this module calculates the analytical convolution of the interpolated spectrum with the FOV function. This procedure is repeated for the derivatives of the spectrum with respect to continuum and the derivatives of the spectra with respect to VMR.

Variables exchanged with external modules

Name	Description
iactugeo	local counter of the geometries of the actual MW corresponding to the observations
imw	number of the actual microwindow
jgeo	actual index of simulated spectrum
rztan	tangent altitude of the spectrum of which we are calculating convolution with FOV.
jgeoup	index of the geometry above the considered one
rztanup	tangent altitude corresponding to geometry *jgeoup*
jgeodown	index of the geometry below the considered one
rztandown	tangent altitude corresponding to geometry *jgeodown*
rspct	rspct(imxi,imxgeo): low-resolution spectrum (rsp * ILS)
rspctcder	rspctcder(imxi,imxgeo,imxlmb): the convolved continuum derivative spectra for each geometry and each parameter level
rspctgder	rspctgder(imxi,imxgeo,imxlmb): low resolution spectra for the perturbed temperature profiles
nsam	nsam(imxmw): no. of sampling points in each MW (coarse grid)
rbase	greater base of trapezium-shape that approximates Field of View pattern
rsl	half-difference between the bases of the trapezium (1/rsl gives the slope)
rspfov	rspfov(imxi,imxgeo,imxmw): simulated spectra corresponding to the different tangent pressures and different microwindows on the frequency coarse grid: (rspct * FOV)

rcderfov	rcderfov(imxi,imxgeo,imxlmb) derivate with respect to continuum after fov convolution
rgderfov	rgderfov(imxi,imxgeo,imxlmb) temperature-perturbed spectra after fov-convolution
igeocder	igeocder(imxgeo,2): for each geometry the highest (x,1) and lowest (x,2) continuum derivative (in the parameter-grid) which has to be calculated
igeogder	igeogder(imxgeo,2): for each simulated geometry the highest (x,1) and lowest (x,2) parameter with respect to which the derivatives of the spectra (VMR) are calculated
rzc	altitude in correspondence of which the convolution with the FOV is calculated
rarea	area of the trapezium that approximates antenna pattern of FOV

Module structure:

1. Definition of the vector *rxa* containing the tangent heights of the spectra used for the interpolation
   Begin loop 1 on the frequencies of the actual MW
   2. Definition of the vector *rya* containing the values of the spectra corresponding to *rxa* for the actual frequency.
   3. Interpolation of the spectrum, analytical convolution and normalisation
   4. Storage of the calculated value in the vector *rspfov* .
      Begin loop 2 on the continuum parameters that affect the spectrum corresponding to geometry jgeo
      5. Definition of the vector *rya* containing the values of the derivatives with respect to continuum corresponding to *rxa* for the actual frequency.
      6. Interpolation of the continuum derivatives, analytical convolution and normalisation
      7. Storage of the calculated value in the vector *rcderfov*.
      End loop 2

      Begin loop 3 on the VMR parameters that affect the spectrum corresponding to geometry jgeo
      8. Definition of the vector *rya* containing the values of the derivatives with respect to VMR corresponding to *rxa* for the actual frequency.
      9. Interpolation of the continuum derivatives, analytical convolution and normalisation
      10. Storage of the calculated value in the vector *rgderfov*.
      End loop 3
   End loop 1

Detailed description

1. Definition of the vector *rxa* containing the tangent heights of the spectra used for the interpolation

The vector *rxa* is set-up with the tangent altitudes of the spectra considered for the interpolation, starting from the lowest tangent altitude.

Begin loop 1 on the frequencies of the actual MW

$jsig=1 \rightarrow nsam(imw)$,

nsam(imw) is total number of sampling points in MW *imw*.

2. Definition of the vector *rya* containing the values of the spectra corresponding to the tangent altitudes contained in *rxa* for the actual frequency.
The vector *rya* is filled with the values of the three considered spectra at the frequency *jsig*:
rya(1)=rspct(jsig,jgeodown), rya(2)=rspct(jsig,jgeo), rya(3)=rspct(jsig,jgeoup).

3. Interpolation of the spectrum, analytical convolution and normalisation
All these operations are performed by calling module **intcon** (*rxa, rya, 3, rbase, rsl, rarea, rzc, rp*)

4. Storage of the calculated value in the vector *rspfov* .
rspfov(jsig, iactugeo,imw)=rp.

Begin loop 2 on the continuum parameters that affect the spectrum corresponding to geometry jgeo
jpar = igeocder(jgeo,1) → igeocder(jgeo,2)
The inputs of this section are, for each observed geometry, all the derivatives of the corresponding spectrum with respect to the continuum parameters that affect it:
rspctcder(jsig,jgeo,jpar);
igeocder(jgeo,1) and igeocder(jgeo,2) represent respectively the highest and lowest parameter level, whose value of continuum affects the spectrum corresponding to the geometry jgeo.
All these quantities have to be convolved with the FOV function, so the operations from 2. to 4. are repeated for the matrix rspctcder(jsig,jgeo,jpar).

5. Definition of the vector *rya* containing the values of the derivatives with respect to continuum corresponding to *rxa* for the actual frequency.
The vector *rya* is set-up with the values of the three considered continuum derivatives at the frequency *jsig*:
rya(1)=rspctcder(jsig,jgeodown,jpar),
rya(2)=rspctcder(jsig,jgeo,jpar),
rya(3)=rspctcder(jsig,jgeoup,jpar).

6. Interpolation of the continuum derivatives, analytical convolution and normalisation
See 3.
7. Storage of the calculated value in the vector *rcderfov*.
rcderfov(jsig, iactugeo,imw)=rp.

Begin loop 3 on the VMR parameters that affect the spectrum corresponding to geometry jgeo
jpar=igeogder(jgeo,1) → igeogder(jgeo,2)
The inputs of this section are, for each observed geometry, all the derivatives of the corresponding spectrum with respect to the VMR parameters that affect it:
rspctgder(jsig,jgeo,jpar);
igeogder(jgeo,1) and igeogder(jgeo,2) represent respectively the highest and lowest parameter level, whose value of continuum affects the spectrum corresponding to the geometry jgeo.
All these quantities have to be convolved with the FOV function, so the operations 2. and 3. are repeated for the matrix rspctgder(jsig,jgeo,jpar).

8. Definition of the vector *rya* containing the values of the derivatives of the spectra with respect to VMR corresponding to *rxa* for the actual frequency.
The vector *rya* is set-up with the values of the three considered perturbed spectra at the frequency *jsig*:
rya(1)=rspctgder(jsig,jgeodown,jpar),
rya(2)=rspctgder(jsig,jgeo,jpar),
rya(3)=rspctgder(jsig,jgeoup,jpar).

9. Interpolation of the continuum derivatives, analytical convolution and normalisation
See 3.

10. Storage of the calculated value in the vector *rgderfov*.
rgderfov(jsig, iactugeo,imw)=rp.

3.2.11.12 FOV4_VMR

Description
This module performs the same operations as module **fov3_vmr**, but the interpolation is performed using 4 spectra instead of 3.
After the interpolation in altitude between the spectra at four contiguous tangent altitudes, this module performs the analytical convolution of the interpolated spectrum with the FOV function. This procedure is repeated for the derivatives of the spectrum with respect to continuum and the derivatives of the spectra with respect to VMR.

Variables exchanged with external modules

Name	Description
iactugeo	local counter of the geometries of the actual MW corresponding to the observations
imw	number of the actual microwindow
jgeo	actual index of simulated spectrum
rztan	tangent altitude of the spectrum of which we are calculating convolution with FOV.
jgeo1	index of the lowest considered geometry
rztan1	tangent altitude corresponding to geometry *jgeo1*
jgeo2	index of the geometry above *jgeo1*
rztan2	tangent altitude corresponding to geometry *jgeo2*
jgeo3	index of the geometry above geometry *jgeo2*
rztan3	tangent altitude corresponding to geometry *jgeo3*
rspct	rspct(imxi,imxgeo): low-resolution spectrum (rsp * ILS)
rspctcder	rspctcder(imxi,imxgeo,imxlmb): the convolved continuum derivative spectra for each geometry and each parameter level
rspctgder	rspctgder(imxi,imxgeo,imxlmb): low resolution spectra for the perturbed temperature profiles
nsam	nsam(imxmw): no. of sampling points in each MW (coarse grid)
rbase	greater base of trapezium-shape that approximates Field of View pattern
rsl	half-difference between the bases of the trapezium (1/rsl gives the slope)

rspfov	rspfov(imxi,imxgeo,imxmw): simulated spectra corresponding to the different tangent pressures and different microwindows on the frequency coarse grid: (rspct * FOV)
rcderfov	rcderfov(imxi,imxgeo,imxlmb) derivate with respect to continuum after fov convolution
rgderfov	rgderfov(imxi,imxgeo,imxlmb) temperature-perturbed spectra after fov-convolution
igeocder	igeocder(imxgeo,2): for each geometry the highest (x,1) and lowest (x,2) continuum derivative (in the parameter-grid) which has to be calculated
igeogder	igeogder(imxgeo,2): for each simulated geometry the highest (x,1) and lowest (x,2) parameter with respect to which the derivatives of the spectra (VMR) are calculated
rzc	altitude in correspondence of which the convolution with the FOV is calculated
rarea	area of the trapezium that approximates antenna pattern of FOV

Module structure:

1. Definition of the vector *rxa* containing the tangent heights of the spectra used for the interpolation

Begin loop 1 on the frequencies of the actual MW

2. Definition of the vector *rya* containing the values of the spectra corresponding to *rxa* for the actual frequency.

3. Interpolation of the spectrum, analytical convolution and normalisation

4. Storage of the calculated value in the vector *rspfov* .

Begin loop 2 on the continuum parameters that affect the spectrum corresponding to geometry jgeo

5. Definition of the vector *rya* containing the values of the derivatives with respect to continuum corresponding to *rxa* for the actual frequency.

6. Interpolation of the continuum derivatives, analytical convolution and normalisation

7. Storage of the calculated value in the vector *rcderfov*.

End loop 2

Begin loop 3 on the VMR parameters that affect the spectrum corresponding to geometry jgeo

8. Definition of the vector *rya* containing the values of the derivatives with respect to VMR corresponding to *rxa* for the actual frequency.

9. Interpolation of the continuum derivatives, analytical convolution and normalisation

10. Storage of the calculated value in the vector *rgderfov*.

End loop 3

End loop 1

Detailed description

1. Definition of the vector *rxa* containing the tangent heights of the spectra used for the interpolation

The vector *rxa* is set-up with the tangent altitudes of the 4 spectra considered for the interpolation. Note that in this case the order of tangent heights is not a-priori defined, so we assign the tangent altitude of the central frequency to rxa(4), while the other rxa(i), i=1-3 are set-up with the other tangent altitudes, from the lowest to the highest one.

Begin loop 1 on the frequencies of the actual MW
jsig=1→ nsam(imw) ,
nsam(imw) is total number of sampling points in MW *imw*.

2. Definition of the vector *rya* containing the values of the spectra corresponding to the tangent altitudes contained in *rxa* for the actual frequency.
The vector *rya* is filled with the values of the four considered spectra at the frequency *jsig*:
rya(1)=rspct(jsig,jgeo1), rya(2)=rspct(jsig,jgeo2), rya(3)=rspct(jsig,jgeoup3), rya(4)=rspct(jsig,jgeo), .

3. Interpolation of the spectrum, analytical convolution and normalisation
All these operations are performed by calling module **intcon** (*rxa, rya, 4, rbase, rsl, rarea, rzc, rp*)

4. Storage of the calculated value in the vector *rspfov* .
rspfov(jsig, iactugeo,imw)=rp.

Begin loop 2 on the continuum parameters that affect the spectrum corresponding to geometry jgeo
jpar = igeocder(jgeo,1) → igeocder(jgeo,2)
The inputs of this section are, for each observed geometry, all the derivatives of the corresponding spectrum with respect to the continuum parameters that affect it:
rspctcder(jsig,jgeo,jpar);
igeocder(jgeo,1) and igeocder(jgeo,2) represent respectively the highest and lowest parameter level, whose value of continuum affects the spectrum corresponding to the geometry jgeo.
All these quantities have to be convolved with the FOV function, so the operations from 2. to 4. are repeated for the matrix rspctcder(jsig,jgeo,jpar).

5. Definition of the vector *rya* containing the values of the derivatives with respect to continuum corresponding to *rxa* for the actual frequency.
The vector *rya* is set-up with the values of the three considered continuum derivatives at the frequency *jsig*:
rya(1)=rspctcder(jsig,jgeo1,jpar),
rya(2)=rspctcder(jsig,jgeo2,jpar),
rya(3)=rspctcder(jsig,jgeo3,jpar)
rya(4)=rspctcder(jsig,jgeo,jpar).

6. Interpolation of the continuum derivatives, analytical convolution and normalisation
See 3.

7. Storage of the calculated value in the vector *rcderfov*.
rcderfov(jsig, iactugeo,imw)=rp.

Begin loop 3 on the VMR parameters that affect the spectrum corresponding to geometry jgeo
jpar=igeogder(jgeo,1) → igeogder(jgeo,2)

The inputs of this section are, for each observed geometry, all the derivatives of the corresponding spectrum with respect to the VMR parameters that affect it:
rspctgder(jsig,jgeo,jpar);
igeogder(jgeo,1) and igeogder(jgeo,2) represent respectively the highest and lowest parameter level, whose value of continuum affects the spectrum corresponding to the geometry jgeo.
All these quantities have to be convolved with the FOV function, so the operations 2. and 3. are repeated for the matrix rspctgder(jsig,jgeo,jpar).

8. Definition of the vector *rya* containing the values of the derivatives of the spectra with respect to VMR corresponding to *rxa* for the actual frequency.
The vector *rya* is set-up with the values of the three considered perturbed spectra at the frequency *jsig*:
rya(1)=rspctgder(jsig,jgeo1,jpar),
rya(2)=rspctgder(jsig,jgeo2,jpar),
rya(3)=rspctgder(jsig,jgeo3,jpar)
rya(4)=rspctgder(jsig,jgeo,jpar).

9. Interpolation of the continuum derivatives, analytical convolution and normalisation
See 3.

10. Storage of the calculated value in the vector *rgderfov*.
rgderfov(jsig, iactugeo,imw)=rp.

3.2.11.13 FOV5_VMR

Description
This module performs the same operations as module FOV3_VMR and FOV4_VMR, but the interpolation of the spectrum is performed using 5 spectra.
After the interpolation in altitude between the spectra at five contiguous tangent altitudes, this module performs the analytical convolution of the interpolated spectrum with the FOV function. This procedure is repeated for the derivatives of the spectrum with respect to continuum and the derivatives of the spectra with respect to VMR.

Variables exchanged with external modules

Name:	Description:
iactugeo	local counter of the geometries of the actual MW corresponding to the observations
imw	number of the actual microwindow
jgeo	actual index of simulated spectrum
rztan	tangent altitude of the spectrum of which we are calculating convolution with FOV.
jgeoup1	index of the geometry just above *jgeo*
rztanup1	tangent altitude corresponding to geometry *jgeoup1*
jgeodown1	index of the geometry just above *jgeo*

rztandown1	tangent altitude corresponding to geometry *jgeodown1*
jgeoup2	index of the geometry above *jgeoup1*
rztanup2	tangent altitude corresponding to geometry *jgeoup2*
jgeodown2	index of the geometry below geometry *jgeoup1*
rztandown2	tangent altitude corresponding to geometry *jgeodown2*
rspct	rspct(imxi,imxgeo): low-resolution spectrum (rsp * ILS)
rspctcder	rspctcder(imxi,imxgeo,imxlmb): the convolved continuum derivative spectra for each geometry and each parameter level
rspctgder	rspctgder(imxi,imxgeo,imxlmb): low resolution spectra for the perturbed temperature profiles
nsam	nsam(imxmw): no. of sampling points in each MW (coarse grid)
rbase	greater base of trapezium-shape that approximates Field of View pattern
rsl	half-difference between the bases of the trapezium (1/rsl gives the slope)
rspfov	rspfov(imxi,imxgeo,imxmw): simulated spectra corresponding to the different tangent pressures and different microwindows on the frequency coarse grid: (rspct * FOV)
rcderfov	rcderfov(imxi,imxgeo,imxlmb) derivate with respect to continuum after fov convolution
rgderfov	rgderfov(imxi,imxgeo,imxlmb) temperature-perturbed spectra after fov-convolution
igeocder	igeocder(imxgeo,2): for each geometry the highest (x,1) and lowest (x,2) continuum derivative (in the parameter-grid) which has to be calculated
igeogder	igeogder(imxgeo,2): for each simulated geometry the highest (x,1) and lowest (x,2) parameter with respect to which the derivatives of the spectra (VMR) are calculated
rzc	altitude in correspondence of which the convolution with the FOV is calculated
rarea	area of the trapezium that approximates antenna pattern of FOV

Module structure:

1. Definition of the vector *rxa* containing the tangent heights of the spectra used for the interpolation

Begin loop 1 on the frequencies of the actual MW

2. Definition of the vector *rya* containing the values of the spectra corresponding to *rxa* for the actual frequency.

3. Interpolation of the spectrum, analytical convolution and normalisation

4. Storage of the calculated value in the vector *rspfov* .

Begin loop 2 on the continuum parameters that affect the spectrum corresponding to geometry jgeo

5. Definition of the vector *rya* containing the values of the derivatives with respect to continuum corresponding to *rxa* for the actual frequency.

6. Interpolation of the continuum derivatives, analytical convolution and normalisation

7. Storage of the calculated value in the vector *rcderfov*.

End loop 2

Begin loop 3 on the VMR parameters that affect the spectrum corresponding to geometry jgeo

8. Definition of the vector *rya* containing the values of the derivatives with respect to VMR corresponding to *rxa* for the actual frequency.

9. Interpolation of the continuum derivatives, analytical convolution and normalisation

10. Storage of the calculated value in the vector *rgderfov*.

End loop 3

End loop 1

Detailed description:

1. Definition of the vector *rxa* containing the tangent heights of the spectra used for the interpolation

The vector *rxa* is set-up with the tangent altitudes of the 5 spectra considered for the interpolation, starting from the lowest geometry *jgeodown2*.

Begin loop 1 on the frequencies of the actual MW

jsig=1→ nsam(imw) ,

nsam(imw) is total number of sampling points in MW *imw*.

2. Definition of the vector *rya* containing the values of the spectra corresponding to the tangent altitudes contained in *rxa* for the actual frequency.

The vector *rya* is filled with the values of the five considered spectra at the frequency *jsig*:

rya(1)=rspct(jsig,jgeodown2), rya(2)=rspct(jsig,jgeodown1), rya(3)=rspct(jsig,jgeo), rya(4)=rspct(jsig,jgeoup1), rya(5)=rspct(jsig,jgeoup2) .

3. Interpolation of the spectrum, analytical convolution and normalisation

All these operations are performed by calling module **intcon** (*rxa, rya, 5, rbase, rsl, rarea, rzc, rp*)

4. Storage of the calculated value in the vector *rspfov* .

rspfov(jsig, iactugeo,imw)=rp.

Begin loop 2 on the continuum parameters that affect the spectrum corresponding to geometry jgeo

jpar = igeocder(jgeo,1) → igeocder(jgeo,2)

The inputs of this section are, for each observed geometry, all the derivatives of the corresponding spectrum with respect to the continuum parameters that affect it:

rspctcder(jsig,jgeo,jpar);

igeocder(jgeo,1) and igeocder(jgeo,2) represent respectively the highest and lowest parameter level, whose value of continuum affects the spectrum corresponding to the geometry jgeo.

All these quantities have to be convolved with the FOV function, so the operations from 2. to 4. are repeated for the matrix rspctcder(jsig,jgeo,jpar).

5. Definition of the vector *rya* containing the values of the derivatives with respect to continuum corresponding to *rxa* for the actual frequency.

The vector *rya* is set-up with the values of the three considered continuum derivatives at the frequency *jsig*:

rya(1)=rspctcder(jsig,jgeodown2,jpar),
rya(2)=rspctcder(jsig,jgeodown1,jpar),
rya(3)=rspctcder(jsig,jgeo,jpar)
rya(4)=rspctcder(jsig,jgeoup1,jpar),
rya(5)=rspctcder(jsig,jgeoup2,jpar).

6. Interpolation of the continuum derivatives, analytical convolution and normalisation
See 3.

7. Storage of the calculated value in the vector *rcderfov.*
rcderfov(jsig, iactugeo,imw)=rp.

Begin loop 3 on the VMR parameters that affect the spectrum corresponding to geometry jgeo
jpar=igeogder(jgeo,1) → igeogder(jgeo,2)
The inputs of this section are, for each observed geometry, all the derivatives of the corresponding spectrum with respect to the VMR parameters that affect it:
rspctgder(jsig,jgeo,jpar);
igeogder(jgeo,1) and igeogder(jgeo,2) represent respectively the highest and lowest parameter level, whose value of continuum affects the spectrum corresponding to the geometry jgeo.
All these quantities have to be convolved with the FOV function, so the operations 2. and 3. are repeated for the matrix rspctgder(jsig,jgeo,jpar).

8. Definition of the vector *rya* containing the values of the derivatives of the spectra with respect to VMR corresponding to *rxa* for the actual frequency.
The vector *rya* is set-up with the values of the three considered perturbed spectra at the frequency *jsig*:
rya(1)=rspctgder(jsig,jgeodown2,jpar),
rya(2)=rspctgder(jsig,jgeodown1,jpar),
rya(3)=rspctgder(jsig,jgeo,jpar)
rya(4)=rspctgder(jsig,jgeoup1,jpar),
rya(5)=rspctgder(jsig,jgeoup2,jpar).

9. Interpolation of the continuum derivatives, analytical convolution and normalisation
See 3.

10. Storage of the calculated value in the vector *rgderfov.*
rgderfov(jsig, iactugeo,imw)=rp.

3.2.11.14 POLCOE_VMR

Description: This module, taken from 'Numerical Recipes in FORTRAN', calculates the coefficients of the interpolating polynomium that passes through a given number of points.

Variables exchanged with external modules

Name:	Description:
rx	rx(n) vector containing the x values of the tabulated points

ry	ry(n) vector containing the y values of the tabulated points
n	number of tabulated points
rcof	rcof(n) returned coefficients, such that: $y_i = \sum_j rcof(j) \cdot x_i^{j-1}$

Detailed description:
see 'Numerical Recipes in FORTRAN' [RD2], pag.114.

3.2.11.15 INTCON_VMR

INTCON_VMR
|-----POLCOE_VMR +

Description: This module calculates the coefficients for the interpolation between the points contained in the vectors *rxa* and *rya* and then computes the result of the convolution for this particular value of frequency.

Variables exchanged with external modules

Name:	Description:
rxa	rx(5) vector containing the x values of the tabulated points
rya	ry(5) vector containing the y values of the tabulated points
n	number of tabulated points
rbase	greater base of the trapezium
rsl	half-difference between the two bases of the trapezium
rarea	area of the trapezium, used for normalisation
rzc	altitude in correspondence of which spectrum with FOV is calculated
rp	returned value of the convolution and normalisation

Module structure

Detailed description:
1. Calculation of the coefficients for the interpolation
Begin condition 1: n=3
2. Calculation of analytical convolution and normalisation
Else, if n=4
3. Calculation of analytical convolution and normalisation
Else, if n=5
4. Calculation of analytical convolution and normalisation
End condition 1

1. Calculation of the coefficients for the interpolation

The coefficients of the interpolation, contained in the vector *rcof*, are calculated by the module **polcoe(***rxa,rya,n,rcof***)**, so that the interpolated spectrum at altitude *r* and frequency *jsig* results given by $\sum_{i=1,n} rcof(i) \cdot r^{n-1}$.

2. Analytical convolution and normalisation

The result of the analytical convolution is given by the following expression:

$$rp = \frac{\left[\begin{array}{l} rcof(1) \cdot rarea + rcof(2) \cdot rarea \cdot rzc + rcof(3) \cdot rzc^2 \cdot rarea + \\ +rcof(3) \cdot \left(\frac{rbase^3}{12} - \frac{rbase^2 \cdot rsl}{4} + \frac{rsl^2 \cdot rbase}{3} - \frac{rsl^3}{6} \right) \end{array} \right]}{rarea}$$

4. Calculation of analytical convolution and normalisation

The result of the analytical convolution is given by the following expression:

$$rp = \frac{\left[\begin{array}{l} rcof(1) \cdot rarea + rcof(3) \cdot rzc^2 \cdot rarea + rcof(4) \cdot rarea \cdot rzc^3 + \\ rzc \cdot \left(rcof(2) \cdot rarea + rcof(4) \cdot \left(\frac{rbase^3}{4} - \frac{3 \cdot rsl \cdot rbase^2}{4} + rbase \cdot rsl^2 - \frac{rsl^3}{2} \right) \right) + \\ +rcof(3) \cdot \left(\frac{rbase^3}{12} - \frac{rbase^2 \cdot rsl}{4} + \frac{rsl^2 \cdot rbase}{3} - \frac{rsl^3}{6} \right) \end{array} \right]}{rarea}$$

5. Calculation of analytical convolution and normalisation

The result of the analytical convolution is given by the following expression:

$$rp = \frac{\left[\begin{array}{l} rcof(5) \cdot \left(\frac{rbase^5}{80} - \frac{rbase^4 \cdot rsl}{16} + \frac{rbase^3 \cdot rsl^2}{6} - \frac{rbase^2 \cdot rsl^3}{4} + \frac{rbase \cdot rsl^4}{5} - \frac{rsl^5}{15} \right) + \\ +rcof(3) \cdot \left(\frac{rbase^3}{12} - \frac{rbase^2 \cdot rsl}{4} + \frac{rsl^2 \cdot rbase}{3} - \frac{rsl^3}{6} \right) + \\ rzc^2 \cdot \left(rcof(3) \cdot rarea + rcof(5) \cdot \left(2 \cdot rbase \cdot rsl^2 - \frac{3 \cdot rbase^2 \cdot rsl}{2} - rsl^3 + \frac{rbase^3}{2} \right) \right) + \\ rarea \cdot rcof(4) \cdot rzc^3 + rcof(5) \cdot rzc^4 \cdot rarea + rcof(1) \cdot rarea + \\ rzc \cdot \left(rcof(2) \cdot rarea + rcof(4) \cdot \left(\frac{rbase^3}{4} - \frac{3 \cdot rsl \cdot rbase^2}{4} + rbase \cdot rsl^2 - \frac{rsl^3}{2} \right) \right) \end{array} \right]}{rarea}$$

3.2.11.16 CROSS_VMR

```
CROSS_VMR
   |-----BLIND_VMR *
   |-----LOFICO_VMR ]
   |-----FLINT_VMR *
   |((((+SHAPECALC_VMR
   |        |(----HUMLI_VMR ]
   |((((+BLIND_VMR *
   |((((+HUMLI_VMR ]
   |((((+FCO2CHI_VMR *
   |((((+POLCOE2ND_VMR +
```

Description:

- Using the spectroscopic line-data this routine determines the absorption cross-sections for each general wavenumber fine grid point, each IAPT number and each gas which has to be considered in the actual microwindow using the equivalent pressures and temperatures calculated in '**curgod_vmr**'.
- In the case cross-section look-up tables are available, this module returns the value of cross-sections obtained decompressing the compressed look-up tables. ORM is able to handle also cases in which the look-up tables are available only for a sub-set of the operational microwindows and for a sub-set of the gases contributing to the emission in each microwindow.
- In the case irregular grid is available, this module returns the values of the cross-sections on the so-called 'compressed' grid, which is the grid containing only the '1' grid points of the irregular grid.

Variables exchanged with external modules

Name:	Description:
imw	number of the actual Mw
rpeq	equivalent pressures
rteq	equivalent temperatures
itglev	number of the tangent-level for each geometry
isigma	number of wavenumber grid points for each Mw
dsigma	general wavenumber fine grid
delta	general fine grid interval [cm-1]
igeo	number of simulated geometries
iocsim	occupation matrix for the simulations to be performed
igasmw	number of gases to be considered in each Mw
ruplin	upper limit where the line has to be considered [km]
rlolin	lower limit where the line has to be considered [km]
iline	number of lines in each microwindow
icode	HITRAN code for each line of each Mw
rint0	line intensity for each line of each Mw
relow	lower state energy for each line of each Mw

rhw0	foreign broadened half width for each line of each Mw
dsilin	central wavenumber for each line of each Mw
ioutin	flag for each line of each Mw =1: line-shape has to be calculated at each wavenumber inside the Mw =2: line is considered as nearby continuum (calculated at three points inside the Mw
igasnr	global gas number for the local gas number of each Mw
rexph	exponent for T dependence of half width for each line of each Mw
rwmol	molecular weight for each HITRAN molecular code and isotope number
igashi	HITRAN code number for each global gas number
iiso	isotope number for each line of each Mw
ipoint	IAPT-number for each layer and each geometry
ninterpol	switch for the decision of interpolation of the absorption cross-sections for the geometries above the lowest geometry (only if the IAPT number of the path is increasing, which was decided during the calculation of ipoint) =-1: no interpolation, all cross-sections recalculated =0: all cross-sections above the lowest geometry are interpolated =1: new calculation only of the tangent-layer, all other layers interpolated =2: new calculation of the tangent-layer and the layer above, all others interpolated =3: ...
raircol	air-column for each layer and each geometry
rcol	column amounts for each layer, each geometry and each gas
rzmod	heights of levels used for the radiat. tranf. calculation
lmgas	logical array lmgas(imxgmw,imxmw) lmgas(mgas,imw)=.true. : calculation of cross-sections without look-up tables lmgas(mgas,imw)=.false. :calculation of cross-sections by means of look-up tables
ilookupmw	integer*4 ilookupmw(imxmw) ilookupmw(imw)=0 no look-up tables for mw imw ilookupmw(imw)=1 look-up tables for all the absorbers of the mw ilookupmw(imw)=2 look-up tables for not all the absorbers of the mw
nll	I*4 : nll(imxgmw,imxmw): number of basis vector
npl	I*4 : npl(imxgmw,imxmw): number of -log(pressure) tabulation points
rp1l	R*4: rp1l(imxgmw,imxmw): lowest -log(pressure) value
rdpl	R*4: rdpl(imxgmw,imxmw): spacing of -log(pressure) tabulation
ntl	I*4: ntl(imxgmw,imxmw): number of temperature tabulation points
rt1l	R*4: rt1l(imxgmw,imxmw): lowest tabulated temperature
rdtl	R*4: rdtl(imxgmw,imxmw): spacing of temperature tabulation
ru	R*4: ru(imxsi2,imxbv,imxgmw,imxmw): U-matrix
rkl	R*4: rkl(imxbv,imxnx,imxgmw,imxmw): K-matrix
tab	character*3: tab(imxgmw,imxmw): tabulation code of cross-section look-up tables
rcross	R*4: rcross(imxsi2,imxpat,imxgmw): absorption cross sections for each general wavenumber fine grid point (1st index), each IAPT number (2nd index) and each gas (3rd index) for the actual Mw
lirrgridm	logical: *lirrgridmw(imxmw):* logical vector that, for each selected microwindow

w	in the actual retrieval, indicates whether the irregular grid is available.
iigrid	integer*4: *iigrid(imxsig,imxgeo,imxmw).* *iigrid (1→ isigma(imw),imw):* irregular grid in the '0' and '1' representation for all the fine grid points of the extended microwindow *imw.*
nused1	integer*4: nused1(imxmw): total number of points of the compressed grid for each microwindow absorption cross-sections for the main gas: for each general wavenumber fine grid point (1st index), for each IAPT number (2nd index), and for the two equivalent temperatures profiles (3rd index). So, rcrosspert(i,j,1) are the cross sections calculated using the temperatures rteqpert(j,1) and rcrosspert(i,j,2) by using rteqpert(j,2).

Module structure:

1. Initialisation of variables
Begin loop 1 over the geometries valid for the actual microwindow
Begin loop 2 over the layers of the actual geometry for which a new cross- section must be determined
Begin condition 1 on the use of look-up tables
2. Calculation of cross-sections by means of available look-up tables
End condition 1
Begin condition 2 : the cross-sections are calculated without look-up tables
Begin condition 3 the cross sections are interpolated
Begin loop 3 over the gases of the actual Mw
3. Interpolation of the cross sections
end loop 3
else condition 3 the cross sections are calculated
4. Definition of local fine and coarse wavenumber grid
Begin loop 4 over all lines of the actual Mw that must be considered for the actual altitude
5. Initialisation of variables for line-calculation
Begin condition 4 the lines are calculated at each point
Begin condition 5 a line shape for HNO_3 is precalculated
6. Precalculation of HNO_3 line shape
end condition 5
7. Calculation of the line in the local coarse grid
8. Calculation of the line in the local fine grid
else condition 4 the lines are handled as near continuum
9. Calculation of the line at 3 points inside the Mw
end condition 4
end loop 4
Begin loop 7 over the gases of the actual Mw
10. Interpolation of the cross sections from the local coarse and fine grid to the general fine grid
11. Interpolation of the nearby continuum to the general fine grid
end loop 5
end condition 3

end condition 2
end loop 2
end loop 1

Detailed description:

loop 1 over the geometries valid for the actual microwindow
kgeo=igeo→1
if [*iocsim(kgeo,imw)*≠0]
Starting from the lowest geometry (*igeo*) this loop (i.e. the commands inside the loop) is only executed if this observation geometry has to be simulated for the actual Mw.

loop 2 over the layers of the actual geometry for which a new cross-section must be determined
llay=1→itglev(kgeo) -1
This loop begins from the outer layer and goes down to the tangent layer (*itglev(kgeo)-1*). It is only executed if new cross-sections must be calculated, i.e. if the IAPT-number *ipoint(llay,kgeo)* is increasing. For the cases that the IAPT number is not increasing, the cross-sections have already been calculated during an earlier execution.

Condition 1 on the use of look-up tables
if ilookupmw(imw) ≠ 0 , at least for some of the absorbers contained in the mw look-up tables are available.

condition 2: the cross-sections are calculated without look-up tables
if *ilookupmw(imw) =0 .OR.. ilookupmw(imw) = 2 ,*
cross-section calculation is performed.

condition 3 the cross sections are interpolated or calculated
The cross sections are interpolated (using the cross sections which have already been calculated for the lowest geometry) if we are not in the lowest geometry and if we are in a layer that has to be interpolated (indicated by *ninterpol*):

if [*kgeo<ilowgeo* ∧ *llay < itglev(kgeo)-ninterpol* ∧ *ninterpol*≠-1]

Where *ilowgeo* is the lowest geometry that must be calculated for the actual Mw.
If this conditions are not fulfilled the cross sections are calculated explicitly using the line data.

loop 3 over the gases of the actual Mw
mgas=1→igasmw(imw)
The calculations inside this loop are performed only if one of the following conditions are verified:
ilookupmw(imw) = 0 .or. lmgas(mgas,imw) = .true.

loop 4 over all lines of the actual Mw that must be considered for the actual altitude and corresponding to gases of which cross-section was not previously calculated with the use of look-up tables
mline=1,iline(imw)
if [*ruplin(mline,imw)>rzmod(llay)>rlolin(mline,imw)*]

if [*ilookupmw(imw) = 0 .or. lmgas(igasact (icode (mline, imw), imw) = .true.*]

condition 4 the lines are calculated at each point or handled as continuum
if [*ioutin(mline,imw)=1*]: the lines are explicitly calculated at each point of the local coarse and fine grid.
if [*ioutin(mline,imw)=2*]: the lines are handled as near continuum and calculated only at three points inside the Mw.

condition 5 a line shape for HNO_3 is precalculated
if the gas is HNO_3 (if [*icode(mline,imw)=nrepcode*]) and the half width is equal to the reference half width (if [*rhw0(mline,imw)=rephw0*]) and the half width exponent is equal to the reference exponent (if [*rexph(mline,imw)=repexph*]) and if the line shape has not already been precalculated (if [*nshape=0*]) then a line shape is precalculated.

loop 5 over the gases of the actual Mw
mgas=1→igasmw(imw)
The calculation inside this loop are performed only if one of the following conditions are verified:
ilookupmw(imw) = 0 .or. lmgas(mgas,imw) = .true.

1. Initialisation of variables
- Calculation of vector *igasact(imxhit)* that gives for each hitran gas number the local Mw gas number: *igasact(igashi(igasnr(j,imw))) = j* for $1 \le j \le igasmw(imw)$
- Determination of the line with the largest intensity of the main gas: line number: *imaxlin*
- The total number of points *nsigma* of the grid to be used for the calculation of cross-section is determined. If an irregular grid is available, the compressed grid is used and *nsigma= nused1(imw),* if the irregular grid is not available, *nsigma= isigma(imw).*

2. Calculation of cross-sections by means of available look-up tables
For each absorbers contained in the considered mw: *mgas = 1 → igasmw(imw),* a control is done in order to see if the look-up table relative to this absorber is available (*lmgas(mgas,imw)=* .false.).
If this is the case, cross-section calculation is performed as follows:
firstly, the preliminary calculations are performed:
the hitran code of the gas:
ihit=igashi(igasnr(mgas,imw));
and the equivalent -log(pressure) and temperature relative to the path ipo:
rp= -alog(rpeq(ipoint(llay,kgeo),igasnr(mgas,imw));
rt= rteq(ipoint(llay,kgeo),igasnr(mgas,imw);

The calculation of cross-section is performed by module **decompr_vmr:**
decompr_vmr(*rp, rt, mgas, imw, ru, rkl, nll, npl, rp1l, rdpl, ntl, rt1l, rdtl, nsigma, tab, rcross1***)**
The vector rcross1 is then stored in the array rcross:
for each *msig,* from 1 to *nsigma:*
rcross(msig,ipo,mgas)=rcross1(msig)

3. Interpolation of the cross sections
The cross sections for the geometries above the lowest geometry are calculated (for each general fine grid point) by linear interpolation using the cross sections already calculated for the lowest

geometry. This linear interpolation is performed with respect to the equivalent pressures, i.e. it is first decided between which equivalent pressures of the lowest geometry the actual equivalent pressure lies and than the cross sections are interpolated to the actual equivalent pressure. This is done for the cross sections of all gases (*rcross*).
E.g. for *rcross* the formula for all wavenumbers on the general fine wavenumber grid (*msig*) is:

$$rcross(msig, ipoint(llay, kgeo), mgas) = r1 + (r2 - r1) \cdot \frac{p - p1}{p2 - p1}$$

$$r1 = rcross(msig, ipoint(llay1, ilowgeo), mgas)$$

$$r2 = rcross(msig, ipoint(llay2, ilowgeo), mgas)$$

with: $p = rpeq(ipoint(llay, kgeo), igasnr(mgas, imw))$

$$p1 = rpeq(ipoint(llay1, ilowgeo), igasnr(mgas, imw))$$

$$p2 = rpeq(ipoint(llay2, ilowgeo), igasnr(mgas, imw))$$

Where *llay1* and *llay2* determine the pressures *p1* and *p2* of the lowest geometry between which the actual pressure *p* lies.

4. Definition of local fine and coarse wavenumber grid

The local (for the actual geometry and layer) coarse and fine wavenumber grid is defined by calling the module **lofico_vmr**:

$$\text{lofico}\begin{bmatrix} dsilin, ipoint(llay, kgeo), imw, rwmol, icode, iiso, rhw0, imaxli, rpeq, rteq, \\ delta, isigma, dsigma, igasmw, rexph, iqlfgf, dsiglf, dsiglc, isiglf, isiglc, \\ deltalf, deltalc, rcrolf, rcrolc, rcrolfpert, rcrolcpert \end{bmatrix}$$

Note that even if an irregular grid is available, and as a consequence the final cross-sections will be stored on the compressed grid, the local fine and coarse grids are built starting not from the compressed grid, but from the regular fine grid.
Therefore, lofico routine is not affected by the use of the compressed grid.

5. Initialisation of variables for line-calculation

- Calculation of the Doppler half width:

$$rdhalf = dsilin(mline, imw) \cdot dcdop \cdot \sqrt{\frac{rteq(ipo, ign)}{rwmol(icode(mline, imw), iiso(mline, imw))}}$$

with: $ipo = ipoint(llay, kgeo)$

and: $ign = igasnr(igasact(icode(mline, imw)), imw)$, the global gas number for the hitran gas number of the actual line.
dcdop is a parameter.

- Calculation of the Lorentz half width:

$$rlhalf = rhw0(mline,imw)\cdot\frac{rpeq(ipo,ign)}{rp0h}\cdot\left[\frac{rt0h}{rteq(ipo,ign)}\right]^{rexph(mline,imw)}$$

With the parameters *rp0h, rt0h.*

- Calculation of the line intensity
  The line intensity *rlint* is calculated by a call to the module **flint_vmr**:

$$rlint = \text{flint\_}vmr\begin{bmatrix} rint0(mline,imw), relow(mline,imw), rteq(ipo,ign), \\ dsilin(mline,imw), icode(mline,imw), iiso(mline,imw) \end{bmatrix}$$

6. Precalculation of HNO_3 line shape

In the case of HNO_3 the line shape is precalculated for the Voigt part of the line. This precalculation is performed in the general fine grid by a call to the subroutine

$$shapecalc\begin{bmatrix} ipo, ign, rteq(ipo,ign), dsigma(1,imw), isigma(imw), delta, \\ rwmol(icode(mline,imw), iiso(mline,imw)), iprec, rshape \end{bmatrix}$$

Then the variable *nshape* is set to 1 in order to indicate, that for this IAPT number *ipo* the shape is already precalculated. When going to the next IAPT *nshape* has to be initialised again to 0 (before begin of next loop 4 over all lines)!

7. Calculation of the line in the local coarse grid

The cross sections on the local coarse grid *rcrolc (*dimension *(imxsig,imxgmw)*) are calculated from the boundaries of the microwindow up to a distance of $(rdhalf + rlhalf)\cdot rvmult$ wavenumbers from the line centre by using the Lorentz function (*rvmult* is a parameter). In the region around the line centre the cross sections on the fine grid are constant. This constant is determined as the mean value of the last Lorentz calculated cross sections on the left and on the right of the line.
The boundary indices for the Lorentz calculation on the local coarse grid are:

$$i1c = 1$$

$$i2c = \text{nint}\left[\frac{dsilin(mline,imw) - (rdhalf + rlhalf)\cdot rvmult - dsiglc(1)}{deltalc}\right] + 1$$

$$i3c = \text{nint}\left[\frac{dsilin(mline,imw) + (rdhalf + rlhalf)\cdot rvmult - dsiglc(1)}{deltalc}\right] + 1$$

$$i4c = isiglc$$

Where *isiglc, dsiglc, deltalc* have been determined in 3.
(One has to take care that for a line very near to the boundary of the microwindow (where i2c could become less than *i1c* ...) these coeffiecents are set to the boundary values!)

Calculation of Lorentz function for $i1c \le i \le i2c\text{-}1$ and $i3c\text{+}1 \le i \le i4c$:

$$rlinfctlc(i) = \frac{1}{\pi} \frac{rlhalf}{rlhalf^2 + \left(dsiglc(i) - dsilin(mline, imw)\right)^2}$$

Calculation of the cross sections and adding to the cross sections from the previous lines:

$$rcrolc(i, ig) = rlint \cdot rlinfctlc(i) + rcrolc(i, ig)$$

with: $ig = igasact\left(icode\left(mline, imw\right)\right)$, the local gas number for the actual line.

The value for the 'plateau' region, i.e. in the vicinity of the line centre is:

$$rplatfctn = \frac{rlinfctlc(i2c - 1) + rlinfctlc(i3c + 1)}{2}$$

So, for $i2c \le i \le i3c$:

$$rcrolc(i, ig) = rlint \cdot rplatfctn + rcrolc(i, ig)$$

8. Calculation of the line in the local fine grid

On the local fine grid the lines are only calculated in the vicinity of the line, where the cross sections on the local coarse grid are constant (see 5.), i.e. for distances less than $\left(rdhalf + rlhalf\right) \cdot rvmult$ wavenumbers from the line centre. In this region the line profile is partly calculated by the Lorentz and partly by the Voigt function. The Voigt function is used inside an intervall of $\pm rdhalf \cdot rdmult$ wavenumbers from the line centre (*rdmult* is a parameter).
The boundary indices on the local fine grid are:

$$i1f = (i2c - 2) \cdot iqlclf + 2$$

$$i2f = \text{nint}\left[\frac{dsilin\left(mline, imw\right) - rdhalf \cdot rdmult - dsiglf\left(i1f\right)}{deltalf}\right] + i1f$$

$$i3f = \text{nint}\left[\frac{dsilin\left(mline, imw\right) + rdhalf \cdot rdmult - dsiglf\left(i1f\right)}{deltalf}\right] + i1f$$

$$i4f = i3c \cdot iqlclf$$

With the parameter *iqlclf*, the quotient between the local coarse and fine grid.

Calculation of Lorentz function for $i1f \le i \le i2f\text{-}1$ and $i3f\text{+}1 \le i \le i4f$:

$$rlinfctlf(i) = \frac{1}{\pi} \frac{rlhalf}{rlhalf^2 + \left(dsiglc(i) - dsilin(mline, imw)\right)^2}$$

Calculation of the cross sections and adding to the cross sections from all the previous lines:

$$rcrolf(i, ig) = rlint \cdot rlinfctlf(i) - rplatcro + rcrolf(i, ig)$$

with: $ig = igasact\left(icode\left(mline, imw\right)\right)$, the local gas number for the actual line, and *rplatcro* the coarse grid 'plateau' value which was determined in 5.

For $i2f \le i \le i3f$ the line function is determined by the Voigt lineshape:

$$rlinfctlf(i) = \sqrt{\frac{\ln 2}{\pi}} \frac{rre}{rdhalf}$$

where *rre* is the result from a call to the routine **humli_vmr**(*rx,ry,rre*), with:

$$rx = \sqrt{\ln 2} \frac{\left|dsiglf(i) - dsilin(mline, imw)\right|}{rdhalf}$$

$$ry = \sqrt{\ln 2} \frac{rlhalf}{rdhalf}$$

The cross sections are calculated from *rlinfctlf* like in the case of the Lorentz calculation (see above).

In the case of HNO_3 for the Voigt part the precalculated line shape is interpolated linearly. If the gas is HNO_3 (if [*icode(mline,imw)=nrepcode*]) and the half width is equal to the reference half width (if [*rhw0(mline,imw)=rephw0*]) and the half width exponent is equal to the reference exponent (if [*rexph(mline,imw)=repexph*]):

$$rlinfctlf(i) = \sqrt{\frac{\ln 2}{\pi}} \frac{r2}{rdhalf}$$

where *r2* is the linear interpolation of the precalculated line shape *rshape* (centred in the centre of the actual line) to the actual local fine grid wavenumber *dsiglf(i)*.

9. Calculation of the line at 3 points inside the Mw

For lines outside the microwindow which are taken into account as near continuum, the cross sections are calculated at the first point, at the middle point and at the last point of the microwindow. Later, in 10, they will be interpolated to the general fine grid.
The prodecure is:

- calculating the line profile using the Lorentz line shape (see above) at the three wavenumbers inside the microwindow.
- if the line is a CO_2 line (if [icode(mline,imw)=2]) the profile is multiplied with the CO_2 chi factor which is calculated by a call to module **fco2chi_vmr**:

  $$\boldsymbol{fco2chi\_vmr}\left[rteq\left(ipo, ign\right), dconsi - dsilin(mline, imw), 1\right]$$

  ipo and *ign* have been defined in 4.

- The absorption cross sections at the 3 points inside the Mw are now calculated like in 7 or 8 by multiplication of the profile with and added to the near continuum cross sections from the previous line calculation.

10. Interpolation of the cross sections from the local coarse and fine grid to the general fine grid
For each gas of the microwindow, the output cross section vector *rcross(i,ipo,mgas)* of the general fine grid is filled by linear interpolation in wavenumber using the vectors *rcrolf(j,mgas)* and *rcrolc(k,mgas)*, where *j* is the index on the local fine grid and *k* on the local coarse grid.
If an irregular grid is available (*lirrgridmw(imw)*= true) only the cross-section values corresponding to the points of the 'compressed' grid have to be stored in
$\mathrm{rcross}(\mathrm{ksig} = 1 \rightarrow nsigma, ipo, mgas)$.

11. Interpolation of the nearby continuum to the general fine grid
For each gas of the Mw the nearby continuum values which were calculated in 8. for three points inside the microwindow are interpolated (2nd order) to the general wavenumber fine grid and added to the cross section output vectors *rcross*. As in 10., if an irregular grid is available (*lirrgridmw(imw)*= true) these operations have to be done only for the points of the 'compressed' grid.
The coefficients for the parabolic interpolation are calculated using module **polcoe2nd_vmr**.

3.2.11.17 SHAPECALC_VMR

Description
Calculation of the line shape which is later used for the calculation of many lines with equal half widths just by shifting and interpolation.

Variables exchanged with external modules

Name:	Dimension:	Description:
rp		equivalent pressure
rt		equivalent temperature
dsi1		first wavenumber of the Mw
isi		number of general fine grid points in the Mw
delta		general fine grid distance [cm^{-1}]
rw		molecular weight
iprec		number of points of the precalculated line shape
rshape	imxsig	precalculated line shape

Module structure:
1.Calculation of rshape

Detailed description:

- the reference doppler half width is calculated using the middle wavenumber of the microwindow:

$$dsil = dsi1 + \frac{isi \cdot delta}{2}$$

$$rdhalf = dsil \cdot dcdop \cdot \sqrt{\frac{rt}{rw}}$$

- the reference Lorentz half width is calculated:

$$rlhalf = rephw0 \cdot \frac{rp}{rp0h} \cdot \left(\frac{rt0h}{rt}\right)^{repexph}$$

- the number of precalculated points is:

$$iprec = \mathrm{int}\left(\frac{rdhalf \cdot rdmult}{delta}\right) + 10$$

- calculation of *rshape*:
  for $0 \le i \le iprec$:

$$rx = \sqrt{\ln 2}\,\frac{|i \cdot delta|}{rdhalf}$$

$$ry = \sqrt{\ln 2}\,\frac{rlhalf}{rdhalf}$$

  the line shape is calculated by a call the humlicek routine:

$$humli[rx, ry, rshape(i)]$$

3.2.11.18 FCO2CHI_VMR

For the description of this module, see par. 2.2.11.12.

3.2.11.19 FLINT_VMR

For the description of this module, see par. 2.2.11.13.

3.2.11.20 FPARTS_VMR

For the description of this module, see par. 2.2.11.14.

3.2.11.21 HUMLI_VMR

For the description of this module, see par. 2.2.11.15.

3.2.11.22 POLCOE2ND_VMR

For the description of this module, see par. 2.2.11.16.

3.2.11.23 LOFICO_VMR

Description

Calculation of the local fine grid and the local coarse grid for the actual atmopheric path.

Variables exchanged with external modules:

Name:	Dimension:	Description:
dsilin	imxlin imxmw	central wavenumber for each line of each Mw
ipo		actual path number
imw		actual microwindow number
rwmol	imxhit, imxism	molecular weight for each HITRAN molecular code and isotope number
icode	imxlin, imxmw	HITRAN code for each line of each Mw
iiso	imxlin, imxmw	isotope number for each line of each Mw
rhw0	imxlin, imxmw	foreign broadened half width for each line of each Mw
imaxli		number of the line of the main gas with largest intensity
rpeq	imxpat, imxgas	equivalent pressures
rteq	imxpat, imxgas	equivalent temperatures
delta		general fine grid interval [cm-1]
isigma	imxmw	number of wavenumber grid points for each Mw
dsigma	imxsig, imxmw	general wavenumber fine grid
igasmw	imxmw	number of gases to be considered in each Mw
rexph	imxlin, imxmw	exponent for T dependence of half width for each line of each Mw
iqlfgf		ratio between local fine and general fine grid
dsiglf	imxsig	local fine grid [cm^{-1}]
dsiglc	imxsig	local coarse grid [cm^{-1}]
isiglf		number of local fine grid points
isiglc		number of local coarse grid points
deltalf		distance between local fine grid points
deltalc		distance between local coarse grid points
rcrolf	imxsig, imxgmw	cross sections on local fine grid
rcrolc	imxsig,	cross sections on local coarse grid

	imxgmw	

Module structure:

1. Determination of the local fine and coarse grid.
2. Initialisation of the cross section vector on the fine and coarse grid.

Detailed description:

1. Determination of the local fine and coarse grid.

The doppler and lorentz half width for the most intense line of the main gas is calculated.

$$rdhalf = dsilin(imaxli,imw) \cdot dcdop \cdot \sqrt{\frac{rteq(ipo,1)}{rwmol(icode(imaxli,imw),iiso(imaxli,imw))}}$$

$$rlhalf = rhw0(imaxli,imw) \cdot \frac{rpeq(ipo,1)}{rp0h} \cdot \left[\frac{rt0h}{rteq(ipo,1)}\right]^{rexph(imaxli,imw)}$$

Calculation of the (approximate) Voigt half width:

$$rvhalf = rdhalt + rlhalf$$

Determine local fine grid:
For *rvlf·rvhalf < delta:*

$$iqlfgf = 1$$
$$deltalf = delta$$

For *rvlf·rvhalf > delta:*

$$iqlfgf = int\left(\frac{rvlf \cdot rvhalf}{delta}\right)$$
$$deltalf = iqlfgf \cdot delta$$

Define the local coarse grid:

$$deltalc = iqlclf \cdot deltalf$$

Number of local fine grid points:

$$isiglf = int\left(\frac{dsigma(isigma(imw),imw) - dsigma(1,imw)}{deltalf}\right) + 2$$

Fill local fine grid vector:

$$dsiglf(1) = dsigma(1, imw)$$
$$dsiglf(i) = dsiglf(i-1) + deltalf$$

Number of local coarse grid points:

$$isiglc = int\left(\frac{dsigma(isigma(imw), imw) - dsigma(1, imw)}{deltalc}\right) + 2$$

Fill local coarse grid vector:

$$dsiglc(1) = dsigma(1, imw)$$
$$dsiglc(i) = dsiglc(i-1) + deltalc$$

2. Initialisation of the cross section vector on the fine and coarse grid.

The vectors *rcrolf* and *rcrolc* are initialized to 0.

3.2.11.24 POINT_VMR

For the description of this module, see par. 2.2.11.4.

3.2.11.25 CONLAY_VMR

For the description of this module, see par. 2.2.11.3.

3.2.11.26 SPECTRUM_VMR

SPECTRUM_VMR]
|-----CONV_VMR*

Description

- calculation of the original spectra and the derivatives with respect to VMR and continuum on the general wavenumber fine grid for all geometries of the actual microwindow
- convolution of these spectra and derivatives with the AILS function to the general wavenumber coarse grid

if an irregular grid is available, the calculation of the high resolution spectrum, the derivatives with respect to the continuum and VMR is made on the so-called 'compressed grid' (the one made with only the '1' points of the irregular grid) , then a direct interpolation and convolution is performed.

Variables exchanged with external modules

Name	Description

imw	number of the actual Mw
itglev	number of the tangent-level for each geometry
igasmw	number of gases to be considered in each Mw
igasnr	global gas number for the local gas number of each Mw
isigma	number of wavenumber grid points for each Mw
rcross	absorption cross sections for each wavenumber each IAPT and each gas for the actual MW
rcol	column amounts for each layer, each geometry and each gas
raircol	air-column for each layer and each geometry
ipoint	IAPT-number for each layer and each geometry
ipath	number of different IAPT-numbers of ipoint
rtmain	equivalent temperature of the main gas
dsigma	general wavenumber fine grid
igeo	number of simulated geometries
iocsim	occupation matrix for the simulations to be performed
nsam	n. of sampling points in each Mw (general coarse grid)
nils	number of elements of rils
rspct	spectrum for each geometry on the general coarse grid 1st index: general wavenumber coarse grid 2nd index: geometries to be simulated for the actual Mw
rils	instrument-line-shape function on the general fine grid
rintils	ratio between the frequency step approximating infinitesimal spectral resolution and the integral of the ILS function
nrd	Ratio between general coarse grid step and fine grid step
rclay	model-layer values of the continuum
iderlay	highest (x,1), lowest (x,3) and middle (x,2) (the one directly above the 'perturbed' layer) which is affected by each derivative
igeocder	for each geometry the highest (x,1) and lowest (x,2) continuum derivative (in the parameter-grid) which has to be calculated
rpartcder	partial derivatives of the continuum layer values with respect to the parameter-level values
rspctcder	continuum derivative spectra on the general coarse grid for each geometry and each parameter level 1st index: general wavenumber coarse grid 2nd index: geometries to be simulated for the actual Mw 3rd index: levels where the parameters are retrieved
igeogder	for each simulated geometry *j* the highest (*igeotder*(*j*,1)) and lowest (*igeotder*(*j*,2)) parameter level which has to be considered for the vmr-derivatives
rpartgder	partial derivative of the main gas column of each layer with respect to the vmr parameter level values
rspctgder	vmr derivative spectra on the general coarse grid for each geometry and each parameter level 1st index: general wavenumber coarse grid 2nd index: geometries to be simulated for the actual Mw 3rd index: levels where the parameters are retrieved
cint	character*3: *cint(imxmw):* it indicates, for each microwindow, what kind of interpolation has to be performed between the spectral points of the irregular

	grid.
lirrgridmw	logical: *lirrgridmw(imxmw):* logical vector that, for each selected microwindow in the actual retrieval, indicates whether the irregular grid is available.
igridc	integer*4: igridc(imxsi2,imxmw): matrix that, to each microwindow and each point of the compressed grid, associates the corresponding value on the regular fine grid.
nused1	integer*4: nused1(imxmw): total number of points of the compressed grid for each microwindow
rsan	real*8: rsan(imxi,imxsi2,4,imxmw): variable used for making the direct interpolation/convolution. *rsan(jsam,i,n,imw)=* $\sum_{j=\max(((jsam-1)*nrd+1),igridc(i,imw))-(jsam-1)*nrd,k=\max(0,-igridc(i,imw)+((jsam-1)\cdot nrd+1)}^{j=\min(igridc(i+1,imw)-1,nils-1+(jsam-1)*nrd)-(jsam-1)*nrd,k=\min(igridc(i+1)-1-igridc(i),((jsam-1)\cdot nrd+nils-igridc(i))} rils(nils-j+1)\cdot k^{n-1}$
ilim	integer*4: ilim(2,imxi,imxmw): variable used for making the direct interpolation/convolution : *ilim(1,jsam,imw):* first point of the compressed grid to be considered for the computation of the low resolution spectral point at *jsam* for microwindow *imw*; *ilim(2,jsam,imw):* total number of points of the compressed grid to be considered for the computation of the low resolution spectral point at *jsam* for microwindow *imw*.logical: *lirrgridmw(imxmw):* logical vector that, for each selected microwindow in the actual retrieval, indicates whether the irregular grid is available.

Module structure:

1. Initialisation of variables and Planck function for later interpolation
Begin loop 1 over geometries valid for the actual microwindow
Begin loop 2 over general wavenumber fine grid
Begin condition 1: the value of the spectrum at this wavenumber has to be calculated and not interpolated
2. Interpolate Planck function
Begin loop 3 over the layers of the actual geometry
3. Calculation of the transmissions
End loop 3
4. Calculation of the radiative transfer
Begin loop 4 over the layers of the actual geometry for which the continuum derivatives are calculated
5. Calculation of the continuum derivatives with respect to the continuum layer values and the derivatives with respect to the column layer values.
End loop 4
Begin loop 5 over the levels for which the continuum derivatives are calculated
6. Calculation of the continuum derivatives with respect to the continuum level values
End loop 5

Begin loop 6 over the levels for which the vmr derivatives are calculated

7. Calculation of the vmr derivatives with respect to the vmr level values

End loop 6

End condition 1

End loop 2

Begin condition 2: irregular grid is available for the actual microwindow

Begin condition 3: cubic interpolation has to be used

8. Computation of coefficients for the cubic interpolation for spectrum, temperature perturbed spectrum and continuum derivative

9. Direct cubic interpolation / convolution

else condition 3: linear interpolation has to be used

10. Computation of coefficients for the linear interpolation for spectrum, temperature perturbed spectrum and continuum derivative

11. Direct linear interpolation / convolution

End condition 3

else condition 2

12. Convolution of the spectra and derivatives with the AILS function

End condition 2

End loop 1

Detailed description:

loop 1 over geometries valid for the actual microwindow
jgeo=1→*igeo*
if (*iocsim(jgeo,imw)*≠0)

loop 2 over general wavenumber fine grid
ksigma=1→*isigma(imw)*

condition 1: the value of the spectrum corresponding to point ksigma has to be calculated and not interpolated.
Operations 3-9 have to be performed only if either an irregular grid is not available for the considered mw or the irregular grid is available but the point ksigma corresponds to '1' on the irregular grid.
if (.not. lirrgridmw(imw) .or. (lirrgridmw(imw) .and. iigrid(ksigma, imw, 1) .eq. 1))

loop 3 over the layers of the actual geometry
klay=1→*nlay*
nlay=itglev(jgeo)-1 is the tangent layer.

loop 4 over the layers of the actual geometry for which the continuum derivatives are calculated
jder=iderlay(igeocder(jgeo,1),1)→nlay
iderlay(igeocder(jgeo,1),1) is the highest layer for which the continuum derivatives have to be determined.

loop 5 over the levels for which the continuum derivatives are calculated
jder=igeocder(jgeo,1)→igeocder(jgeo,2)

loop 6 over the levels for which the vmr derivatives are calculated
jder=igeogder(jgeo,1)→igeogder(jgeo,2)

Condition 2: irregular grid is available for the actual microwindow
Only if an irregular grid is available for the actual microwindow (if *lirrgridmw(imw)* is true), either operations 8. and 9. or operations 10. and 11. (i.e. direct interpolation / convolution) are performed, otherwise only convolution (12.) is performed.

Begin condition 3: cubic interpolation has to be used
if (cint(imw) .eq. 'cub' .or. 'CUB'), cubic interpolation has to be performed between the points of the spectrum on the compressed grid; if this is not the case, it means that *(cint(imw) .eq. 'lin' .or. 'LIN'),* and as consequence linear interpolation has to be performed between the points of the spectrum on the compressed grid.

1. Initialisation of variables and Plack function for later interpolation
Output variables set to 0.

The total number of points *nsig* of the grid to be used for the Radiative Transfer computation is determined. If an irregular grid is available, the compressed grid is used and *nsig= nused1(imw),* if the irregular grid is not available, *nsig= isigma(imw).*

The Planck function values at the first grid point and the last grid point of the actual microwindow and from this the increment for the later linear interpolation is calculated for the temperatures of the profiles of the main gas (*rtmain).* This is done for all different IAPT-numbers ($1 \le jpath \le ipath$). The formula used for the Planck function is:

$$B = \frac{rc1 \cdot \sigma^3}{\exp\left[\frac{rhck \cdot \sigma}{T}\right] - 1}$$

T = rtmain(jpath)
σ *= dsigma(1,imw)* or σ *= dsigma(isigma(imw),imw)*
(rc1, rhck: parameters)

Care has to be taken to perform a correct interpolation of the Plank function when the compressed grid is used (i.e. if lirrgridmw(imw) = true): in this case the value of the Planck function corresponding to the actual point *i* of the compressed grid is obtained adding to the Planck function value at the first grid point the product of the coefficient of the linear interpolation times *(igridc(i,imw)-1).*

2. Interpolate Planck function
Using the values calculated in 1. the Planck function is linearly interpolated to the actual wavenumber for all IAPT numbers (*jpath=1→ipath)*:
The results are the interpolated Planck function values for the T-profile (*rtmain):*
db(jpath),

3. Calculation of the transmission

The transmission for each layer is calculated by the formula:

$$rtau(klay) = \exp-\left[\begin{array}{l} rclay(klay, imw)\cdot raircol(klay, imw)\cdot 10^{-30} + \\ \sum_{mgas=1}^{igas}\left\{\begin{array}{l} rcross(ksig, ipoint(klay,\ jgeo), mgas)\cdot \\ rcol(klay,\ jgeo, igasnr(mgas, imw)) \end{array}\right\} \end{array}\right]$$

Two other variables are also determined:

$$rtau1(klay) = \prod_{l=1}^{klay-1} rtau(l)$$

and:

$$rtau2(klay) = rtau1(klay)\cdot rtau(klay)\cdot \prod_{l=klay+1}^{nlay} rtau(l)^2$$

with: $nlay = itglev(jgeo) - 1$, the number of layers for the actual geometry,

and the definition: $\prod_{l=m}^{m-1} x_l \equiv 1$.

4. Calculation of the radiative transfer

The spectrum is determined by the equation:

$$rsp(ksig) = \sum_{klay=1}^{nlay} db(ipoint(klay, jgeo))\cdot\left(1 - rtau(klay)\right)\left(rtau1(klay) - rtau2(klay)\right)$$

with: $nlay = itglev(jgeo) - 1$,

and *db*, the value of the Planck function for each IAPT-number. *db* was determined in 3. by linear interpolation to the actual general fine grid wavenumber.

5. Calculation of the continuum derivatives with respect to the continuum layer values and the derivatives with respect to the column layer values

In this section the continuum derivatives with respect to the layer values of the continuum and the derivatives with respect to the layer columns of the main gas are calculated for the layers (*jder*).

The formula for the continuum is:

$$rcder2(jder) = -raircol(jder, jgeo)\cdot 10^{-30}\cdot$$

$$\left[\begin{array}{l} \sum_{klay=1}^{jder-1} 2\cdot rtau2(klay)\cdot db(ipoint(klay, jgeo))\cdot(1-rtau(klay)) \\ +db(ipoint(jder, jgeo))\cdot\left(\begin{array}{l} rtau2(jder)- \\ rtau(jder)\cdot(rtau1(jder)+2\cdot rtau2(jder)) \end{array}\right) \\ +\sum_{klay=jder+1}^{nlay}\left(\begin{array}{l} db(ipoint(klay, jgeo))\cdot(1-rtau(klay))\cdot \\ (rtau1(klay)+rtau2(klay)) \end{array}\right) \end{array}\right]$$

and formula for the main gas is:

$$rgder2(jder) = -rcross(ksig, ipoint(klay, jgeo),1)\cdot$$

$$\left[\begin{array}{l} \sum_{klay=1}^{jder-1} 2\cdot rtau2(klay)\cdot db(ipoint(klay, jgeo))\cdot(1-rtau(klay)) \\ +db(ipoint(jder, jgeo))\cdot\left(\begin{array}{l} rtau2(jder)- \\ rtau(jder)\cdot(rtau1(jder)+2\cdot rtau2(jder)) \end{array}\right) \\ +\sum_{klay=jder+1}^{nlay}\left(\begin{array}{l} db(ipoint(klay, jgeo))\cdot(1-rtau(klay))\cdot \\ (rtau1(klay)+rtau2(klay)) \end{array}\right) \end{array}\right]$$

6. Calculation of the continuum derivatives with respect the continuum level values

The continuum derivatives with respect to the continuum level values (*rcder*) are determined by using the results from 6. which are multiplied by the input *rpartcder*:

$$rcder(ksig, jder) = \sum_{klay=iderlay(jder,1)}^{iderlay(jder,3)} rcder2(klay)\cdot rpartcder(klay, jder, imw)$$

7. Calculation of the vmr derivatives with respect the vmr level values

The vmr derivatives with respect to the vmr level values are determined by using the results from 6. which are multiplied by the input *rpartgder*:

$$rgder(ksig, jder) = \sum_{klay=iderlay(jder,1)}^{iderlay(jder,2)} rgder2(klay)\cdot rpartgder(klay, jgeo,1) + \sum_{klay=iderlay(jder,2)+1}^{iderlay(jder,3)} rgder2(klay)\cdot rpartgder(klay, jgeo,2)$$

8. Computation of coefficients for the cubic interpolation for spectrum, continuum and VMR derivatives

For each point *i* of the compressed grid between *2* and *(nused1(imw)-2),*

do i=2,nsig-2

the coefficients of the cubic interpolation *a, b, c* according to the following equation:

$$y = y_2 + a \cdot (x - x_2)^3 + b \cdot (x - x_2)^2 + c \cdot (x - x_2),$$

with (x_2, y_2) coordinates of the second of the four points used for making the interpolation,

for spectrum, continuum and VMR derivatives, are computed in two steps. First of all the variables which are independent on the value of the spectrum, continuum and VMR derivatives are computed:

```
ii2=igridc(i,imw)
ii3=igridc(i+1,imw)
ii1=igridc(i-1,imw)
ii4=igridc(i+2,imw)
iD12 =  ii1 - ii2
iD13 =  ii1 - ii3
iD14 =  ii1 - ii4
iD23 =  ii2 - ii3
iD24 =  ii2 - ii4
iD34 =  ii3 - ii4
 rdc1=1.d0/dble(iD12*iD13*iD14)
 rdc2=1.d0/dble(iD12*iD23*iD24)
 rdc3=1.d0/dble(iD13*iD23*iD34)
 rdc4=1.d0/dble(iD14*iD24*iD34)
```

Then the variables c1, c2, c3, c4, dependent on the four points through which the interpolating polynomial is drawn, are computed: in the case of the spectrum we have:

```
c1=rsp(i-1)*rdc1
c2=-rsp(i)*rdc2
c3=rsp(i+1)*rdc3
c4=-rsp(i+2)*rdc4
```

In the case of VMR derivatives:

```
c1=rgder(i-1,jder)*rdc1
 c2=- rgder(i,jder)*rdc2
 c3= rgder (i+1,jder)*rdc3
 c4=- rgder(i+2,jder)*rdc4,
```

jder= igeogder(jgeo,1) $\rightarrow$ *igeogder(jgeo,2)*

In the case of continuum derivatives:

*c1=rcder(i-1,jder)*rdc1*
*c2=- rcder(i,jder)*rdc2*
*c3= rcder (i+1,jder)*rdc3*
*c4=- rcder(i+2,jder)*rdc4,*

jder= igeocder(jgeo,1) → igeocder(jgeo,2)

The *a, b, c* coefficients are determined by the following formula:

```
a(i)=c1+c2+c3+c4
b(i)= =-(c1*dble(ii4+ii3+ii2)+c2*dble(ii4+ii3+ii1)+c3*dble(ii4+ii2+ii1)+
        c4*dble(ii3+ii2+ii1))+3.0d0*a(i)*dble(ii2)
c(i)= c1*dble(ii4*ii3+ii4*ii2+ii2*ii3)+c2*dble(ii1*ii3+ii4*ii1+ii4*ii3)+
      c3*dble(ii1*ii2+ii4*ii1+ii4*ii2)+c4*dble(ii1*ii2+ii3*ii1+ii2*ii3)+
      +3.0d0*a(i)*dble(ii2*ii2) -2.0d0*dble(ii2)*(c1*dble(ii4+ii3+ii2)+
      +c2*dble(ii4+ii3+ii1)+c3*dble(ii4+ii2+ii1)+c4*dble(ii3+ii2+ii1))
```

The coefficients *a, b, c* for the VMR derivatives are stored in the following matrices:

agder(i, jder = igeogder(jgeo,1) → igeogder(jgeo,2))
bgder(i, jder = igeogder(jgeo,1) → igeogder(jgeo,2))
cgder(i, jder = igeogder(jgeo,1) → igeogder(jgeo,2))

The coefficients for the continuum derivatives are stored in the following matrices:

ader(i, jder= igeocder(jgeo,1) → igeocder(jgeo,2))
bder(i, jder= igeocder(jgeo,1) → igeocder(jgeo,2))
cder(i, jder= igeocder(jgeo,1) → igeocder(jgeo,2))

9. Direct cubic convolution and interpolation

For the meaning of subsequent computations, please refer to description of routine read_irregular grid_pt, par. 2.2.30.

The direct interpolation / convolution is performed using the coefficients computed in 8. and the matrices *rsan(imxi,imxsi2,4,imxmw)* and *lim(2,imxi,imxmw)* computed by routine read_irrgrid_vmr.f.

For each *jsam* between *2* and *(nsam(imw)-1),* the value of the low resolution spectrum is computed using the following formula:

$$rspct(jsam, jgeo) = \sum_{i\lim(1,jsam,imw)}^{i\lim(1,jsam,imw)+i\lim(2,jsam,imw)-1} \begin{pmatrix} rsp(i)\cdot rsan(jsam,i,1,imw) + a(i)\cdot rsan(jsam,i,4,imw) + \\ +b\cdot rsan(jsam,i,3,imw) + c\cdot rsan(jsam,i,2,imw) \end{pmatrix}$$

At the end the low resolution spectrum is normalised:

$rspct(jsam, jgeo) = rspct(jsam, jgeo) * rintils(imw)$

Since the first and the last point of the regular fine grid had not been taken into account during the computation of *rsan*, an addition summation has to be performed for *jsam =1* and *jsam=nsam(imw).*

For *jsam=1*:

$$rspct(1, jgeo) = rsp(1) \cdot rils(nils, imw) + \sum_{i\lim(1,1,imw)}^{i\lim(1,1,imw)+i\lim(2,1,imw)-1} \begin{pmatrix} rsp(i) \cdot rsan(1,i,1,imw) + a(i) \cdot rsan(1,i,4,imw) + \\ +b \cdot rsan(1,i,3,imw) + c \cdot rsan(1,i,2,imw) \end{pmatrix}$$

$rspct(1, jgeo) = rspct(1, jgeo) * rintils(imw)$

For *jsam=j=nsam(imw):*

$$rspct(j, jgeo) = rsp(nsig) \cdot rils(1, imw) + \sum_{i\lim(1,j,imw)}^{i\lim(1,j,imw)+i\lim(2,j,imw)-1} \begin{pmatrix} rsp(i) \cdot rsan(j,i,1,imw) + a(i) \cdot rsan(j,i,4,imw) + \\ +b \cdot rsan(j,i,3,imw) + c \cdot rsan(j,i,2,imw) \end{pmatrix}$$

$rspct(j, jgeo) = rspct(j, jgeo) * rintils(imw)$

The same operations have to be performed also for all the VMR derivatives (*rspctgder(jsam=1→nsam(imw),jgeo,jder = igeogder(jgeo,1) → igeogder(jgeo,2))*) and the continuum derivatives (*rspctcder(jsam=1→nsam(imw),jgeo,jder = igeocder(jgeo,1)→igeocder(jgeo,2))*).

```
      j = nsam(imw)
C     RSPCT
      r8fac = rsp(1)*rils(nils,imw)
      DO i = ilim(1,1,imw), ilim(1,1,imw)+ilim(2,1,imw)-1
       r8fac = r8fac + rsp(i)*rsan(1,i,1,imw) +
     &                 a(i)*rsan(1,i,4,imw) +
     &                 b(i)*rsan(1,i,3,imw) +
     &                 c(i)*rsan(1,i,2,imw)
      ENDDO
      rspct(1,jgeo) = r8fac*rintils(imw)
      DO jsam = 2, nsam(imw)-1
       r8fac = 0.0D0
       DO i = ilim(1,jsam,imw),
     &        ilim(1,jsam,imw)+ilim(2,jsam,imw)-1
         r8fac = r8fac + rsp(i)*rsan(jsam,i,1,imw) +
     &                   a(i)*rsan(jsam,i,4,imw) +
     &                   b(i)*rsan(jsam,i,3,imw) +
     &                   c(i)*rsan(jsam,i,2,imw)
       ENDDO
```

```
          rspct(jsam,jgeo) = r8fac*rintils(imw)
        ENDDO
        r8fac = rsp(nsig)*rils(1,imw)
        DO i = ilim(1,j,imw), ilim(1,j,imw)+ilim(2,j,imw)-1
          r8fac = r8fac + rsp(i)*rsan(j,i,1,imw) +
     &                a(i)*rsan(j,i,4,imw) +
     &                b(i)*rsan(j,i,3,imw) +
     &                c(i)*rsan(j,i,2,imw)
        ENDDO
        rspct(j,jgeo) = r8fac*rintils(imw)
C     RSPCTGDER
        DO jder = igeogder(jgeo,1), igeogder(jgeo,2)
          r8fac = rgder(1,jder)*rils(nils,imw)
          DO i = ilim(1,1,imw), ilim(1,1,imw)+ilim(2,1,imw)-1
            r8fac = r8fac + rgder(i,jder)*rsan(1,i,1,imw) +
     &                agder(i,jder)*rsan(1,i,4,imw) +
     &                bgder(i,jder)*rsan(1,i,3,imw) +
     &                cgder(i,jder)*rsan(1,i,2,imw)
          ENDDO
          rspctgder(1,jgeo,jder) = r8fac*rintils(imw)
          DO jsam = 2, nsam(imw)-1
            r8fac = 0.0D0
            DO i = ilim(1,jsam,imw),
     &           ilim(1,jsam,imw)+ilim(2,jsam,imw)-1
              r8fac = r8fac + rgder(i,jder)*rsan(jsam,i,1,imw) +
     &                agder(i,jder)*rsan(jsam,i,4,imw) +
     &                bgder(i,jder)*rsan(jsam,i,3,imw) +
     &                cgder(i,jder)*rsan(jsam,i,2,imw)
            ENDDO
            rspctgder(jsam,jgeo,jder) = r8fac*rintils(imw)
          ENDDO ! jsam
          r8fac = rgder(nsig,jder)*rils(1,imw)
          DO i = ilim(1,j,imw), ilim(1,j,imw)+ilim(2,j,imw)-1
            r8fac = r8fac + rgder(i,jder)*rsan(j,i,1,imw) +
     &                agder(i,jder)*rsan(j,i,4,imw) +
     &                bgder(i,jder)*rsan(j,i,3,imw) +
     &                cgder(i,jder)*rsan(j,i,2,imw)
          ENDDO
          rspctgder(j,jgeo,jder) = r8fac*rintils(imw)
        ENDDO   ! jder
C     RSPCTCDER
        DO jder = igeocder(jgeo,1), igeocder(jgeo,2)
          r8fac = rcder(1,jder)*rils(nils,imw)
          DO i = ilim(1,1,imw),ilim(1,1,imw)+ilim(2,1,imw)-1
            r8fac =r8fac + rcder(i,jder)*rsan(1,i,1,imw) +
     &                ader(i,jder)*rsan(1,i,4,imw) +
     &                bder(i,jder)*rsan(1,i,3,imw) +
     &                cder(i,jder)*rsan(1,i,2,imw)
          ENDDO
```

```
      rspctcder(1,jgeo,jder) = r8fac*rintils(imw)
      DO jsam = 2, nsam(imw)-1
       r8fac = 0.0D0
       DO i = ilim(1,jsam,imw),
     &        ilim(1,jsam,imw)+ilim(2,jsam,imw)-1
        r8fac = r8fac + rcder(i,jder)*rsan(jsam,i,1,imw) +
     &                  ader(i,jder)*rsan(jsam,i,4,imw) +
     &                  bder(i,jder)*rsan(jsam,i,3,imw) +
     &                  cder(i,jder)*rsan(jsam,i,2,imw)
       ENDDO
       rspctcder(jsam,jgeo,jder) = r8fac*rintils(imw)
      ENDDO ! jsam
      r8fac = rcder(nsig,jder)*rils(1,imw)
      DO i = ilim(1,j,imw), ilim(1,j,imw)+ilim(2,j,imw)-1
       r8fac = r8fac + rcder(i,jder)*rsan(jsam,i,1,imw) +
     &                 ader(i,jder)*rsan(jsam,i,4,imw) +
     &                 bder(i,jder)*rsan(jsam,i,3,imw) +
     &                 cder(i,jder)*rsan(jsam,i,2,imw)
      ENDDO
      rspctcder(j,jgeo,jder) = r8fac*rintils(imw)
     ENDDO ! jder = igeocder(jgeo,1), igeocder(jgeo,2)
```

10. Computation of coefficients for the linear interpolation for spectrum, temperature perturbed spectrum and continuum derivative

For each point *i* of the compressed grid between *1* and *(nused1(imw)-1),*

```
do i=1,nsig-1
```

the coefficient of the linear interpolation *rm* for spectrum, continuum and VMR derivatives, is computed as follows:
first of all the variables which are independent on the value of the spectrum, continuum and VMR derivatives are computed:

```
ii2=igridc(i,imw)
ii3=igridc(i+1,imw)
rdc1=1.d0/dble(ii3-ii2)
```

The coefficient *rm(i)* is determined by the following formula:

```
rm(i)=rdc1*(rsp(i+1)-rsp(i))
```

The coefficient *rm* is computed for all continuum and VMR derivatives.

The coefficient *rmgder(i, jder = igeogder(jgeo,1) → igeogder(jgeo,2))* for the temperature perturbed spectra is equal to:

*rmgder(i,jder)= (rgder(i+1,jder)-rgder(i,jder))*rdc1*

The coefficient *rmcder(i, jder = igeocder(jgeo,1) → igeocder(jgeo,2))* for the continuum derivatives is equal to:

*rmcder(i,jder)= (rcder(i+1,jder)-rcder(i,jder))*rdc1*

11. Direct linear interpolation / convolution

For the meaning of subsequent computations, please refer to description of routine read_irregular grid_pt, par. 2.2.30.

The direct interpolation / convolution is performed using the coefficients computed in 10. and the matrices *rsan(imxi,imxsi2,4,imxmw)* and *lim(2,imxi,imxmw)* computed by routine read_irrgrid_vmr.f.

For each *jsam* between *1* and *(nsam(imw))*, the value of the low resolution spectrum is computed using the following formula:

$$rspct(jsam, jgeo) = \sum_{i\lim(1,jsam,imw)}^{i\lim(1,jsam,imw)+i\lim(2,jsam,imw)-1} \left(rsp(i) \cdot rsan(jsam, i, 1, imw) + rm(i) \cdot rsan(jsam, i, 2, imw) \right)$$

At the end the low resolution spectrum is normalised:

*rspct(jsam, jgeo) = rspct(jsam, jgeo) *rintils(imw)*

The same operations have to be performed also for all the VMR derivatives (*rspctgder(jsam=1→nsam(imw),jgeo,jder = igeogder(jgeo,1) → igeogder(jgeo,2))*) and the continuum derivatives (*rspctcder(jsam=1→nsam(imw),jgeo,jder = igeocder(jgeo,1)→igeocder(jgeo,2))*).

```
        DO jsam = 1, nsam(imw)
C    RSPCT
        r8fac = 0.0D0
        DO i = ilim(1,jsam,imw),
     &          ilim(1,jsam,imw)+ilim(2,jsam,imw)-1
         r8fac = r8fac + rsp(i)*rsan(jsam,i,1,imw) +
     &                  rm(i)*rsan(jsam,i,2,imw)
        ENDDO
        rspct(jsam,jgeo) = r8fac*rintils(imw)
C    RSPCTGDER
        DO jder = igeogder(jgeo,1), igeogder(jgeo,2)
         r8fac = 0.0D0
         DO i = ilim(1,jsam,imw),
     &          ilim(1,jsam,imw)+ilim(2,jsam,imw)-1
          r8fac = r8fac + rgder(i,jder)*rsan(jsam,i,1,imw) +
     &                  rmgder(i,jder)*rsan(jsam,i,2,imw)
         ENDDO
         rspctgder(jsam,jgeo,jder) = r8fac*rintils(imw)
        ENDDO ! jder = igeogder(jgeo,1), igeogder(jgeo,2)
C    RSPCTCDER
        DO jder = igeocder(jgeo,1), igeocder(jgeo,2)
         r8fac = 0.0D0
         DO i = ilim(1,jsam,imw),
     &          ilim(1,jsam,imw)+ilim(2,jsam,imw)-1
          r8fac = r8fac + rcder(i,jder)*rsan(jsam,i,1,imw) +
     &                  rmcder(i,jder)*rsan(jsam,i,2,imw)
         ENDDO
         rspctcder(jsam,jgeo,jder) = r8fac*rintils(imw)
        ENDDO ! jder = igeocder(jgeo,1), igeocder(jgeo,2)
       ENDDO   ! jsam = 1, nsam(imw)
```

12. Convolution of the spectra and derivatives with the AILS function

In a call to module **conv_vmr** the convolution with the AILS function *rils* is performed for the original spectrum *rsp*, for the continuum derivatives *rcder* and for the VMR derivatives *rgder*. The results are the spectra and derivatives on the general coarse wavenumber grid: *rspct*, *rspctcder*, *rspctgder*.

3.2.12 ABCALC_VMR

Description

This module calculates the matrices $\mathbf{A} = \mathbf{K}^T\mathbf{S}^{-1}\mathbf{K}_1$ and $\mathbf{B}^T = \left(\mathbf{K}^T\mathbf{S}^{-1}\right)^T$ (see AD6 for the definition of these matrices).

Variables exchanged with external modules

Name	Description
rjacob	The **K** matrix (used dimensions (itop*iobs))
rvcmobinv	Inverse of the VCM of the observations, not divided by the square of the noise
ra	**A** matrix of [AD6]
rbt	**Transpose of the B** matrix of [AD6]
iobs	total N. of observations
itop	total n. of unknown parameters
nselmw	Number of selected microwindows (used to build S^{-1} matrix)
ilimbmw	ilimbmw(imxmw) = n. of sweeps at which the current MW is used
nsam	nsam(imxmw) = n. of sampling points in each MW (coarse grid)
rnoise	Noise used to build the S^{-1} matrix
ilimb	ilimb = N. of considered sweeps
lokku	lokku(imxgeo,imxmw) = MW occupation matrix

Module structure

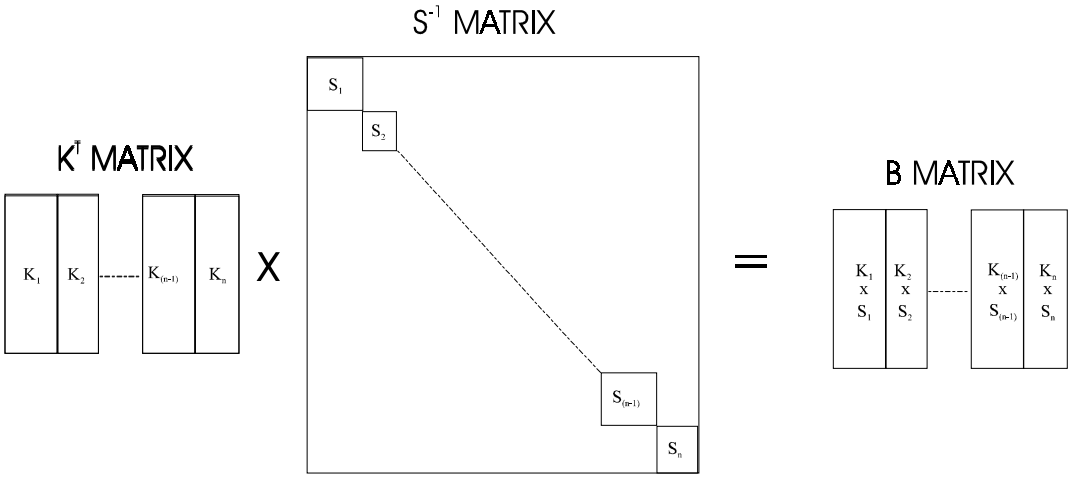
This module computes **A**, and transpose of B, $\mathbf{B^t}$, matrices. The $\mathbf{S^{-1}}$ matrix is a block diagonal matrix. The square sub-block referring to the *j-th* MW has dimension equal to *nsam(j)*. The number '*Nblocks*' of blocks of $\mathbf{S^{-1}}$ is given by the summation on all the microwindows (*j=1..nselmw*) of *ilimbmw(j)*.

- Step 1

The rjacob matrix (*itop,iobs*) is divided in blocks. The block referring to a considered geometry of microwindow *j* has dimensions (*itop*nsam(j)*).

- Step 2

Each block is multiplied by the corresponding *j-th* square block (dimension *nsam(j)· nsam(j*) of *rvcmobinv* matrix; the result is copied in the corresponding block of matrix B.



- Step 3

Matrix **B** is multiplied by **K** matrix to get **A** matrix.

- Step 4

If LOS info is to be used it calculates matrix $\mathbf{B}_1$ and adds the contribution $\mathbf{K}_1^T(\mathbf{V}^z)^{-1}\mathbf{K}_1$ to matrix **A.**

Detailed description

```
* Standard calculation (A=K'*S^-1*K):
      icount=0
           do j=1,nselmw
             do k=1,ilimb
            if (lokku(k,j))then
C           Set up constant multiplier (instead of divider)
           rnoise2 = SNGL(1.0D0/rnoise(j,k)**2)
* Multiplication rjacob * rvcmobinv (Kt * V**-1):
              do l=1, itop
             do m=1, nsam(j)
C             Inititialise accumulator with first element and shorten loop
             rtemp1 = rjacob(1+icount,l)*rvcmobinv(1,m,j)
             do m1 = 2, nsam(j)
C               Single precision arithmetic
              rtemp1 = rtemp1 +
     &                      rjacob(m1+icount,l)*rvcmobinv(m1,m,j)
             end do
              rbt(icount+m,l)=rtemp1*rnoise2
             end do
             end do

                 icount = icount + nsam(j)
                 end if
              end do     ! end loop LS (k=1,...ilimb)
             end do      ! end loop MWs (j=1,...nselmw)

* Multiplication of rb *rjacob ---> ra
      do k=1,itop
       do j=1,k
C        Inititialise accumulator with first element and shorten loop
C        Use single precision arithmetic
         rtemp1 = rbt(1,j)*rjacob(1,k)
         do l = 2, iobs
          rtemp1 = rtemp1 + rbt(l,j)*rjacob(l,k)
         end do
         r1 = DBLE(rtemp1)
         ra(j,k) = r1
         ra(k,j) = r1
       end do
      end do

      return
      end
```

3.2.13 DIFCHI_VMR

DIFCHI_VMR]
|-----CHISQ_VMR *

Description
It calculates the χ^2 function that has to be minimised in retrieval procedure.
After the computation of the vector of the residuals (*rnres*) from the observed (*robs*) and simulated (*rspfov*) spectra, it performes the matrix product between the transpose of *rnres* and *rnres*, weighted by the inverse of the variance covariance matrix of the observations (*rvcmobinv).*

Variables exchanged with external modules:

Name	Description
iobs	total number of observations
itop	total number of parameters to be fitted
robs	robs(imxi,imxgeo,imxmw)): observed spectra corresponding to the different tangent altitudes and different microwindows (on the coarse frequency grid)
rspfov	rspfov(imxi,imxgeo,imxmw): simulated spectra corresponding to the different tangent pressures and different microwindows on the frequency coarse grid: (rspct * FOV)
rvcmobinv **real*4**	rvcmobinv(imxi,imxi): elementary block of the inverse of the variance covariance matrix of the observations associated to the wider microwindow.
rnoise	rnoise(imxmw,imxgeo):NESR dependent on geometry and microwindow
nsam	nsam(imxmw): no. of sampling points in each MW (coarse grid)
nselmw	total number of selected microwindows for the retrieval
ilimb	number of measured geometries
lokku	lokku(imxgeo,imxmw) occupation matrix used for the selection of operational MW's for each observation geometry
ilimbmw	ilimbmw(imxmw): number of valid measured geometries per microwindow number of 2 in each column of iocsim)
iterg	iterg = index of the actual iteration
rnres	rnres(imxobs) : vector of the differences between the observated spectraand the calculated ones;first all the geometries of the first microwindow starting from the first geometry, then all the other microwindows
rchisq	rchisq(0:imxite): total chi-square for each iteration
rchisqp	rchisqp(imxlmb,imxmw) chi-square for each observation geometry and each microwindow

Module structure
1. Calculation of the vector of the residuals
2. Control on the correctness of the computations
3. Definition of *lpart*
4. Calculation of chi-square
5. Storage of chi-square.

Detailed description

```
* Calculation of the residuals vector:
     jobs=0
     do imw=1,nselmw
      kgeo1=0
      do kgeo=1,ilimb
       if (lokku(kgeo,imw)) then
        kgeo1=kgeo1+1
        do jsig=1,nsam(imw)
          jobs=jobs+1
          rnres(jobs) = robs(jsig,kgeo,imw) - rspfov(jsig,kgeo1,imw)
        end do
       end if
      end do
     end do
* Internal consistency check (the program never stops here if there are no bugs):
     if (iobs.ne.jobs)stop 'program stopped in difchi'
* Since also the partial chi-square has to be computed we set lpart = TRUE in order to ask ‘chisq’
* for the computation of the partial chi-square.
     lpart=.true.
* Calculation of chi-square by using chisq_vmr module
     call chisq_vmr(rnres, iobs, itop, nselmw, ilimbmw, nsam, rnoise, rvcmobinv, lpart, rchi, rchisqp,
ilimb,
                    lokku, rnreslos, rinvclos, lextinf1)
* Storage of the computed chi-square
     rchisq(iterg)=rchi
     end
```

3.2.13.1 CHISQ_VMR

Description

It calculates χ^2 value, performing the matrix product between the transpose of *rnres* and *rnres*, weighted by the inverse of the variance covariance matrix of the observations (*rvcmobinv)*. If this subroutine is called with ‘lpart’ = TRUE, it calculates also the partial chi-square related to the differnt MWs and sweeps.

Variables exchanged with external modules

Name	Description
rnres	rnres(imxobs): vector of the differences between the observed spectra and the calculated ones; first all the geometries of the first microwindow starting from the first geometry, then all the other microwindows
iobs	total number of observations
itop	total number of parameters to be fitted
nselmw	total number of selected microwindows for the retrieval
ilimbmw	ilimbmw(imxmw): number of valid measured geometries per microwindow (number of 2 in each column of iocsim)

nsam	nsam(imxmw): n. of sampling points in each MW (coarse grid)
rnoise	rnoise(imxmw,imxgeo): NESR dependent on geometry and microwindow
rvcmobinv **real*4**	rvcmobinv(imxi,imxi): elementary block of the inverse of the variance covariance matrix of the observations related to the widest microwindow.
lpart	switch for enabling the storage of the partial chi-square
rchi	returned value of the χ^2 function
rchisqp	rchisqp(imxlmb,imxmw) chi-square for each observation geometry and each microwindow temperature profiles
ilimb	number of measured sweeps
lokku	lokku(imxgeo,imxmw): MW occupation matrix used for the selection of operational MW's for each observation geometry.

Module structure:

1. Calculation of the matrix product: $(rnres)^T \cdot (S)^{-1} \cdot (rnres)$
2. Storage of partial chi-square
3. Calculation of total reduced chi-square

Detailed description:

```
* Calculation of the n. of degrees of freedom 'ifrede' of the problem: observations - parameters
     ifrede = iobs - itop
* Some initialisations:
     rchi=0.      ! initialisation of chi-square
     inres=0      ! index of vector rnres (vector of the residuals)
* calculation of the chi-square:
     do 10 imw=1,nselmw
       kgeo1=0
       do 20 kgeo=1,ilimb
         if (lokku(kgeo,imw))then
           kgeo1=kgeo1+1
           rchi1=0.
           do 30 jsig=1,nsam(imw)
             rpart(jsig)=0.
             do 40 jsig1=1,nsam(imw)
               rpart(jsig) = rpart(jsig) + rnres(jsig1+inres) * rvcmobinv(jsig1,jsig,imw)  ! transpose of
nres * rvcmobinv
 40          continue
             rchi1=rchi1+rnres(jsig+inres)*rpart(jsig)
 30        continue
           inres=inres+nsam(imw)
           rchi1=rchi1/(rnoise(imw,kgeo)*rnoise(imw,kgeo))
           if(lpart)rchisqp(kgeo1,imw)=rchi1
           rchi=rchi+rchi1
         endif
20     continue
10   continue
```

```
*     Calculation of the total reduced chi-square:
       rchi = rchi / ifrede

      end
```

3.2.14 AMODIF_VMR

Description
Multiplication of the diagonal elements of the matrix *ra* by (1+*rlambda*).

Variables exchanged with external modules

Name	Description
ra	matrix defined as (transpose of rjacob) * rvcmobinv * rjacob (as the output the diagonal elements are multiplied with 1+*rlambda*
rlambda	Marquardt damping factor
itop	total number of parameters to be fitted
ipar	ipar = number of parameter-levels (i.e. N. of elements of rzpar vector)
icontpar	n. of fitted continuum parameters

Module structure
1. Multiplication of the diagonal elements with 1+*rlambda*. A personalised damping factor is used for the elements which correspond to continuum parameters.

Detailed description

1. Multiplication of the diagonal elements by 1+*rlambda*:

For $1 \le j \le itop$:

if $j > ipar$ AND $j \le ipar + icontpar$ then
$ra(j,j) = ra(j,j) \cdot (1 + rlambda * 100)$
else
$ra(j,j) = ra(j,j) \cdot (1 + rlambda)$
end if

3.2.15 NEWPAREST_VMR

NEWPAREST_VMR]
|((((+CHISQ_VMR *

Description
Calculates the new estimate of the vector of the unknown parameters *rxpar* and, if *iterm*=0, calculates the χ^2 in the linear approximation as well.

Variables exchanged with external modules

Name	Description
rainv	matrix inverse of *ra*
rbt	matrix B^t defined as transpose((transpose of *rjacob*) * *rvcmobinv)*
rnres	vector of the differences between the observed spectra and the calculated ones
rxparold	vector of the fitted parameters at the previous iteration
itop	total number of parameters to be fitted
iobs	total number of observations to be fitted
iterm	micro - iteration index (Marquardt)
rjacob	Jacobian Matrix
rxpar	vector of the fitted parameters
rlinchisq	χ^2 calculated in the linear approximation
rvcmobinv (real*4)	elementary block of inverse of the variance covariance matrix of the observations associated to the widest microwindow
rnoise	NESR dependent on geometry and microwindow
nsam	number of sampling points in each MW (general coarse grid)
nselmw	total number of selected microwindows for the retrieval
ilimbmw	number of valid measured geometries per microwindow (total number of '2's in each column of *iocsim*)
ilimb	number of measured geometries (sweeps)
lokku	occupation matrix used for the selection of operational MW's for each observation geometry

Module structure

1. Set *rxparold = rxpar*
2. Calculate the correction for the parameters
3. If the routine is called during a macro-iteration, the linear χ^2 is computed
4. Calculation of the new parameters

Detailed description

```
* Makes the backup of the parameters vector:
      do jpar=1,itop
        rxparold(jpar) = rxpar(jpar)
      end do

*     Initialisation of rxpar and local variable r2v

      do 15 k=1,itop
        r2v(k)=0.d0
        rxpar(k)=0.d0
15    continue

* calculates the correction parameter vector: y=(A^-1)Bn (rxpar is overwritten by this!!)

       do 20 k=1,itop
```

```
      do 30 l=1,iobs
         r2v(k)=r2v(k)+rbt(l,k)*rnres(l)
30     continue
20    continue

     do 40 k=1,itop
       do 45 jpar=1,itop
         rxpar(jpar)=rxpar(jpar)+rainv(jpar,k)*r2v(k)
45     continue
40    continue


* if iterm=0 (macro-iteration) it calculates the 'linear difference vector' of the observations: n{lin} =
n - K y
* i.e. rnreslin = rnres - rjacob * rxpar
     if (iterm.eq.0) then                    ! begin condition on macro-iteration
       do 50 jobs=1,iobs
         r1=0.
         do 60 kpar=1,itop
           r1=r1+rjacob(jobs,kpar)*rxpar(kpar)
60       continue
         rnreslin(jobs)=rnres(jobs)-r1
50     continue

* calculates the linear chi square; rchisqp is not calculated new for the linear chi square:
       lpart=.false.
       call chisq_pt(rnreslin, iobs, itop, nselmw, ilimbmw, nsam, rnoise, rvcmobinv, lpart, rlinchisq,
rchisqp, ilimb,                    lokku, rnresloslin,rinvclos,lextinf1)
     end if                                ! end condition on macro-iteration

* calculates the new estimate of the parameters vector:
     do 70 jpar=1,itop
        rxpar(jpar)=rxparold(jpar) + rxpar(jpar)
70  continue
     end
```

3.2.16 UPDPROF_VMR

UPDPROF_VMR]
|((((+FICARRA_VMR *

Description: updates the VMR profile of the main gas (retrieved gas), continuum profiles and instrumental offsets on the basis of the new estimate of the parameters vector *rxpar*.

Variables exchanged with external modules:

Name:	Description:
rxpar	rxpar(imxtop) = vector of the fitted parameters
itop	itop = total number of parameters to be fitted
ipar	ipar = number of parameter-levels (i.e. N. of elements of rzpar vector)
rzpar	rzpar(imxlmb) = vector of the altitudes where the temperature profile is fitted
rzbase	rzbase(imxpro) = altitude of the base-levels
ibase	ibase = number of base-levels
rcbase	rcbase(imxpro,imxmw) = continuum on the base-levels for each MW
nselmw	nselmw = total number of selected microwindows for the retrieval
rvmrbase	rvmrbase(imxpro,imxgas) = volume mixing ratio of the gases on the base levels
igas	igas = total number of different gases
roffs	roffs(imxmw) = instrumental offsets personalised for microwindow
lparbase	lparbase(imxpro) = logical vector which identifies the altitudes where the T profile is fitted, among the altitudes rzbase.
lokku	lokku(imxgeo,imxmw) = occupation matrix used for the selection of operational MW's for each observation geometry
ilimb	ilimb = number of measured geometries
ilimbmw	ilimbmw(imxmw) = number of valid measured geometries per microwindow (number of 2 in each column of iocsim)
icontpar	icontpar = total number of continuum parameters to be fitted
isaved	isaved(imxsav) = vector containing all the necessary quantities for the reconstruction of continuum profiles performed by *ficarra* subroutine
nsam	nsam(imxmw) = number of sampling points in each MW (general coarse grid)
ifspmw	ifspmw(imxmw) = index of the first sampling point of each MW * NOTE: the sampling point at frequency=0 has index=1
dstep	dstep = distance between coarse-wavenumber grid points [cm-1]
rjaccon	rjaccon(imxpro*imxmw,imxcop) = jacobian matrix for the derivative of the continuum base-level values with respect to the continuum parameters
nucl	nucl = number of limb geometries to be skipped before starting continuum fit; numbering starts from top.
rpbase	rpbase(imxpro)= pressure at the base levels

Module structure and detailed description

The module proceeds along the steps identified by the following bullets:

- All the '*base*' input profiles are saved into '*old*' vectors and matrices:
  ibaseold = ibase
  begin loop I on '*base*' levels *j=1, ..., ibaseold*
  rvmrbaseold(j) = rvmrbase(j,1)

begin loop II on MW's: *k=1, ..., nselmw*
rcbaseold(j,k) = rcbase(j,k)
end loop II on MW's
end loop I on *'base'* levels

- Now the indexes of the '*base*' profiles that correspond to altitudes where the VMR profile is fitted are identified:
  k = 1
  begin loop on the *'base'* levels: *j=1, ..., ibase*
  if *lparbase(j)* = TRUE then: *imodif(k)=j, k=k+1*
  end loop on the *'base'* levels

- At this point *k-1* should be equal to *ipar*. If these two quantities are different a fatal error is produced and the program is stopped.

- calculates now the scaling factors for the VMR profile for the regions above the highest fitted point *'rtscalabove'* and below the lowest fitted point *'rtscalbelow'*:
  rtscalabove = rxpar(1)/rtbaseold(imodif(1))
  rtscalbelow = rxpar(ipar)/rtbaseold(imodif(ipar))

- Updates now the VMR profile, this is done in 3 steps. The obtained profile is recorded in the vector *rvmrbase*
  Step 1: region above the highest fitted point, the profile is scaled
  begin loop on levels: *k=1, ..., imodif(1)-1*
  *rvmrbase(k,1) = rvmrbaseold(k) * rtscalabove*
  end loop on levels *k*
  Step 2: region between first and last fitted points (linear interpolation used).
  begin loop on parameter levels: *j=1, ..., ipar-1*
  begin loop on levels where the VMR is changed by the current parameter: *k=imodif(j),..,imodif(j+1)*
  r3=rxpar(j+1)-rxpar(j)
  r4=rzbase(imodif(j+1))-rzbase(imodif(j))
  r5 = rzbase(k)-rzbase(imodif(j))
  *rvmrbase(k,1) = rxpar(j) + ((r3/r4)*r5)*
  end loop on levels *k* where the VMR is changed by the current parameter
  end loop on parameter levels *j*.
  Step 3: region below lowest fitted point, the profile is scaled
  begin loop on levels: *k=imodif(ipar)+1,..., ibase*
  *rvmrbase(k,1) = rvmrbaseold(k) * rtscalbelow*
  end loop on levels *k.*

- Updates the vector of the continuum parameters: *rcpar(j) = rxpar(ipar+j)* for *j=1, ..., icontpar*

- Updates continuum profiles and computes *rjaccon* by using **FICARRA_VMR** module:
  FICARRA_VMR*(nsam,dstep,ifspmw,rcbase,rpbase,ibase,nselmw,ilimb, rcpar,isaved,rjaccon)*

- Updates the vector *roffs* of the instrumental offset:

k = 0
begin loop on parameters: *j=2*ipar+icontpar+1, itop*
k = k + 1
roffs(k) = rxpar(j)
end loop on parmeters *j*

3.2.17 CONVCHK_VMR

Description
Checks whether the convergence has been reached or a further iteration (iterg) is required.

Variables exchanged with external modules

Name	Description
rchisq	rchisq(0:imxite) = total chi-square at each iteration
iterg	iterg = index of the current macro-iteration
rlinchisq	rlinchisq = value of the chi-square, in the linear approximation, relative to the current macro-iteration
rxpar	rxpar(imxtop) = vector of the fitted parameters at the current iteration
rxparold	rxparold(imxtop) = vector of the fitted parameters in the previous iteration
ipar	n. of fitted points in the T profile
itop	itop = total number of retrieved parameters
iobs	iobs = total number of fitted spectral data points
rlambda	rlambda = Marquardt's damping factor
rthres1	rthres1 = threshold n.1 used to check convergence criteria
rthres2	rthres2 = threshold n.2 used to check convergence criteria
rthres3	rthres3 = threshold n.3 used to check convergence criteria
lconverg	lconverg = logical variable which is TRUE only if the convergence has been reached

Detailed description

- Initialisation of *lconverg*:
  lconverg = .FALSE.

- Check that *iterg* > 0, otherwise stops the program. This is only a consistency check, i.e. the convergence has to be checked only after the initial iteration.

```
if (iterg.lt.1) then
  write(*,'(a)')'FATAL ERROR in CONVCHK: '
  write(*,'(a)')'Subroutine CONVCHK has been called with iterg < 1.'
  write(*,'(a)')'--------- PROGRAM STOPPED ----------'
  stop
end if
```

- Evaluation of the first convergence criterion, i.e. variation of the chi-square. The result is stored in the logical variable *lcrit1*:
  rchivar = abs*((rchisq(iterg)-rlinchisq)/rchisq(iterg))*
  lcrit1 = rchivar.le.rthres1

- Evaluation of the second convergence criterion, i.e. max. relative variation of tangent VMR parameters. The result is stored in the logical variable *lcrit2*:
  rmaxvarpar = 0.
  do 100 *j=1, ipar*
  if (*rxparold(j)*.ne.0) then
  rvarpar = abs(*(rxparold(j) - rxpar(j))/rxparold(j)*)
  end if
  if (*rvarpar*.gt.*rmaxvarpar*) *rmaxvarpar = rvarpar*
  100 end do
  lcrit2 = rvarpar.le.*rthres2*

- The final result of the convergence checks in then evaluated:
  lconverg = lcrit1 .or. *lcrit2*

3.2.18 AINVCAL_VMR

For the description of this module, see section 2.2.18.

3.2.19 OUTPUT_VMR

Description: Routine which generates the output files described in [AD7]. Source code of this module is listed in [AD7]. Please note that this subroutine uses (a call to) the subroutine cont_char_vmr.f in order to write-out the qualifiers characterising continuum retrieved parameters. The subroutine cont_char_vmr.f in described in Sect. 3.2.34.

Variables exchanged with external modules

Name	Description
rxpar	See description in section 3.3
ipar	See description in section 3.3
icontpar	See description in section 3.3
rainv	See description in section 3.3
nsam	See description in section 3.3
robs	See description in section 3.3
rspfov	See description in section 3.3
rchisq	See description in section 3.3
iobs	See description in section 3.3
itop	See description in section 3.3
iterg	See description in section 3.3
iterm	See description in section 3.3
rlambda	See description in section 3.3

rlinchisq	See description in section 3.3
ilimb	See description in section 3.3
igeo	See description in section 3.3
nselmw	See description in section 3.3
rchisqp	See description in section 3.3
slab	See description in section 2.3
lokku	See description in section 3.3
linloop	Swich which allows (if true) to save only information concerning each iteration and not information concerning the entire retrieval. (See the source code in appendix)
lcfit	lcfit(imxgeo,imxmw)= logical matrix identifying altitudes/MWs for which a continuum parameter is retrieved.
lccmat	lccmat(imxgeo,imxmw)= logical matrix identifying altitudes/MWs where the continuum is assumed to be equal to its value at a neighbouring MW,
nucl	sweep above which the continuum is not fitted
rvcol	rvcol(imxlmb) = vertical columns of the main gas of the retrieval
rconc	rconc(imxlmb) = concentrations of the main gas at the tangent altitudes
rvcmcol	rvcmcol(imxlmb,imxlmb) = VCM of the main gas columns
rvcmconc	rvcmconc(imxlmb,imxlmb)=VCM of the main gas concentrations

3.2.20 LININT_VMR

For the description of this module, see section 2.2.20.

3.2.21 GRAVITY

For the description of this module, see section 2.2.21. This module does not require the inclusion of the ‘parameters.inc’ file, therefore, the same module can be used for both p,T and VMR retrievals.

3.2.22 ESPINT_VMR

For the description of this module, see section 2.2.22.

3.2.23 LOGINT_VMR

For the description of this module, see section 2.2.23.

3.2.24 PTFROMZ_VMR

For the description of this module, see section 2.2.24.

3.2.25 CONV_VMR

For the description of this module, see section 2.2.25.

3.2.26 MWCONT_VMR

For the description of this module, see section 2.2.26.

3.2.27 FICARRA_VMR

For the description of this module, see section 2.2.27.

3.2.28 CONCANDCOL

```
CONCANDCOL
   |(----PARTCOL >
   |((((+LININT_VMR *
   |((((+PARTCOL >
```

Description: This subroutine calculates the vertical columns the concentrations and related variance - covariance data of the gas whose VMR profile has been retrieved.

Variables exchanged with external modules

Name	Description
rzmod	rzmod(imxlev): heights of levels used for the radiat. tranf. calc.
rtmod	rtmod(imxlev): temperature on levels used for the radiat. transf. calc.
rpmod	rpmod(imxlev): pressure on levels used for the radiat. transf. calc.
rxmod	rxmod(imxlev): VMR of the main gas of the actual retrieval on levels used for the radiat. transf. calc.
itglev	Vector that associates to each geometry, the corresponding number of the tangent level.
igeo	Total number of simulated geometries
lfitgeo	logical vector that identify the levels where the profiles are fitted: referred to rzsi (to the simulated geometries)
rzbase	altitude of the base-levels
rxbase	VMR of the main gas on the base-levels
ibase	number of base-levels
lparbase	lparbase(imxpro) = logical vector which identifies the altitudes where the T profile is fitted, among the altitudes rzbase.
rainv	matrix inverse of *ra*
rvcmcol	rvcmcol(imxlmb,imxlmb):
rvcmconc	rvcmconc(imxlmb,imxlmb):
rvcol	rvcol(imxlmb): total vertical column for the different limb views
rconc	rconc(imxlmb): concentration of the main gas at the different tangent altitudes

Module structure

1. Calculation of all the partial columns within the ‘mod’ levels
2. Calculation of the total vertical columns for the different limbs and calculation of the concentrations at the tangent altitudes
3. Computation of the derivatives of the column with respect to the VMR at the fitted points.
4. Calculation of the VCM associated to both the concentrations and the total vertical column.

Detailed description:

1. Calculation of all the partial columns within the ‘mod’ levels

For all the layers (*lay = 1* $\rightarrow$ *itglev(igeo) - 1*) the vertical column relative to a single layer is calculated by the routine **partcol**.
partcol (*rzmod(lay+1), rzmod(lay), rtmod(lay+1), rtmod(lay), rpmod(lay+1), rpmod(lay), rxmod(lay+1), rxmod(lay), rpvcol(lay)*)

2. Calculation of the total vertical columns for the different limbs and calculation of the concentration at the tangent altitudes

For each limb geometry corresponding to an observation the total vertical column is obtained by summing the partial vertical columns computed in 1. The result is stored in the vector *rvcol(imxlmb)*.
Besides, the concentration of the main gas corresponding to each tangent altitude is calculated by multiplying the VMR of the gas at that altitude by the relative numerical density.

The index *jj* is set to 0
Begin loop on all the simulated geometries: *j=1, ..., igeo*
Begin condition: the geometry *j* corresponds to an observation
(i.e. if *lfitgeo(j)* is *true*)

- the index *jj* is incremented of one unit
- $rvcol(jj) = \sum_{k=1}^{itglev(j)-1} rpvcol(k)$
- the molecule numerical density at the *jj*-th tangent altitude:

$$rden(jj) = \frac{rpmod(itglev(j))}{rtmod(itglev(j))} \cdot rk \cdot 10^{-6} ,$$

and the main gas concentration:

$$rconc(jj) = rden(jj) \cdot rxmod(itglev(j))$$

are calculated.

End condition
End loop

3. Computation of the derivatives of the columns with respect to the VMR at the fitted points.

The derivatives are computed numerically, so perturbed VMR profiles have to be generated first:

- The modifying factor *dx* for the VMR 'base' profile is set up:
  dx=1.01
- Then the indexes of the 'base' profile that correspond to altitudes where the profiles are fitted are identified and stored in the vector *amodif(imxlmb)*:
  k=0
  Begin loop on the levels of 'base' profiles: *j*
  Begin condition: if level *j* corresponds to a tangent altitude
  (if *lparbase(j)* is *true*)
  imodif(k)=j
  k=k+1
  End condition
  End loop
- The perturbed VMR profiles are built:
  Firstly we copy in the columns of the matrix *rxbasepert(imxpro, imxlmb)* the unperturbed VMR profile of the main gas of the retrieval.
  The perturbed profiles are obtained as follows:

a) Perturbed VMR profile corresponding to the first (uppermost) parameter:
Begin loop on 'base' levels: *k=1, ...,imodif(1)*
*rxbasepert(k,1) = rxbase(k) * dx*
End loop on 'base' levels.
b) Intermediate VMR perturbed profiles:
Begin loop on parameters: *j=2, ...,ipar-1*
*rxbasepert(imodif(j),j) = rxbase(imodif(j) * dx*
End loop on parameters
c) Perturbed VMR profile corresponding to the last (lowest) parameter:
Begin loop on 'base' levels: *k= imodif(ipar), ..., ibase*
*rxbasepert(k,ipar) = rxbase(k) * dx*
End loop on 'base' levels.

- The so obtained perturbed VMR profiles in the 'base' grid are interpolated to the 'rzmod' grid by using **LININT_VMR** module:
Begin loop I on the parameters: *k=1, ..., ipar*
Begin loop II on the 'mod' levels: *j=1, ..., itglev(igeo)*
LININT_VMR(*rzbase,rxbasepert(1,k),ibase,rzmod(j),*
rxmodpert(j,k))
End loop II on the 'mod' levels.
End loop I on the parameters.

- The perturbed partial vertical columns are calculated by using **PARTCOL** module:
Begin loop I on the parameters: *kk=1, ..., ipar* (ends at next bullet)
Begin loop II on the 'mod' levels: *k=1, ..., itglev(igeo)-1*
PARTCOL(*rzmod(k+1),rzmod(k),rtmod(k+1),rtmod(k),rpmod(k+1),*
rpmod(k),rxmodpert(k+1,kk),rxmodpert(k,kk),
rpvcolpert(k,kk))
End loop II on the 'mod' levels.

- The perturbed total vertical columns are calculated:
jj = 0
Begin loop II on geometries: *j=1,..., igeo*
if *litgeo(j)* = TRUE then
jj = jj + 1
if *kk* $\leq$ *jj* then
rvcolpert(jj,kk) = 0.
Begin loop III on 'mod' levels: *k=1,..., itglev(j)-1*
rvcolpert(jj,kk)=rvcolpert(jj,kk)+rpvcolpert(k,kk)
End loop III on 'mod' levels
else
rvcolpert(jj,kk)=rvcol(jj)
end if
end if
End loop II on geometries
End loop I on the parameters

- The jacobian matrix for the transformation from VMR to columns is computed:

Begin loop I on parameters: *k=1, ..., ipar*
Begin loop II on parameters: *j=1, ..., ipar*
rjcol(j,k) = (rvcolpert(j,k)-rvcol(j))/
(rxbasepert(imodif(k),k)-rxbase(imodif(k)))
End loop II on parameters
End loop I on parameters

4. Calculation of the VCMs related to vertical columns and concentrations.
The Variance - Covariance matrix *rvcmcol(imxlmb,imxlmb)* of the vector of the vertical columns is computed by performing the following matrix operation:

$$rvcmcol = rjcol * rainv * (rjcol)^T$$

where ^{T} indicates the transpose of matrix.
The Variance - Covariance matrix *rvcmconc(imxlmb,imxlmb)* of the vector of the concentrations is then computed as:
Begin loop I on parameters: *k=1, ..., ipar*
Begin loop II on parameters: *j=1, ..., ipar*
*rvcmconc(j,k) = rainv(j,k) * rden(j) * rden(k)*
End loop II on parameters
End loop I on parameters

3.2.28.1 PARTCOL

Description: This subroutine calculates the vertical column relative to a single layer

Variables exchanged with external modules:

Name:	Description:
rz0	height of the lower boundary of the layer
rz1	height of the higher boundary of the layer
rt0	temperature on the lower boundary of the layer
rt1	temperature on the higher boundary of the layer
rp0	pressure on the lower boundary of the layer
rp1	pressure on the higher boundary of the layer
rx0	VMR of the main gas on the lower boundary of the layer
rx1	VMR of the main gas on the higher boundary of the layer
rpcol	vertical column relative to the considered layer

Module structure:
Begin condition 1: the temperature gradient is different from 0
1. Numerical calculation of the partial column
Else condition 1: the temperature gradient is equal 0
2. Analytical calculation of the partial column
End condition 1

Condition 1.
if $|rt1 - rt0| > 10^{-8}$, i.e. if the temperature gradient is significantly different from 0, the calculation of the partial column is performed numerically, otherwise the calculation is performed analytically.

1. Numerical calculation of the partial column

The variable *deps* is set to 10^{-3}.
The numerical integral of the column is performed by the routine **qsimp1**, together with its sub-module **trapz1.**
qsimp1(*rz0, rz1, rt0, rt1, rp0, rp1, rx0, rx1, deps, rpcol*).

The column (in number of molecules per square centimetre) of the actual path is finally calculated by multiplying *rpcol* by a constant:

$$rpcol = rpcol \cdot rk \cdot 10^{-6},$$

where *rk* is a parameter contained in the file 'parameters_vmr.inc' and the factor 10^{-6} is due to the fact that the VMRs are read from **input** in parts per million (ppm).

2. Analytical calculation of the partial column

After the calculation of the following preliminary quantities:

$$rgx = \frac{rx1 - rx0}{rz1 - rz0};$$

$$rc = \frac{-rt0 \cdot \log(rp1/rp0)}{rz1 - rz0};$$

$$rex0 = \exp\left[-\frac{rc \cdot (rz1 - rz0)}{rt0}\right];$$

$$r0 = rx0 + \frac{rt0 \cdot rgx}{rc},$$

rpcol is then computed by using the formula:

$$rpcol = rk \cdot 10^{-6} \cdot \frac{rp0}{rc} \cdot \left[r0 - rex0 \cdot (r0 + rx1 - rx0)\right].$$

3.2.28.2 QSIMP1 & TRAPZ1

Description:
Starting from:

- the value of temperature, pressure and VMR of the gas at the boundaries of the layer, which are the limits of integration
- the interpolation law in altitude of all these quantities inside the layer,
  these two modules calculate the numerical integral *rpcol*.

Variables exchanged with external modules

Name:	Description:
rz0	altitude of the lower boundary of the layer
rz1	altitude of the higher boundary of the layer
rt0	temperature corresponding to the lower boundary of the layer
rt1	temperature corresponding to the higher boundary of the layer
rp0	pressure corresponding to the lower boundary of the layer
rp1	pressure corresponding to the higher boundary of the layer
rx0	VMR corresponding to the lower boundary of the layer
rx1	VMR corresponding to the higher boundary of the layer
deps	required accuracy for the integrals calculation
rpcol	returned column of this path (to be moved to the chosen measurement units)

Module structure
See 'Numerical Recipes in FORTRAN' [RD2] pag. 130-133.

Detailed description
The structure of this module is exactly the same of the one reported on 'Numerical Recipes in FORTRAN', pag. 130-133.
In particular, the integral is computed numerically, by using Simpson rule: in the implemented method, the trapezoidal rule is refined until a specified degree of accuracy (*deps)* has been achieved.

The integral calculated by **qsimp1** and **trapz1** modules is the following:

$$rpcol = \int_{rz0}^{rz1} \mathrm{X}_{gas}(z) \cdot \frac{p(z)}{T(z)} \cdot dz$$

$\mathrm{X}_{gas}(z)$, $p(z)$, $T(z)$ represent respectively the gas VMR, pressure and temperature behaviour as a function of the altitude, which is the integration variable.

The values of pressure, temperature and VMR, at a particular height *z* is computed by the module **ptxfromz (***z, rz0, rt0, rp0, rx0, rz1, rt1, rp1, rx1,* rti, rpi, rxi**)**.

3.2.28.3 PTXFROMZ

Description
Starting from the value of pressure, temperature, VMR at the boundaries of a given layer, this module calculates the value of pressure, temperature, VMR for a given altitude *z* inside the layer.

Variables exchanged with external modules:

Name	Description
z	altitude, referred to the surface of the earth, where the values of pressure, temperature, refractive index and VMR are required.
rz0	altitude of the lower boundary of the layer
rt0	temperature in correspondence of *rz0*
rp0	pressure in correspondence of *rz0*
rx0	VMR in correspondence of *rz0*
rz1	altitude of the higher boundary of the layer
rt1	temperature in correspondence of *rz1*
rp1	pressure in correspondence of *rz1*
rx1	VMR in correspondence of *rz1*
rti	returned temperature at *z*
rpi	returned pressure at *z*
rxi	returned VMR at *z*

Module structure:
1. Calculation of the temperature *rti* at the altitude *z* using a linear interpolation.
2. Calculation of pressure *rpi* at the altitude *z* using exponential interpolation.
3. Calculation of VMR *rxi* at the altitude *z* using a linear interpolation.

Detailed Description:
1. Inside the layer, the temperature *rti* is linearly interpolated in altitude, known the values of temperature (*rt0* and *rt1*) and the heights of the levels which mark the boundaries of the layer (*rz0* and *rz1*).
2. The pressure *rpi* at altitude *z* is obtained performing an exponential interpolation of the values of pressure on the boundaries of the layer (*rp0* and *rp1*).
3. The value of VMR at *rzi* is obtained by performing a linear interpolation of the values of VMR at the boundaries of the layer (*rx0* and *rx1*).

3.2.29 LINP_VMR

Description: Module used to calculate linear interpolations in pressure domain

Variables exchanged with external modules:

Variable	Description:
rx	rx(imxpro) = vector of 'ipro' elements containing the values to which 'ry(imxpro)' profile is referred.
ry	ry(imxpro) = vector of 'ipro' elements containing the profile used for the interpolation
ipro	ipro = number of elements in rx(i) and ry(i) profiles
rx1	rx1 = value of rx where the value of the profile is required.
ry1	ry1 = value of the profile corresponding to rx1

Algorithm Description

We have a vector *ry(imxpro)*containing a general profile, the elements of this vector are referred to the grid recorded in the vector *rx(imxpro).* The problem is to find the value of the profile corresponding to the altitude *rx1* assuming a linear behaviour of the profile within the points represented in *ry(imxpro).* For optimisation purposes, the vector *rx* is supposed as sorted with small values for small values of the index (i.e stating from high altitudes if *rx* is a pressure profile).

Detailed description

The calculation proceeds in the following two steps:

- Search for the index *j* so that: *rx(j+1) < rx1 < rx(j);*
- if an index *j* is found for which *rx1 = rx(j)* then we set *ry1 = ry(j)* and we exit from the subroutine.
- if none of the two previous conditions can be satisfied, a fatal error is produced (this can happen only if the profiles are not ordered starting from high altitudes or the requested pressure *rx1* does not belong to the range covered by the vector *rx*).
- Linear interpolation is then performed:

$$ry1 = ry(j)+((ry(j+1)-ry(j))/(rx(j+1)-rx(j)))*(rx1-rx(j)).$$

3.2.29 ADDOFF_VMR

For the description of this module, see section 2.2.28.

3.2.30 READ_IRRGRID_VMR

For the description of this module, refer to section 2.2.30.

3.2.31 READ_LOOKUP_VMR

For the description of this module, refer to section 2.2.31.

3.2.32 DECOMPR_VMR

For the description of this module, refer to section 2.2.32.

3.2.33 CONT_CHAR_VMR

Description

This subroutine evaluates the qualifiers that characterise continuum retrieved parameters. The routine is called by the output_vmr module. In that occasion the qualifiers are calculated and directly written into the main output file of the current VMR retrieval (xxxx_out.dat).

Variables exchanged with external modules

Name	Description

rxpar	rxpar(imxtop) = vector of the fitted parameters
rainv	rainv(imxtop,imxtop)= VCM of the retrieved parameters
ilimb	ilimb = number of measured geometries
ipar	ipar = number of parameter-levels (i.e. N. of elements of rzpar vector)
nselmw	nselmw = total number of selected microwindows for the retrieval
nucl	nucl = number of limb geometries to be skipped before starting continuum fit; numbering starts from top.
lokku	lokku(imxgeo,imxmw) = occupation matrix used for the selection of operational MW's for each observation geometry
lcfit	lcfit(imxgeo,imxmw) = continuum occupation matrix
lccmat	lccmat(imxgeo,imxmw) = logical matrix which identifies altitudes & MWs where the continuum is set equal to the continuum of a nearby MW (close-close MWs).

Detailed description

```
* This module contains the algorithm for deriving the quantities to be reported in Level 2 products
* for characterisation of continuum fitted parameters, starting from the ORM variables.
*
* The matrix lccmat(imxgeo,imxmw) identifies among the MWs/altitudes of the
* occupation matrix 'lokku',
* the MWs/altitudes which are tightly grouped with the next (leftwards) MW/altitude where
* continuum is fitted. This matrix is computed in the modules 'mwcont_pt(vmr)'
*

      subroutine cont_char_vmr(rxpar,rainv,ilimb,ipar,nselmw,nucl,
     &           lokku,lcfit,lccmat)

      implicit none
      include 'parameters_vmr.inc'
* Declaration of variables is omitted here.

      icpar = 0  ! initialisation of a counter

* Initialisation of computed variables:
      do i=1,ilimb
        do j=1,nselmw
          igroup_type(i,j) = 0          ! continuum parameter group_type
          xsect(i,j) = 0.d0             ! retrieved continuum cross-section at sweep i and MW j
          var(i,j) = 0.d0               ! variance of xsect(i,j)
          covx(i,j) = 0.d0              ! covariance between xsect(i,j) and retrieved VMR at sweep i
        end do
      end do
*
* Start of loop over sweeps where continuum is considered
* and loop over microwindows of the current retrieval
*
      do i=nucl+1,ilimb        ! loop on sweeps (altitudes)
        do j=1,nselmw          ! loop on microwindows
          if (lokku(i,j)) then   ! if the current mw 'j' is used at sweep 'i'
            if (lcfit(i,j)) then   ! if continuum is fitted at this sweep/mw
              icpar = icpar + 1 ! icpar counts continuum parameters
*
* We have to setup the group_type(i,j) for the current continuum parameter:
* 1 - this mw is isolated
* 2 - this mw is an edge of a loose group
* 3 - this mw is a leftmost  edge of a tight group
* 4 - this mw is a leftmost edge of a tight group AND an edge of a loose group
* 5 - this mw belongs to a tight group (but is not an edge of the group)
```

```
* 6 - this mw belongs to a loose group (but is not an edge of the group)
*
          loose = .FALSE.
          ltight = .FALSE.
* Look right:
          do k=j+1,nselmw
            if (.not.lcfit(i,k)) then
             if (.not.lccmat(i,k).and.lokku(i,k)) loose=.TRUE.
             if (lccmat(i,k).and.lokku(i,k)) ltight=.TRUE.
            else
             goto 12
            end if
          end do
12          continue
* Look left:
          do k=j-1,1,-1
            if (.not.lcfit(i,k)) then
             if (lokku(i,k).and.(.not.lccmat(i,k))) loose=.TRUE.
            else
             goto 13
            end if
          end do
13          continue
* Take a decision:
          if (.not.(loose.or.ltight)) igroup_type(i,j)=1
          if (loose.and.(.not.ltight)) igroup_type(i,j)=2
          if (ltight.and.(.not.loose)) igroup_type(i,j)=3
          if (ltight.and.loose) igroup_type(i,j)=4

********
* In VMR retrieval the following correspondences are valid:
          xsect(i,j) = rxpar(ipar+icpar)
* var(xsect(i,j)) =
          var(i,j) = rainv(ipar+icpar,ipar+icpar)
* cov(xsect(i,j),VMRx(i)) =
          covx(i,j) = rainv(i,ipar+icpar)
********


        else                ! if continuum is not fitted at this sweep/mw

          if (lccmat(i,j)) then
            igroup_type(i,j)=5   ! the mw belongs to a tight group
          else
            igroup_type(i,j)=6   ! the mw belongs to a loose group
          end if

        end if               ! end if cont. is fitted at this sweep/mw
      end if                 ! end if mw is used at sweep 'i'
    end do                   ! end loop on microwindows
  end do                     ! end loop on sweeps

* Writing into the main output file of the retrieval (xxxx_out.dat) :
    do j=1,nselmw
      do i=1,ilimb
        write(29,'(a5,i2,a9,i2)')'mw = ',j,',  lmb = ',i
        write(29,*)'group_type(lmb,mw) = ', igroup_type(i,j)
        write(29,*)'xsect(lmb,mw) = ',xsect(i,j)
        write(29,*)'var(xsect(lmb,mw)) = ', var(i,j)
```

```
    write(29,*)'cov(xsect(lmb,mw),vmr(lmb)) = ', covx(i,j)
   end do
  end do


  end
```

3.3 Variables and parameters used in the VMR retrieval program

The parameters used in VMR retrieval are described in the following table. These parameters are defined in the 'parameters_vmr.inc' file.

Name	Description	Value
dcdop	used in Doppler broadening: sqrt(2 ln2 k avog / c^2)	3.5811737d-7
dext	extension of the (already with iadd*delta extended) microwindow where ioutin is set to 1 [cm^{-1}]	0.4
dinvpi	1/pi	0.318309886
dsqln2	sqrt(ln2)	0.832554611
dsqpi	sqrt(pi)	1.772453851
dtineig	minimim permitted value for the eigenvalues of A	1.0d-40
iqlclf	the quotient between coarse and fine wavenumber grid intervalls	5
imxapo	maximum number of points of the apodisation function (path difference domain)	513
imxbv	max. number of base vectors of compressed look-up tables	10
imxcof	max number of coefficients for the calculation of thequotient of the partition sum (=4)	4
imxcop	max. number of continuum parameters	180
imxcta	max number of elements in the correction table of tangent altitudes due to refraction index	50
imxept	max number of extra paths	1
imxfcs	max number of frequencies to which cross sections are provided in the look-up tables	1
imxfpg	max number of elements in the fixed P grid imposed to the retrieval	50
imxgas	max number of gas in the retrieval	10
imxgeo	max number of simulated observations	18
imxgmw	max number of gases per MW	4
imxhit	number of gases in the HITRAN 96 data base	36
imxhol	max. number of holes between true elements in the columns of the occupation matrix	100
imxi	maximum number of sampling points in the synthetic spectra computed at the observed frequencies	100
imxsi2	max number of '1' points in the irregular grids of the considered microwindows	1100
imxilc	max number of sampling point in the instrument line-shape function (course grid!)	1000
imxils	maximum number of sampling points in the instrument line-shape function (fine-grid!)	2400
imxism	max number of isotopes in HITRAN data base per molecule (=8)	8
imxiso	number of total isotopes in the HITRAN database	85
imxite	maximum number of macro-iterations in retrieval procedure	15

imxj	maximum dimension of J matrix (VCM obs = $J \cdot J^T$)	imxilc+imxi
imxlay	max number of layers for modelling the atmosphere (=imxlev-1)	imxlev-1
imxlev	max number of levels used for modelling the atmosphere	70
imxlin	max number of lines per microwindow	300
imxlmb	max number of parameters to be retrieved for each set of parameters (p,T,C,vmr)	18
imxmw	max number of microwindows	20
imxnx	max. number of p and T points considered in cross-section look-up tables (nx=np*nt)	1000
imxobs	max number of observational point (for Jacobian matrix)	2700
imxpat	max number of possible paths (be careful: imxpat* imxsig*4*imxgas is the number of bytes needed for the biggest field (variable rcross) in the program!)	imxlay+imxept*(imxgeo-1)
imxpcs	max number of P to which cross sections are provided in the look-up tables	1
imxpre	maximum number of points for the precalculated line shape	imxsig
imxpro	max number of elements in p, t profiles	100
imxpun	max. dimension of a pointer in ficarra_pt	100
imxri	max number of refraction indices provided in the corresponding file	50
imxsav	max. dimension of the saved vector used dy mwcont_pt and ficarra_pt	3000
imxsig	max number of wavenumber grid-points for a microwindow	5500
imxsl	max number of sub-levels between the pointings of the simulations	20
imxsnc	max number of sampling point for the sinc function used to interpolate the instrument line-shape function	4800
imxtcs	max number of T to which cross sections are provided in the look-up tables	1
imxtop	max number of parameters to be fitted	60
imxvt	max number of vibrational T provided in the corresponding file	20
nrepiso	HITRAN isotope number of the gas for which the line shape is precalculated	1
nrepcode	HITRAN code for the gas for which the line shape is precalculated (HNO_3)	12
rairmass	average molec. weigth of the air (kg/kmol) (US STD)	28.9644
rbc	Boltzmann constant (for density in mol/cm-3)	1.380658e-19
rc1	constant in the Planck-function (2 h c^2)	1.191043934e-3
rcn	constant in the refraction index expression (n=1.+(rcn*rt0n/rp0n)*p/T)	.000272632
rdmult	the number of Doppler half-widths from the line-centre from which the Lorentz function instead of the Voigt-function is used Error: rdmult=10 -> 1.5% ; rdmult=20 -> 0.4% ; rdmult=30 -> 0.18%	30.

refind	multiplicative constant in the expression of refraction index n: refind= rcn*rt0n/rp0n	rt0n*rcn/ rp0n
repexph	reference half width exponent of the line to be precalculated	0.75
rephw0	reference half width of the line shape to be precalculated	0.11
rg0	acceleration of gravity (m/s**2)	9.80665
rhck	h*c/k [K/cm-1]	1.4387687
rk	10^{-5}/rbc	10^{-5}/rbc
rmovr	1000 * rairmass / R(=8314.32[N.m/(kmol.K)])	3.483676
rp0h	reference pressure for pressure broadening	1013.25
rp0n	pressure on level sea for refraction index calculation	1013.25
rt0h	reference temperature for pressure broadening	296.
rt0int	reference temperature for the line intensity	296.
rt0n	temperature on level sea for refraction index calculation	288.16
rvlf	multiplier for (Doppler+Lorentz=~Voigt) half-width to determine the local fine grid	0.1
rvmult	rvmult is the number of (Doppler+Lorentz=~Voigt) half-widths from the line-centre where the transition between local coarse and local fine grid occurs (rvmult >= rdmult !!	50

The variables used in VMR retrieval and exchanged by modules are listed in the table below

Name:	**Dim-ension:**	**Description:**	**Modified in:**
cint	imxmw	character*3: it indicates, for each microwindow, what kind of interpolation has to be performed between the spectral points of the irregular grid.	read_irrgrid_ vmr
delta		distance between fine-wavenumber grid points [cm^{-1}]	input_vmr
deps		maximum relative variation for each iteration in calculation of curtis-godson variables	input_vmr
dsigm0		central frequency of the line used for testing P levels [cm^{-1}]	input_vmr
dsigma	imxsig, imxmw	wavenumber fine grid for each microwindow [cm^{-1}]	grid_vmr
dsilin	imxlin, imxmw	central wavenumber for each line of each Mw [cm^{-1}]	input_vmr
dstep		distance between coarse-wavenumber grid points [cm-1]	input_vmr
iadd		number of fine-wavenumber grid points to be added on both sides of each microwindow (due to the ils-convolution)	ails_vmr
ibase		number of base-levels	chbase_vmr
icode	imxlin, imxmw	HITRAN molecular code for each line of each Mw	input_vmr
icontpar		total number of continuum parameters to be fitted	guesspar_vmr
iderlay	imxlmb ,3	highest (x,1), lowest (x,3) and middle (x,2) (the one directly above the perturbed layer) which is affected by each derivative (imxlmb refers to the parameter-levels)	mkplev_vmr
iept		actual number of extra paths	input_vmr
ifspmw	imxmw	index of the first sampling point of each MW * NOTE: the	input_vmr

		sampling point at frequency=0 has index=1	
igas		number of different gases for actual retrieval	inigas_vmr
igashi	imxgas	HITRAN code number for each global gas number	inigas_vmr
igasmw	imxmw	number of gases to be considered for each mw	inigas_vmr
igasnr	imxgas, imxmw	global gas number for the local gas number of each Mw	inigas_vmr
igeo		number of simulated geometries	occusim_vmr
igeocder	imxgeo, 2	for each simulated geometry *j* the highest (*igeocder*(*j*,1)) and the lowest (*igeocder*(*j*,2)) parameter level which has to be considered for the continuum-derivatives	tcgeo_vmr
iigrid	imxsig, imxgeo, imxmw	irregular grid in the '0' and '1' representation for all the fine grid points of the extended microwindow *imw*	read_irrgrid_ vmr
igridc	imxsi2, imxmw	matrix which associates to each microwindow and each point of the compressed grid, the corresponding index on the regular fine grid.	read_irrgrid_ vmr
iiso	imxlin, imxmw	isotope number for each line of each Mw.	input_vmr
ilev		number of levels for simulations	mkplev_vmr
ilim	2, imxi, imxmw	variable used for making the direct interpolation/convolution : *ilim(1,jsam,imw):* first point of the compressed grid to be considered for the computation of the low resolution spectral point at *jsam* for microwindow *imw*; *ilim(2,jsam,imw):* total number of points of the compressed grid to be considered for the computation of the low resolution spectral point at *jsam* for microwindow *imw*.	read_irrgrid_ vmr
ilimb		number of measured geometries	input_vmr
ilimbmw	imxmw	number of valid measured geometries per microwindow (number of 2 in each column of iocsim)	occusim_vmr
iline	imxmw	number of lines in each microwindow	input_vmr
ilookup mw	imxmw	ilookupmw(imw)=0 no look-up tables for mw imw ilookupmw(imw)=1 look-up tables for all the absorbers of the mw ilookupmw(imw)=2 look-up tables for not all the absorbers of the mw	input_vmr
imaingas		HITRAN code of the main gas of the retrieval (=2 for CO_2 in the case of p-T-retrieval)	input_vmr
imw		number of the actual microwindow	fwdmdl_vmr
iobs		total number of observations to be fitted	occusim_vmr
iocsim	imxgeo, imxmw	occupation matrix for the simulations to performed = 0 no simulation required, = 1 simulation required without FOV = 2 simulation required with FOV	occusim_vmr
ioutin	imxlin, imxmw	flag for each line =1: line-shape has to be calculated at each wavenumber inside the microwindow =2: line is considered as nearby continuum	input_vmr

ipar		number of parameter-levels	occusim_vmr
ipath		number of different IAPT numbers in ipoint	point_vmr
ipoint	imxlay, imxgeo	matrix, which attaches to each pair of layer/geometry the IAPT number	point_vmr
ipro		number of elements contained in P, T and VMR profiles initial guess	input_vmr
irowmw	imxmw	the row of the Jacobian matrix where the actual microwindow starts	occusim_vmr
isigma	imxmw	number of general wavenumber fine grid points in each microwindow	grid_vmr
iterg		macro - iteration index (Gauss)	retr_vmr
iterm		micro - iteration index (Marquardt)	retr_vmr
itglev	imxgeo	number of the tangent-level for each geometry	mkplev_vmr
itop		total number of parameters to be fitted	guesspar_vmr
lccmat	imxgeo, imxmw	This matrix identifies among the MWs/altitudes of the occupation matrix 'lokku', the MWs & altitudes which are tightly grouped with the next (leftwards) MW/altitude where continuum is fitted.	mwcont_vmr
lconverg		logical variable which is true if convergence is reached	convchk_vmr
lfit	imxlmb	logical vector that identify the levels where the profiles are fitted: referred to rztang (to the mearsured geometries)	input_vmr
lfitgeo	imxgeo	logical vector that identify the levels where the profiles are fitted: referred to rzsi (to the simulated geometries)	occusim_vmr
lirrgridmw	imxmw	logical vector that, for each selected microwindow in the actual retrieval, indicates whether the irregular grid is available.	read_irrgrid_vmr
lmgas	imxgmw,imxmw	lmgas(mgas,imw)=.true. : calculation of cross-sections without look-up tables lmgas(mgas,imw)=.false. :calculation of cross-sections by means of look-up tables	input_vmr read_lookup_vmr
lokku	imxgeo, imxmw	occupation matrix used for the selection of operational MW's for each observation geometry	input_vmr
lparbase	imxpro	logical vector that identify the levels where the profiles are fitted: referred to rzbase (to the base-levels)	chbase_vmr
nailsdp		number of AILS data points	input_vmr
napod	imxapo	number of points of the apodisation function in the interferogram domain (rapod)	input_vmr
nils		number of elements of rils	ails_vmr
ninterpol		switch for the decision of interpolation of the absorption cross-sections for the geometries above the lowest geometry (only if the IAPT number of the path is increasing, which was decided during the calculation of ipoint) =-1: no interpolation, all cross-sections recalculated =0: all cross-sections above the lowest geometry are interpolated =1: new calculation only of the tangent-layer, all other layers interpolated	input_vmr

		=2: new calculation of the tangent-layer and the layer above, all others interpolated =3: ...	
nll	imxgw, imxmw	number of basis vectors in cross-section look-up tables	read_lookup_vmr
npl	imxgmw, imxmw	number of -log(pressure) tabulation points in cross-section look-up tables	read_lookup_vmr
nrd		Ratio between general coarse grid step and general fine grid step	input_vmr
nsam	imxmw	number of sampling points in each MW (general coarse grid)	input_vmr
nselmw		total number of selected microwindows for the retrieval	input_vmr
ntl	imxgmw, imxmw	number of temperature tabulation points in cross-section look-up tables	read_lookup_vmr
nucl		nucl+1 = upper parameter level for continuum fit	retr_vmr
nused1	imxmw	total number of points of the compressed grid for each microwindow	read_irrgrid_vmr
ra	imxtop, imxtop	matrix defined as (transpose of rjacob) * rvcmobinv * rjacob	abcalc_vmr, amodif_vmr
rails	imxilc, imxmw	apodised instrument line shape for all selected MWs	input_vmr
rainv	imxtop, imxtop	matrix inverse of ra	ainvcal_vmr
raircol	imxlay, imxgeo	air-column for each layer and each geometry [moec/cm^{-2}]	curgod_vmr
rapod **real*4**	imxapo	apodisation function in interferogram domain	input_vmr
rapod_sigma **real*4**	imxilc	apodisation function in spectral domain	sinvcal_vmr
rb **real*4**	imxtop, imxobs	matrix defined as (transpose of rjacob) * rvcmobinv	abcalc_vmr
rbase		greater base of trapezium of Field of View function [km]	input_vmr
rcbase	imxpro, imxmw	continuum on the base-levels for each MW [cm^2/molec]	chbase_vmr
rcderfov	imxi, imxgeo, imxlmb	derivate with respect to continuum after fov convolution [r.u./(cm^2/molec)]	fov_vmr
rchisq	0:imxite	total chi square in the different iterations	difchi_vmr
rchisqp	imxlmb,imxm	chi-square for each observation geometry and each microwindow temperature profiles	difchi_vmr

	w		
rclay	imxlay, imxmw	model-layer values of the continuum [cm^2/molec]	conlay_vmr
rcol	imxlay, imxgeo, imxgas	column amounts for each layer, each geometry and each gas [molec/cm^2]	curgod_vmr
rcolpert	imxlay, imxgeo, 2	columns of the main gas for the perturbed temperature profiles [molec/cm^2]	curgod_vmr
rconc	imxlmb	Concentration profile of the retrieved gas	concandcol
rconint	imxlmb ,imxmw	frequency range around each MW,in which the continuum can be considered as varying linearly. [cm^{-1}]	input_vmr
rconvc	3	thresholds used to check convergence criteria (see convchk-description)	input_vmr
rcprof	imxpro, imxmw	array containing continuum cross section as a function of altitude and microwindow [cm^2/molec]	input_vmr
rcross **real*4**	imxsi2, imxpat, imxgmw	absorption cross sections for each irregular grid point (1^{st} index), each IAPT number (2^{nd} index) and each gas (3^{rd} index) for the actual Mw [cm^2/molec]	cross_vmr
rdpl **real*4**	imxgw, imxmw	spacing of -log(pressure) tabulation in cross-section look-up table	read_lookup_vmr
rdtl **real*4**	imxgw, imxmw	spacing of temperature tabulation in cross-section look-up table	read_lookup_vmr
rearad		local radius of curvature of the earth [km]	input_vmr
redfact		reduction factor applied to 'rincz' when it produces not acceptable P levels	input_vmr
relow	imxlin, imxmw	lower state energy for each line of each Mw [cm^{-1}]	input_vmr
rexph	imxlin, imxmw	exponent for temp. dependence of air-broadenedhalf width	input_vmr
rexphref		exponent for the calculation of Lorentz h-w in mkplev	input_vmr
rgderfov	imxi, imxgeo, imxlmb	derivate with respect to vmr after fov convolution	fov_vmr
rhw0	imxlin, imxmw	air broadened half width [cm^{-1}/atm] at 296 K	input_vmr
rhw0ref		half-width of the line used for testing P levels [cm^{-1}/atm] at 296 K	input_vmr
rhwvar		relative max. half-width variation allowed between two neighbouring P levels	input_vmr
rils	imxils, imxmw	instrument-line-shape function in the frequency fine grid	ails_vmr

rincz		trial increment given to altitude for building P levels	input_vmr
rint0	imxlin, imxmw	line intensity for each line of each Mw [cm^{-1}/(molec*cm^{-2}]	input_vmr
rintils		ratio between the frequency step approximating infinitesimal spectral resolution and the integral of the ILS function	ails_vmr
rjaccon	imxlmb * imxmw , imxtop	jacobian matrix for the derivatives of the continuum parameter-level values with respect to the continuum parameters	ficarra_vmr
rjacob **real*4**	imxobs, imxtop	Jacobian Matrix 1st index: observations 2nd index: parameters	jacsetmw_vmr
rkl **real*4**	imxbv, imxnx, imxgmw, imxmw	K-matrix	read_lookup_vmr
rlambda		Marquardt damping factor **(this value is now set up in retr_pt. In future versions it should be read by input.)**	retr_vmr
rlat		latitude of the actual limb-scan (deg.)	input_vmr
rlinchisq		χ^2 calculated in the linear approximation	newparest_vmr
rlolin	imxlin, imxmw	lower limit where the line has to be considered [km]	input_vmr
rmaxtv1		max. allowed temp. variation between levels,when: 0 < altitude of level < rzt12 [K]	input_vmr
rmaxtv2		max. allowed temp. vatiation between levels, when: rzt12 < altitude of level < rulatm [K]	input_vmr
rmrmod	imxlev, imxgas	volume mixing ratio for each gas considered in actual retrieval on levels used for rad. tra. calc.	mkplev_vmr
rnoise	imxmw ,imxgeo	NESR dependent on geometry and microwindow	input_vmr
rnres	imxobs	vector of the differences between the observed spectra and the calculated ones; first all the geometries of the first microwindow starting from the first geometry, then all the other microwindowsvector of the differences between the observed spectra and the calculated ones	difchi_vmr
robs	imxi, imxgeo, imxmw	observed spectra corresponding to the different tangent pressures and different microwindows (on the general wavenumber coarse grid) [r.u.]	input_vmr
roffs	imxmw	fitted instrumental offset for each mw [r.u.]	updprof_vmr
ropath	imxlay, imxgeo	optical path length for each layer, each geometry [km]	curgod_vmr
rp1l	imxg	lowest -log(pressure) value in cross-section look-up table	read_lookup_

real*4	w, imx mw		vmr
rpartcder	imxlay, imxlmb ,imxm w	partial derivatives of the continuum layer values with respect to the parameter-level values	conlay_vmr
rpartgder	imxlay, imxgeo, 2	partial derivative of the main gas column of each layer with respect to the vmr parameter level values	curgod_vmr
rpbase	imxpro	pressure on the base-levels [hPa]	chbase_vmr, updprof_vmr
rpeq	imxpat, imxgas	equivalent pressures [hPa]	curgod_vmr
rperc		maximum relative (with respect to rconint) distance between central frequencies of two microwindows which are defined as close-close ones for the definition of continuum emission	retr_vmr
rpmod	imxlev	pressure on levels used for the radiat. transf. calc. [hPa]	mkplev_vmr
rpprof	imxpro	vector of pressure profile as a function of altitude Z. [hPa]	input_vmr
rsan	imxi, imxsi2, 4, imxmw	variable used for making the direct interpolation/convolution of the spectra.	read_irrgrid_ vmr
rsl		half-difference between the bases of the trapezium (1/rsl gives the slope) [km]	input_vmr
rspct	imxi, imxgeo	spectrum for each geometry on the general coarse grid [r.u.] 1st index: general wavenumber coarse grid 2nd index: geometries to be simulated for the actual Mw	spectrum_vm r
rspctcder	imxi, imxgeo, imxlmb	continuum derivative spectra on the general coarse grid for each geometry and each parameter level [r.u./(cm^2/molec)] 1st index: general wavenumber coarse grid 2nd index: geometries to be simulated for the actual Mw 3rd index: levels where the parameters are retrieved	spectrum_vm r
rspctgder	imxi, imxgeo, imxlmb	vmr derivative spectra on the general coarse grid for each geometry and each parameter level 1st index: general wavenumber coarse grid 2nd index: geometries to be simulated for the actual Mw 3rd index: levels where the parameters are retrieved	spectrum_vm r
rspfov	imxi, imxgeo, imxmw	simulated spectra corresponding to the different tangent pressures and different microwindows on the general wavenumber coarse grid: (rspct * FOV) [r.u.]	fov_vmr, addoff_vmr
rt1l **real*4**	imxg mw, imx mw	lowest temperature value in cross-section look-up table	read_lookup_ vmr
rtbase	imxpro	temperature of the base levels [K]	chbase_vmr,

			updprof_vmr
rteq	imxpat, imxgas	equivalent temperatures [K]	curgod_vmr
rthres1		thresholds used to check convergence criteria (see convchk-description)	input_vmr
rthres2		thresholds used to check convergence criteria	input_vmr
rthres3		thresholds used to check convergence criteria	input_vmr
rtmain	imxpat	Curtis-Godson temperature of the main gas [K]	curgod_vmr
rtmod	imxlev	temperature on levels used for the radiat. transf. calc. [K]	mkplev_vmr
rtprof	imxpro	vector of temperature as a function of altitude Z. [km]	input_vmr
ru **real*4**	imxsi2, imxbv, imxgmw, imxmw	U-matrix	read_lookup_vmr
rulatm		upper limit of the atmosphere [km]	input_vmr
ruplin	imxlin, imxmw	upper limit where the line has to be considered [km]	input_vmr
rvcmcol	imxlmb, imxlmb	VC matrix ot the vertical columns of the retrieved gas	concandcol
rvcmconc	imxlmb, imxlmb	VC matrix ot the concentration profile of the retrieved gas	concandcol
rvcmobinv **real*4**	imxi, imxi, imxmw	blocks of the inverse of the variance-covariance matrix of the observations for each selected microwindow of the actual retrieval	sinvcal_mw_vmr
rvcmobinvopt **real*4**	imxi, imxi	optimised block of the inverse of the variance- covariance matrix of the observations	sinvcal_vmr
rvcol	imxlmb	Vertical column of the retrieved gas	concandcol
rvmrbase	imxpro, imxgas	volume mixing ratio of the gases on the base levels [ppm]	chbase_vmr, updprof_vmr
rvmrprof	imxpro, imxgas	matrix of VMR profiles [ppm]	input_vmr
rwmol	imxhit, imxism	molecular weight for each HITRAN molecular code and isotope number [g/mol]	wmol_vmr
rwmolref		molecular weight of the gas used for testing P levels [g/mol]	input_vmr
rxpar	imxtop	vector of the fitted parameters	guesspar_vmr ,newparest_vmr
rxparold	imxtop	vector of the fitted parameters at the previous iteration	newparest_vmr
rzbase	imxpro	altitude of the base-levels [km]	chbase_vmr, updprof_vmr
rzerof		zero-filling expressed as the ratio between measured and	input_vmr

		transformed interferogram	
rzmod	imxlev	heights of levels used for the radiat. tranf. calc. [km]	mkplev_vmr
rzmodpert	imxlev, imxlmb	perturbed altitude grids after the perturbation of temp. profiles. [km]	mkplev_vmr
rzpar	imxlmb	vector of the altitudes where the temperature profile is fitted [km]	updprof_vmr, guesspar_vmr
rzprof	imxpro	vector of altitudes Z to which rtprof, rpprof and rvmrprof are referred [km]	input_vmr
rzsi	imxgeo	tangent altitudes of the geometries to be simulated [km]	occusim_vmr, updprof_vmr
rzt12		altitude where the temperature threshold changes from rmaxtv1 to rmaxtv2 [km]	input_vmr
rztang	imxgeo	vector containing the engineering values of tangent altitudes [km]	input_vmr
smw	imxmw	character*6: vector containing the identifying label of the selected microwindows	input_vmr
tab	imxgmw, imxmwi	character*3: tabulation code of cross-section look-up tables character*6: vector containing the identifying label of the selected microwindows	read_lookup_vmr

4. Software architecture and algorithms of the OFM scientific code

In this section the software architecture and the algorithms used in the Optimised Forward Model (OFM) are specified. Section 4.1 shows the high level flow diagram of the calls between main modules and the detailed calling tree. The tree of calls of each module, its I/O data and the algorithms are described in section 4.2.
Section 4.3 contains a description of parameters and variables used by OFM modules.

4.1 High level flow diagram of calls

Below the calling tree and the structure of the starting module of the OFM program are described.

```
OFM ]
    |-----INPUT *
    |-----SAPOD *
    |-----OCCUSIM *
    |-----CHBASE *
    |-----FAILS *
    |-----GRID *
    |-----FWDMDL *
    |?----ADDNOISE *
    |-----OUTESA *
    |-----OUTOBSERV *
    |-----WRITECONT *
```

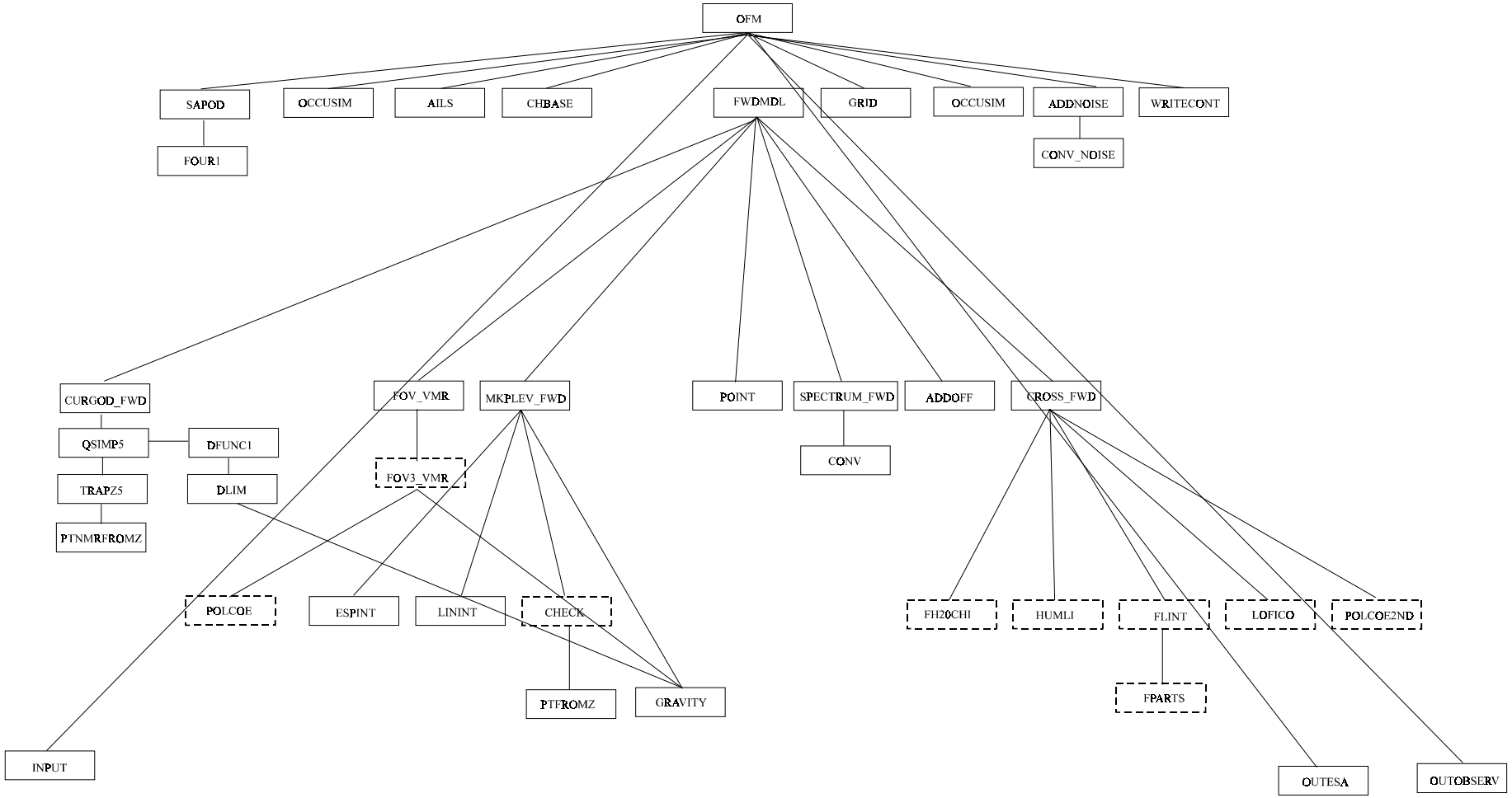


Fig. 10: Complete calling tree of the self-standing OFM program.

4.2 OFM modules architecture and algorithms

In this section the architecture and the algorithms of the forward model (self-standing) are described.
The description of each module is made using the rules showed in section 2.2.

4.2.1 INPUT

```
INPUT                                   :
   |-----SKIP *
   |-----READMW ]
   |       |-----BLIND *
   |-----UPLIMIT +
   |-----SPETFILL ]
   |-----INIGAS_FWD ]
   |       |(((--BLIND *
   |-----READVMR +
   |-----WMOL +
   |-----R_APOD_VMR ]
   |-----GASDEV *
   |-----GRAVITY *
```

For the description of this module, please refer to its source code reported in AD7.

4.2.1.1 READMW

Description: module used to read the files defining the MWs that have to be simulated.
For the description of this module, please refer to its source code reported in [AD7].

4.2.1.2 SPETFILL

Description: module used to read the spectroscopic database file.
For the description of this module, please refer to its source code reported in [AD7].

4.2.1.3 INIGAS_FWD

Description
Initialisation of the variables *'igas, igashi,igasmw,igasnr'* that define the two internal gas codes.

Variables exchanged with external modules

Name	Description
nselmw	total number of selected microwindows
iline	number of lines in each microwindow

icode	HITRAN molecular code for each line of each MW
imaingas	HITRAN code of the main gas of the retrieval (=2 in the case of p-T retrieval)
imaskcont	switch for the calculation of H_2O, O_2 and N_2 continuum: imaskcont(1)=1: H_2O continuum calculation imaskcont(2)=1: O_2 continuum calculation imaskcont(3)=1: N_2 continuum calculation
igas	number of different gases for actual retrieval
igashi	HITRAN code number for each global gas number
igasmw	number of gases to be considered for each mw
igasnr	global gas number for the local gas number of each Mw

Module structure:

Begin loop 1 over all microwindows
1. Adding virtual continuum line
Begin loop 2 over all lines of the actual Mw
2. Calculation of the output variables
end loop 2
3. Redo adding of virtual continuum line
end loop 1

Detailed description:

loop 1 over all microwindows
jmw=1→*nselmw*

loop 2 over all lines of the actual Mw
kline=1→*iline(jmw)*

1. Adding virtual continuum line
if [*imaskcont(1)=1*]: *iline(jmw)=iline(jmw)+1, icode(iline(jmw),jmw)=1*
if [*imaskcont(2)=1*]: *iline(jmw)=iline(jmw)+1, icode(iline(jmw),jmw)=7*
if *[imaskcont(3)=1]: iline(jmw)=iline(jmw)+1, icode(iline(jmw),jmw)=22*

2. Calculation of the output variables
Three different types of gas codes are distinguished inside the program:
1. the HITRAN code which is represented by the variable *icode* that attaches to each line of each microwindow the HITRAN code number of the gas.
2. the global gas code of the retrieval which is going from 1 to *igas*, the number of different gases that have to be considered for the retrieval. In this code the number 1 always belongs to the main gas of the retrieval, i.e. in the case of p-T retrieval 1 refers to CO_2. This numbering is connected to the HITRAN gas code by the vector *igashi*, which gives to each global gas number the HITRAN code number, i.e.:
   HITRAN gas number = *igashi(* global gas number *)*
3. the local gas code of each microwindow (*jmw*) which is going from 1 to *igasmw(jmw)*, the number of different gases that have to be considered for each microwindow. As in the global gas code, the number 1 belongs to the main gas of the retrieval (i.e. CO_2 for p-T retrieval). This

numbering is connected to the global gas code by the matrix *igasnr* which gives to each local gas number of each microwindow the global number of the gas, i.e.:
global gas number = *igasnr(*local gas number, *jmw)*

While *icode* is initialised during reading the line data base, the variables *igas, igashi, igasmw,* and *igasnr* are set up in this module by using the information of *icode*.
First, the variables *igashi* and *igasnr* are initialised so that the main gas of the retrieval is number 1 in the local and the global code:

*igashi(*1*)*=*imaingas* (=2 for p-T retrieval)
*igasnr(*1,*jmw)*=1

Then, for each line *kline* of each microwindow *jmw* it is checked, if the related gas (given by *icode(kline,jmw)*) is already included in the global and local gas codes. If this is not the case *igas, igashi, igasmw,* and *igasnr* are enhanced by 1.
During this procedure it is checked that in each microwindow there is at least one line of the main gas. It is also tested that *igas* becomes not larger than *imxgas* and *igasmw(jmw)* ≤ *imxgmw* for each microwindow *jmw*. If one of these conditions is not fulfilled, the program is stopped.

Example:

The inputs are:

Two microwindows are considered for the retrieval: *nselmw*=2.
3 lines in the 1st Mw and 4 lines in the 2nd: $iline = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$.
In the 1st Mw there are CO_2 (HITRAN code =2) and H_2O (HITRAN code =1) lines and in the 2nd Mw there is CH_4 (HITRAN code =6), CO_2, and H_2O: $icode = \begin{bmatrix} 2 & 6 \\ 1 & 2 \\ 2 & 1 \\ & 1 \end{bmatrix}$.
The main gas of the retrieval is CO_2: $imaingas = 2$.

The results of **inigas** are:

Total number of different gases: $igas = 3$.
HITRAN code for each global gas number: $igashi = \begin{bmatrix} 2 \\ 1 \\ 6 \end{bmatrix}$.
Number of different gases per microwindow: $igasmw = \begin{bmatrix} 2 \\ 3 \end{bmatrix}$.
Global gas number for the local gas number of each Mw: $igasnr = \begin{bmatrix} 1 & 1 \\ 2 & 3 \\ & 2 \end{bmatrix}$.

3. Redo adding virtual continuum line
if [*imaskcont(1)=1*]: *iline(jmw)=iline(jmw)-1*
if [*imaskcont(2)=1*]: *iline(jmw)=iline(jmw)-1*
if *[imaskcont(3)=1]: iline(jmw)=iline(jmw)-1*

4.2.1.4 UPLIMIT

Description: module used to check the upper limit of the atmosphere against the uppermost point in the atmospheric profiles.
For the description of this module, please refer to its source code reported in [AD7].

4.2.1.5 READVMR

Description: module used to read the file containing the VMR profiles.
For the description of this module, please refer to its source code reported in [AD7].

4.2.1.6 WMOL

Description: module used for initialising the molecular weights of the chemical species.
For the description of this module, please refer to its source code reported in [AD7].

4.2.1.7 R_APOD_VMR

Description: module used to read the file containing the apodisation function.
For the description of this module, please refer to its source code reported in [AD7].

4.2.2 SAPOD

SAPOD
|----FOUR1]

Description
This module calculates the apodisation function in the spectral domain from the input apodisation function in the interferogram domain.

Variables exchanged with external modules

Name	Description
rapod	real*4 rapod(imxapo) = apodisation function represented in the OPD domain
napod	integer*4: no. of points used to represent apodisation function in OPD domain (rapod)
rapod_sigma	real*4 rapod_sigma(imxilc): apodisation function in the spectral domain
nailsdp	integer*4: no. of points used to represent the apodisation in the spectral domain

Detailed description

The apodisation function in the spectral domain *rapod_sigma(imxilc)* is calculated by performing the following operations:

- The number of points used to make the FFT is determined:

$$nn = (napod - 1) \cdot 2$$

  Note that $napod = 2^n + 1$, with *n* integer: this control has been performed in **input** routine

- The vector *rapod* is stored in a proper way into a complex vector. This allows to use the module **four1** for performing the FFT:

  complex*8 *rapod_xft(imxapo*2)*
  rapod_xft(1) = cmplx*(rapod(1),0.)*
  begin loop *i=2, ..., (napod-1)*
  rapod_xft(i) = cmplx*(rapod(i),0.)*
  rapod_xft(nn-i+2) = cmplx*(rapod(i),0.)*
  end loop *i.*
  rapod_xft(napod) = cmplx*(rapod(napod),0.)*

- The FFT of *rapod_xft* is then performed using **four1**:

  four1(*rapod_xft,nn,1*)

[Refer to W.H.Press and others: 'Numerical Recipes in Fortran', Second Edition, Cambridge University Press, pag.501 for the description of the module **four1**. Note that the FFT of *rapod_xft* is overwritten in *rapod_xft.*]

- The vector *realapod_sigma* defined as *realapod_sigma(i)* = REAL*(rapod_xft(i)/nn)* for *i=1, ..., nn,* is computed.
- The vector *rapod_sigma,* containing the apodisation function in the spectral domain, is finally built: it consists of *nailsdp* points corresponding to both the negative and positive frequencies, centered around the 0-frequency point, of the apodisation function.

4.2.3 OCCUSIM

For the description of this module, please refer to section 2.2.5.

4.2.4 CHBASE

For the description of this module, please refer to section 2.2.7.

4.2.5 FAILS

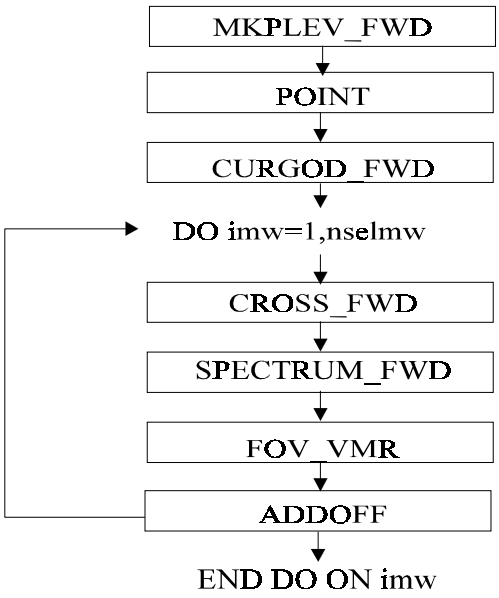
For the description of this module, please refer to section 2.2.8.

4.2.6 GRID

For the description of this module, please refer to section 2.2.9.

4.2.7 FWDMDL

This module is used only to carry-out the flow of the calls of the forward model, that is represented here below.



4.2.7.1 CROSS_FWD

```
CROSS_FWD
   |-----LOFICO ]
   |-----FLINT *
   |((((+SHAPECALC
   |        |(----HUMLI ]
   |((((+HUMLI ]
   |((((+FCO2CHI *
   |((((+POLCOE2ND +
   |((((+FCONH2O *
   |((((+FCONN2 *
   |((((+FCONO2 *
```

Description

- Using the spectroscopic line-data this routine determines the absorption cross-sections for each general wavenumber fine grid point, each IAPT number and each gas which has to be considered in the actual microwindow using the equivalent pressures and temperatures calculated in '**curgod_fwd**'.

Variables exchanged with external modules

Name:	Description:
imw	number of the actual Mw
rpeq	equivalent pressures
rteq	equivalent temperatures
itglev	number of the tangent-level for each geometry
isigma	number of wavenumber grid points for each Mw
dsigma	general wavenumber fine grid
delta	general fine grid interval [cm-1]
igeo	number of simulated geometries
iocsim	occupation matrix for the simulations to be performed
igasmw	number of gases to be considered in each Mw
ruplin	upper limit where the line has to be considered [km]
rlolin	lower limit where the line has to be considered [km]
iline	number of lines in each microwindow
icode	HITRAN code for each line of each Mw
rint0	line intensity for each line of each Mw
relow	lower state energy for each line of each Mw
rhw0	foreign broadened half width for each line of each Mw
dsilin	central wavenumber for each line of each Mw
ioutin	flag for each line of each Mw =1: line-shape has to be calculated at each wavenumber inside the Mw =2: line is considered as nearby continuum (calculated at three points inside the Mw
igasnr	global gas number for the local gas number of each Mw
rexph	exponent for T dependence of half width for each line of each Mw
rwmol	molecular weight for each HITRAN molecular code and isotope number
igashi	HITRAN code number for each global gas number
iiso	isotope number for each line of each Mw
ipoint	IAPT-number for each layer and each geometry
ninterpol	switch for the decision of interpolation of the absorption cross-sections for the geometries above the lowest geometry (only if the IAPT number of the path is increasing, which was decided during the calculation of ipoint) =-1: no interpolation, all cross-sections recalculated =0: all cross-sections above the lowest geometry are interpolated =1: new calculation only of the tangent-layer, all other layers interpolated =2: new calculation of the tangent-layer and the layer above, all others interpolated =3: ...
raircol	air-column for each layer and each geometry
rcol	column amounts for each layer, each geometry and each gas
ropath	
nswh2ocon	=imaskcont(1): switch for H_2O continuum nswh2ocon=1: H_2O continuum considered
nswn2con	=imaskcont(3): switch for N_2 continuum nswn2con=1: N_2 continuum considered

nswo2con	=imaskcont(2): switch for O_2 continuum nswo2con=1: O_2 continuum considered
rzmod	heights of levels used for the radiat. tranf. calculation
rcross	absorption cross sections for each general wavenumber fine grid point (1st index), each IAPT number (2nd index) and each gas (3rd index) for the actual Mw

Module structure

1. Initialisation of variables
Begin loop 1 over the geometries valid for the actual microwindow
Begin loop 2 over the layers of the actual geometry for which a new cross- section must be determined
Begin condition 1 the cross sections are interpolated
Begin loop 3 over the gases of the actual Mw
2. Interpolation of the cross sections
end loop 3
else condition 1 the cross sections are calculated
3. Definition of local fine and coarse wavenumber grid
Begin loop 4 over all lines of the actual Mw that must be considered for the actual altitude
4. Initialisation of variables for line-calculation
Begin condition 2 the lines are calculated at each point
Begin condition 3 a line shape for HNO_3 is precalculated
5. Precalculation of HNO_3 line shape
end condition 3
6. Calculation of the line in the local coarse grid
7. Calculation of the line in the local fine grid
else condition 2 the lines are handled as near continuum
8. Calculation of the line at 3 points inside the Mw
end condition 2
Begin condition 4 if the H_2O continuum is calculated
9. The contribution at 25 cm^{-1} is subtracted from the line
end condition 4
end loop 4
Begin loop 5 over the gases of the actual Mw
10. Interpolation of the cross sections from the local coarse and fine grid to the general fine grid
11. Interpolation of the nearby continuum to the general fine grid
Begin condition 5 if the H_2O continuum is calculated
12. The H_2O continuum is calculated
end condition 5
Begin condition 6 if the N_2 continuum is calculated
13. The N_2 continuum is calculated
end condition 6

Begin condition 7 if the O_2 continuum is calculated
14. The O_2 continuum is calculated
end condition 7
end loop 5
end condition 1
end loop 2
end loop 1

Detailed description:

loop 1 over the geometries valid for the actual microwindow
kgeo=igeo→1
if [*iocsim(kgeo,imw)*≠0]
Starting from the lowest geometry (*igeo*) this loop (i.e. the commands inside the loop) is only executed if this observation geometry has to be simulated for the actual Mw.

loop 2 over the layers of the actual geometry for which a new cross-section must be determined
llay=1→itglev(kgeo) -1
This loop begins from the outer layer and goes down to the tangent layer (*itglev(kgeo)-1*). It is only executed if new cross-sections must be calculated, i.e. if the IAPT-number *ipoint(llay,kgeo)* is increasing. For the cases that the IAPT number is not increasing, the cross-sections have already been calculated during an earlier execution.

Condition 1: the cross sections are interpolated or calculated
The cross sections are interpolated (using the cross sections which have already been calculated for the lowest geometry) if we are not in the lowest geometry and if we are in a layer that has to be interpolated (indicated by *ninterpol*):

if [*kgeo<ilowgeo* ∧ *llay* < *itglev(kgeo)-ninterpol* ∧ *ninterpol*≠-1]

Where *ilowgeo* is the lowest geometry that must be calculated for the actual Mw.
If this conditions are not fulfilled the cross sections are calculated explicitly using the line data.

loop 3 over the gases of the actual Mw
mgas=1→igasmw(imw)

loop 4 over all lines of the actual Mw that must be considered for the actual altitude
mline=1,iline(imw)
if [*ruplin(mline,imw)>rzmod(llay)>rlolin(mline,imw)*]

condition 2 the lines are calculated at each point or handled as continuum
if [*ioutin(mline,imw)=1*]: the lines are explicitly calculated at each point of the local coarse and fine grid.
if [*ioutin(mline,imw)=2*]: the lines are handled as near continuum and calculated only at three points inside the Mw.

condition 3 a line shape for HNO_3 is precalculated
if the gas is HNO_3 (if [*icode(mline,imw)=nrepcode*]) and the half width is equal to the reference half width (if [*rhw0(mline,imw)=rephw0*]) and the half width exponent is equal to the reference exponent (if [*rexph(mline,imw)=repexph*]) and if the line shape has not already been precalculated (if [*nshape=0*]) then a line shape is precalculated.

condition 4 if the H_2O continuum is calculated
if *[icode(mline,imw)=1*] and [*nswh2ocon=1*]

loop 5 over the gases of the actual Mw
mgas=1→igasmw(imw)

condition 5 if the H_2O continuum is calculated
if [*igashi(igasnr(mgas,imw))=1*] and [*nswh2ocon=1*]

condition 6 if the N_2 continuum is calculated
if [*igashi(igasnr(mgas,imw))=22*] and [*nswn2con=1*]

condition 7 if the O_2 continuum is calculated
if [*igashi(igasnr(mgas,imw))=7*] and [*nswo2con=1*]

1. Initialisation of variables

- Calculation of vector *igasact(imxhit)* that gives for each hitran gas number the local Mw gas number: *igasact(igashi(igasnr(j,imw))) = j* for $1 \le j \le igasmw(imw)$
- Determination of the line with the largest intensity of the main gas: line number: *imaxlin*

2. Interpolation of the cross sections

The cross sections for the geometries above the lowest geometry are calculated (for each general fine grid point) by linear interpolation using the cross sections already calculated for the lowest geometry. This linear interpolation is performed with respect to the equivalent pressures, i.e. it is first decided between which equivalent pressures of the lowest geometry the actual equivalent pressure lies and than the cross sections are interpolated to the actual equivalent pressure. This is done for the cross sections of all gases (*rcross*).
E.g. for *rcross* the formula for all wavenumbers on the general fine wavenumber grid (*msig*) is:

$$rcross(msig, ipoint(llay, kgeo), mgas) = r1 + (r2 - r1) \cdot \frac{p - p1}{p2 - p1}$$

$$r1 = rcross(msig, ipoint(llay1, ilowgeo), mgas)$$

$$r2 = rcross(msig, ipoint(llay2, ilowgeo), mgas)$$

with: $p = rpeq(ipoint(llay, kgeo), igasnr(mgas, imw))$

$$p1 = rpeq(ipoint(llay1, ilowgeo), igasnr(mgas, imw))$$

$$p2 = rpeq(ipoint(llay2, ilowgeo), igasnr(mgas, imw))$$

Where *llay1* and *llay2* determine the pressures *p1* and *p2* of the lowest geometry between which the actual pressure *p* lies.

3. Definition of local fine and coarse wavenumber grid

The local (for the actual geometry and layer) coarse and fine wavenumber grid is defined by calling the module **lofico**:

$$\text{lofico}\left[\begin{array}{l} dsilin, ipoint(llay, kgeo), imw, rwmol, icode, iiso, rhw0, imaxli, rpeq, rteq, \\ delta, isigma, dsigma, igasmw, rexph, iqlfgf, dsiglf, dsiglc, isiglf, isiglc, \\ deltalf, deltalc, rcrolf, rcrolc, rcrolfpert, rcrolcpert \end{array}\right]$$

4. Initialisation of variables for line-calculation

- Calculation of the Doppler half width:

$$rdhalf = dsilin(mline, imw) \cdot dcdop \cdot \sqrt{\frac{rteq(ipo, ign)}{rwmol(icode(mline, imw), iiso(mline, imw))}}$$

with: $ipo = ipoint(llay, kgeo)$

and: $ign = igasnr(igasact(icode(mline, imw)), imw)$, the global gas number for the hitran gas number of the actual line.

dcdop is a parameter.

- Calculation of the Lorentz half width:

$$rlhalf = rhw0(mline, imw) \cdot \frac{rpeq(ipo, ign)}{rp0h} \cdot \left[\frac{rt0h}{rteq(ipo, ign)}\right]^{rexph(mline, imw)}$$

With the parameters *rp0h, rt0h.*

- Calculation of the line intensity

The line intensity *rlint* is calculated by a call to the module **flint**:

$$rlint = \text{flint}\left[\begin{array}{l} rint0(mline, imw), relow(mline, imw), rteq(ipo, ign), \\ dsilin(mline, imw), icode(mline, imw), iiso(mline, imw) \end{array}\right]$$

5. Precalculation of HNO_3 line shape

In the case of HNO_3 the line shape is precalculated for the Voigt part of the line. This precalculation is performed in the general fine grid by a call to the subroutine

$$shapecalc\left[\begin{array}{l} ipo, ign, rteq(ipo, ign), dsigma(1, imw), isigma(imw), delta, \\ rwmol(icode(mline, imw), iiso(mline, imw)), iprec, rshape \end{array}\right]$$

Then the variable *nshape* is set to 1 in order to indicate, that for this IAPT number *ipo* the shape is already precalculated. When going to the next IAPT *nshape* has to be initialised again to 0 (before begin of next loop 4 over all lines)!

6. Calculation of the line in the local coarse grid

The cross sections on the local coarse grid *rcrolc (*dimension *(imxsig,imxgmw)*) are calculated from the boundaries of the microwindow up to a distance of $(rdhalf + rlhalf) \cdot rvmult$ wavenumbers from the line centre by using the Lorentz function (*rvmult* is a parameter). In the region around the line centre the cross sections on the fine grid are constant. This constant is determined as the mean value of the last Lorentz calculated cross sections on the left and on the right of the line.
The boundary indices for the Lorentz calculation on the local coarse grid are:

$$i1c = 1$$

$$i2c = \text{nint}\left[\frac{dsilin(mline,imw) - (rdhalf + rlhalf)\cdot rvmult - dsiglc(1)}{deltalc}\right] + 1$$

$$i3c = \text{nint}\left[\frac{dsilin(mline,imw) + (rdhalf + rlhalf)\cdot rvmult - dsiglc(1)}{deltalc}\right] + 1$$

$$i4c = isiglc$$

Where *isiglc, dsiglc, deltalc* have been determined in 3.
(One has to take care that for a line very near to the boundary of the microwindow (where i2c could become less than *i1c* ...) these coeffiecents are set to the boundary values!)

Calculation of Lorentz function for $i1c \le i \le i2c\text{-}1$ and $i3c+1 \le i \le i4c$:

$$rlinfctlc(i) = \frac{1}{\pi}\frac{rlhalf}{rlhalf^2 + (dsiglc(i) - dsilin(mline,imw))^2}$$

Calculation of the cross sections and adding to the cross sections from the previous lines:

$$rcrolc(i,ig) = rlint \cdot rlinfctlc(i) + rcrolc(i,ig)$$

with: $ig = igasact(icode(mline,imw))$, the local gas number for the actual line.

The value for the 'plateau' region, i.e. in the vicinity of the line centre is:

$$rplatfctn = \frac{rlinfctlc(i2c-1) + rlinfctlc(i3c+1)}{2}$$

So, for $i2c \le i \le i3c$:

$$rcrolc(i,ig) = rlint \cdot rplatfctn + rcrolc(i,ig)$$

7. Calculation of the line in the local fine grid

On the local fine grid the lines are only calculated in the vicinity of the line, where the cross sections on the local coarse grid are constant (see 5.), i.e. for distances less than $(rdhalf + rlhalf) \cdot rvmult$ wavenumbers from the line centre. In this region the line profile is partly calculated by the Lorentz and partly by the Voigt function. The Voigt function is used inside an intervall of $\pm rdhalf \cdot rdmult$ wavenumbers from the line centre (*rdmult* is a parameter).
The boundary indices on the local fine grid are:

$$i1f = (i2c - 2) \cdot iqlclf + 2$$

$$i2f = \text{nint}\left[\frac{dsilin(mline, imw) - rdhalf \cdot rdmult - dsiglf(i1f)}{deltalf}\right] + i1f$$

$$i3f = \text{nint}\left[\frac{dsilin(mline, imw) + rdhalf \cdot rdmult - dsiglf(i1f)}{deltalf}\right] + i1f$$

$$i4f = i3c \cdot iqlclf$$

With the parameter *iqlclf,* the quotient between the local coarse and fine grid.

Calculation of Lorentz function for $i1f \le i \le i2f\text{-}1$ and $i3f\text{+}1 \le i \le i4f$:

$$rlinfctlf(i) = \frac{1}{\pi} \frac{rlhalf}{rlhalf^2 + \left(dsiglc(i) - dsilin(mline, imw)\right)^2}$$

Calculation of the cross sections and adding to the cross sections from all the previous lines:

$$rcrolf(i, ig) = rlint \cdot rlinfctlf(i) - rplatcro + rcrolf(i, ig)$$

with: $ig = igasact\left(icode(mline, imw)\right)$, the local gas number for the actual line, and *rplatcro* the coarse grid 'plateau' value which was determined in 5.

For $i2f \le i \le i3f$ the line function is determined by the Voigt lineshape:

$$rlinfctlf(i) = \sqrt{\frac{\ln 2}{\pi}} \frac{rre}{rdhalf}$$

where *rre* is the result from a call to the routine **humli**(*rx,ry,rre*), with:

$$rx = \sqrt{\ln 2} \frac{|dsiglf(i) - dsilin(mline, imw)|}{rdhalf}$$

$$ry = \sqrt{\ln 2} \frac{rlhalf}{rdhalf}$$

The cross sections are calculated from *rlinfctlf* like in the case of the Lorentz calculation (see above).

In the case of HNO_3 for the Voigt part the precalculated line shape is interpolated linearly. If the gas is HNO_3 (if [*icode(mline,imw)=nrepcode*]) and the half width is equal to the reference half width (if [*rhw0(mline,imw)=rephw0*]) and the half width exponent is equal to the reference exponent (if [*rexph(mline,imw)=repexph*]):

$$rlinfctlf(i) = \sqrt{\frac{\ln 2}{\pi}} \frac{r2}{rdhalf}$$

where *r2* is the linear interpolation of the precalculated line shape *rshape* (centred in the centre of the actual line) to the actual local fine grid wavenumber *dsiglf(i)*.

8. Calculation of the line at 3 points inside the Mw

For lines outside the microwindow which are taken into account as near continuum, the cross sections are calculated at the first point, at the middle point and at the last point of the microwindow. Later, in 10., they will be interpolated to the general fine grid.
The procedure is:

- calculating the line profile using the Lorentz line shape (see above) at the three wavenumbers inside the microwindow.

- if the line is a CO_2 line (if [icode(mline,imw)=2]) the profile is multiplied with the CO_2 chi factor which is calculated by a call to module **fco2chi**:

  $$\text{fco2chi}\left[rteq(ipo,ign), dconsi - dsilin(mline,imw), 1\right]$$

  ipo and *ign* have been defined in 4..

- The absorption cross sections at the 3 points inside the Mw are now calculated like in 7 or 8 by multiplication of the profile with and added to the near continuum cross sections from the previous line calculation.

9. The contribution at 25 cm^{-1} is subtracted from the line

The value of the Lorentz line shape (if the line is a water line) at 25 cm^{-1} from the line centre is subtracted from the H_2O cross-sections (if H_2O continuum has to be considered):

$$r1 = rlint \cdot \frac{1}{\pi} \frac{rlhalf}{rlhalf^2 + 625}$$

and for $1 \le i \le isigma(imw)$:

$$rcross(m1sig, ipo, ig) = rcross(m1sig, ipo, ig) - r1$$

10. Interpolation of the cross sections from the local coarse and fine grid to the general fine grid

For each gas of the microwindow, the output cross section vector *rcross(i,ipo,mgas)* of the general fine grid is filled by linear interpolation in wavenumber using the vectors *rcrolf(j,mgas)* and *rcrolc(k,mgas)*, where *j* is the index on the local fine grid and *k* on the local coarse grid.

11. Interpolation of the nearby continuum to the general fine grid

For each gas of the Mw the nearby continuum values which were calculated in 8. for three points inside the microwindow are interpolated (2nd order) to the general wavenumber fine grid and added to the cross section output vectors *rcross*. The coefficients for the parabolic interpolation are calculated using module **polcoe2nd**.

12. The H_2O continuum is calculated

Calculation of the mean density of water and of the other gases in the path:

$$rsden = \frac{rcol(llay, kgeo, igasnr(mgas, imw))}{10^5 \cdot ropath(llay, kgeo)}$$

$$rfden = \frac{raircol(llay, kgeo)}{10^5 \cdot ropath(llay, kgeo)} - rsden$$

Calculation of the H_2O continuum at the boundaries of the microwindow by two calls to the subroutine:

$$\text{fconh2o}\begin{bmatrix} dsigma(1, imw) \text{ or } dsigma(isigma(imw), imw), \\ rteq(ipo, igasnr(mgas, imw)), rsden, rfden \end{bmatrix}$$

The results are linearly interpolated to the other general wavenumber fine grid points and added to the actual cross sections of H_2O.

13. The N_2 continuum is calculated

Calculation of the N_2 continuum at the boundaries of the microwindow by two calls to the subroutine:

$$\text{fconn2}\begin{bmatrix} dsigma(1, imw) \text{ or } dsigma(isigma(imw), imw), \\ rteq(ipo, igasnr(mgas, imw)), rpeq(ipo, igasnr(mgas, imw)) \end{bmatrix}$$

The results are linearly interpolated to the other general wavenumber fine grid points and written into the actual cross sections of N_2.

14. The O_2 continuum is calculated

Calculation of the O_2 continuum at the boundaries of the microwindow by two calls to the subroutine:

$$\text{fcono2}\begin{bmatrix} dsigma(1,imw) \text{ or } dsigma\big(isigma(imw),imw\big), \\ rteq\big(ipo,igasnr(mgas,imw)\big),rpeq\big(ipo,igasnr(mgas,imw)\big)\end{bmatrix}$$

The results are linearly interpolated to the other general wavenumber fine grid points and written into the actual cross sections of O_2.

4.2.7.2 FCONH2O

Description
Calculation of the water vapour continuum. It is an implementation of the routine 'contnm_ckd_2.1.f' revision 3.3 (28-4-94) of Clough.
References:
S.A. Clough, F.X. Kneizys, R.W. Davies, 'Line shape and the water vapour continuum', Atmospheric Research, 23, 229-241, 1989.

Variables exchanged with external modules:

Name	Dimension	Description
dsi		wavenumber where the H_2O -continuum should be calculated
rt		equivalent temperature of H_2O of the actual path
rsden		mean density of H_2O in the actual path [Molecules/cm^3]
rfden		mean density of all other gases in the actual path [Molecules/cm^3]
fconh2o		absorption cross section of the water continuum [cm^2/Molecule]

Module structure:

1. Calculation of water continuum.

Detailed description:

The program uses parameterised values for the self continuum at the temperatures 296 K and 260 K and for the foreign continuum at 296 K. These data are in the block data routines **h2os296, h2os260** and **h2of296** which will be given as source codes.

The source code of **fconh2o** is given here since it also contains many parameters:

```
real*8 function fconh2o(dsi,rt,rsden,rfden)
      implicit none
      include 'parameters.inc'
      real*8 rt,rsden,rfden,r0den,rxfac(0:50),rs296(2003),rs260(2003),
     &       rf296(2003),rs0,rs1,rs,rf,rfs,r1,r2,r3
      real*8 dsi,dsi1,dsi2,ddsi,dsii1,d1,d2,d3
      integer*4 inpt,j,i1,i2
      common/h2os0/rs296
      common/h2os1/rs260
      common/h2of/rf296
      parameter(r0den=1013./(rbc*296.))
*
*  These are self-continuum modification factors from 700-1200 cm-1
```

```
*
      data (rxfac(j),j=0,50)/
     1    1.00000,1.01792,1.03767,1.05749,1.07730,1.09708,
     2    1.10489,1.11268,1.12047,1.12822,1.13597,1.14367,
     3    1.15135,1.15904,1.16669,1.17431,1.18786,1.20134,
     4    1.21479,1.22821,1.24158,1.26580,1.28991,1.28295,
     5    1.27600,1.26896,1.25550,1.24213,1.22879,1.21560,
     6    1.20230,1.18162,1.16112,1.14063,1.12016,1.10195,
     7    1.09207,1.08622,1.08105,1.07765,1.07398,1.06620,
     8    1.05791,1.04905,1.03976,1.02981,1.00985,1.00000,
     9    1.00000,1.00000,1.00000/
*
*  first,last wavenumber, wavenumber difference and number of points
*  in the blockdata files for the h2o-continuum
*
      data dsi1,dsi2,ddsi,inpt /  -20.0, 20000.0, 10.0, 2003/
*
*  linear interpolation in wavenumber
*
      i1=int((dsi-dsi1)/ddsi) + 1
      dsii1=dsi1+(i1-1)*ddsi
      r1=(dsi-dsii1) / ddsi
      i2=i1+1
      rs0=rs296(i1)+(rs296(i2)-rs296(i1)) * r1
      rs1=rs260(i1)+(rs260(i2)-rs260(i1)) * r1
      rf=rf296(i1)+(rf296(i2)-rf296(i1)) * r1
*
*  exponential interpolation in temperature of the self-continuum
*  rs0*(rs1/rs0)^((296K-T)/(296K-260K))
*
      rs=rs0*(rs1/rs0)**((296.-rt)/36.)
*
*  correction to the self-continuum
*   (if the wavenumber dsi is between 700 and 1200cm-1 a
*    further correction is made using the rxfac -factors
*    (linearly interpolation: not in Clough's 28-4-94 version))
*
      d1=(dsi-1310.)**2
      r2= ( 1.-0.2333*40000./(40000.+ (dsi-1050.)**2) ) *
     &    ( 1.-0.15*14400./(14400.+d1+5.e-6*d1*d1) )
      if (dsi.gt.700.0.and.dsi.lt.1200.0) then
        i1=int((dsi-700.)/10.)
        r3=rxfac(i1)+(rxfac(i1+1)-rxfac(i1))*(dsi-700.-i1*10.)/10.
        rs=rs * r2 * r3
      else
        rs=rs * r2
      end if
*
*  correction to the foreign-continuum
```

```
*
      d2=(dsi-1130.)**2
      d3=(dsi-1900.)**2
      rf=rf *
     &   ( 1.-0.97*108900./(108900.+d2+8.e-11*d2*d2*d2) ) *
     &   ( 1.-0.6*22500./(22500.+d3+3.e-6*d3*d3) )
*
*  weighting of the self and foreign continuum parts and adding them
*
      rfs=1.e-20 * (rs*rsden + rf*rfden) / r0den
*
*  calculation of the absorption cross-section
*
      fconh2o=dsi * tanh(0.5*rhck*dsi/rt) * rfs
      end
```

4.2.7.3 FCONN2

Description

Calculation of the N_2- continuum (2090-2650cm-1).
Reference:
V. Menoux, R. Le Doucen, C. Boulet, A. Roblin, and A.M. Bouchardy, 'Collision-induced absorption in the fundamental band of N_2: temperature dependence of the absorption for N_2-N_2 and N_2-O_2 pairs', Appl. Opt., 32, 263-268, 1993.

Variables exchanged with external modules:

Name	Dimension	Description
dsi		wavenumber where the N_2 -continuum should be calculated
rt		equivalent temperature of N_2 of the actual path
rp		equivalent pressure of N_2 of the actual path
fconn2		absorption cross section of the N_2 continuum [cm^2/Molecule]

Module structure

1. Calculation of N_2 continuum.

Detailed description:

Since this subroutine contains many values for the parameterised continuum, it will be provided as a source code.
The program calculates the continuum in the wavenumber range 2090-2650 cm^{-1}. The binary absorption coefficients for N-N collisions are given at different wavenumber grid points (5-10 cm^{-1} distance) for 6 temperatures. In the first step these are linearly interpolated to the actual wavenumber *dsi* and temperature *rt*. The result is the binary absorption coefficient at *rt* and *dsi*: *rbiabco_nn*.

Then the linear collision efficiency, which is given for the 6 temperatures is also interpolated linearly to *rt*. Result: *reff*.

The air density is calculated by: $rden = 7.24292 \cdot \frac{rp}{rt}$.

At last the absorption cross section for N_2 is determined:

$$fconn2 = (0.789 + reff \cdot 0.211) \cdot rbiabco\_nn \cdot rden \cdot 1.3852919 \cdot 10^{-45}$$

4.2.7.4 FCONO2

Description
Calculation of the O_2- continuum (1400-1800 cm^{-1}).
Reference:
J.J. Orlando,G.S.Tyndall,K.E.Nickerson,J.G.Calvert, 'The temperature dependence of collision induced absorption by Oxygen near 6 μm', JGR,96,D11,20755-20760

Variables exchanged with external modules

Name	Dimension	Description
dsi		wavenumber where the O_2 -continuum should be calculated
rt		equivalent temperature of O_2 of the actual path
rp		equivalent pressure of O_2 of the actual path
fcono2		absorption cross section of the O_2 continuum [cm^2/Molecule]

Module structure

1. Calculation of O_2 continuum.

Detailed description:
The program uses the block data file **cocono2** which contains the parameters for the O_2 continuum. The continuum is only calculated in the range 1400-1800 cm^{-1}.

First the coefficients for O_2-N_2 and O_2-O_2 are linearly interpolated to *rsi*. Results: *ran2i, rbn2i, rcn2i, rao2i, rbo2i, rco2i.*

Then the binary absorption coefficients for O_2-N_2 (*rkn2*) and for O_2-O_2 (*rko2*) are calculated for the actual temperature *rt* in the following way:

$$rkn2 = ran2i + rbn2i \cdot \frac{rt}{100} + rcn2i \cdot \left(\frac{rt}{100}\right)^2$$

$$rko2 = rao2i + rbo2i \cdot \frac{rt}{100} + rco2i \cdot \left(\frac{rt}{100}\right)^2$$

The air density is calculated by: $rden = 7.24292 \cdot \frac{rp}{rt}$.

At last the absorption cross section for O_2 is determined:

$$fcono2 = \left(0.789 \cdot rkn2 + 0.211 \cdot rko2\right) \cdot rden \cdot 10^{-45}$$

4.2.7.6 FCO2CHI

For the description of this module, see section 2.2.11.12

4.2.7.7 FLINT

For the description of this module, see section 2.2.11.13

4.2.7.8 FPARTS

For the description of this module, see section 2.2.11.14

4.2.7.9 HUMLI

For the description of this module, see section 2.2.11.15

4.2.7.10 POLCOE2ND

For the description of this module, see section 2.2.11.16

4.2.7.11 LOFICO

For the description of this module, see section 3.2.11.23.

4.2.7.12 SHAPECALC

For the description of this module, see section 3.2.11.17.

4.2.7.13 FOV_VMR

For the description of this module, see section 3.2.11.10

4.2.7.14 FOV3

For the description of this module, see section 3.2.11.11

4.2.7.15 FOV4

For the description of this module, see section 3.2.11.12

4.2.7.16 FOV5

For the description of this module, see section 3.2.11.13

4.2.7.17 SPECTRUM_FWD

SPECTRUM_FWD]
|-----CONV *

Description

- calculation of the original spectra on the general wavenumber fine grid for all geometries of the actual microwindow
- convolution of these spectra and derivatives with the AILS function to the general wavenumber coarse grid

Variables exchanged with external modules

Name	Description
imw	number of the actual Mw
itglev	number of the tangent-level for each geometry
igasmw	number of gases to be considered in each Mw
igasnr	global gas number for the local gas number of each Mw
isigma	number of wavenumber grid points for each Mw
rcross	absorption cross sections for each wavenumber each IAPT and each gas for the actual MW
rcol	column amounts for each layer, each geometry and each gas
ipoint	IAPT-number for each layer and each geometry
ipath	number of different IAPT-numbers of ipoint
rtmain	equivalent temperature of the main gas
dsigma	general wavenumber fine grid
igeo	number of simulated geometries
iocsim	occupation matrix for the simulations to be performed
nsam	n. of sampling points in each Mw (general coarse grid)
nils	number of elements of rils
rspct	spectrum for each geometry on the general coarse grid 1st index: general wavenumber coarse grid 2nd index: geometries to be simulated for the actual Mw
rils	instrument-line-shape function on the general fine grid
rintils	ratio between the frequency step approximating infinitesimal spectral resolution and the integral of the ILS function
nrd	Ratio between general coarse grid step and fine grid step
delta	general fine grid interval [cm-1]
iadd	number of fine-wavenumber grid points to be added on both sides of each microwindow (due to the ils-convolution)

Module structure

1. Initialisation of variables and Planck function for later interpolation
Begin loop 1 over geometries valid for the actual microwindow
Begin loop 2 over general wavenumber fine grid

2. Interpolate Planck function
Begin loop 3 over the layers of the actual geometry
3. Calculation of the transmissions
End loop 3
4. Calculation of the radiative transfer
End loop 2
5. Convolution of the spectra with the AILS function
End loop 1

Detailed description:

loop 1 over geometries valid for the actual microwindow
jgeo=1→*igeo*
if (*iocsim(jgeo,imw)*≠0)

loop 2 over general wavenumber fine grid
ksigma=1→*isigma(imw)*

loop 3 over the layers of the actual geometry
klay=1→*nlay*
nlay=itglev(jgeo)-1 is the tangent layer.

1. Initialisation of variables and Plack function for later interpolation
Output variables set to 0.
The Planck function values at the first grid point and the last grid point of the actual microwindow and from this the increment for the later linear interpolation is calculated for the temperatures of the profiles of the main gas (*rtmain).* This is done for all different IAPT-numbers ($1 \le jpath \le ipath$).
The formula used for the Planck function is:

$$B = \frac{rc1 \cdot \sigma^3}{\exp\left[\frac{rhck \cdot \sigma}{T}\right] - 1}$$

$T = rtmain(jpath)$
$\sigma = dsigma(1,imw)$ or $\sigma = dsigma(isigma(imw),imw)$
(rc1, rhck: parameters)

2. Interpolate Planck function
Using the values calculated in 1. the Planck function is linearly interpolated to the actual wavenumber for all IAPT numbers (*jpath=1→ipath)*:
The results are the interpolated Planck function values for the T-profile (*rtmain):*
db(jpath),

3. Calculation of the transmission
The transmission for each layer is calculated by the formula:

$$rtau(klay) = \exp\left[-\sum_{mgas=1}^{igas}\left\{\begin{matrix} rcross(ksig, ipoint(klay, jgeo), mgas)\cdot \\ rcol(klay, jgeo, igasnr(mgas, imw)) \end{matrix}\right\}\right]$$

Two other variables are also determined:

$$rtau1(klay) = \prod_{l=1}^{klay-1} rtau(l)$$

and:

$$rtau2(klay) = rtau1(klay)\cdot rtau(klay)\cdot \prod_{l=klay+1}^{nlay} rtau(l)^2$$

with: $nlay = itglev(jgeo) - 1$, the number of layers for the actual geometry,

and the definition: $\prod_{l=m}^{m-1} x_l \equiv 1$.

4. Calculation of the radiative transfer
The spectrum is determined by the equation:

$$rsp(ksig) = \sum_{klay=1}^{nlay} db(ipoint(klay, jgeo))\cdot\left(1 - rtau(klay)\right)\left(rtau1(klay) - rtau2(klay)\right)$$

with: $nlay = itglev(jgeo) - 1$,
and *db*, the value of the Planck function for each IAPT-number. *db* was determined in 3. by linear interpolation to the actual general fine grid wavenumber.

5. Convolution of the spectra with the AILS function
In a call to module **conv** the convolution with the AILS function *rils* is performed for the original spectrum *rsp*. The results are the spectra and derivatives on the general coarse wavenumber grid: *rspct.*

4.2.7.18 CONV

For the description of this module, see section 2.2.25.

4.2.7.19 MKPLEV_FWD

MKPLEV_FWD
|(----CHECK]
|((?--CHECK]
|((?--LININT *
|((((-LININT *
|((((+ESPINT *
|((((+GRAVITY *
|((((+GRAVITY *

Description: builds the layering of the atmosphere that allows the calculation of the radiative transfer integral.

Variables exchanged with external modules:

Name	Description
rzsi	rzsi(imxgeo) = tangent altitudes of the geometries to be simulated
igeo	igeo = number of simulated geometries
rzbase	rzbase(imxpro) = altitude of the base-levels
rtbase	rtbase(imxpro) = temperature of the base levels
rpbase	rpbase(imxpro) = pressure on the base-levels
rvmrbase	rvmrbase(imxpro,imxgas) = volume mixing ratio of the gases on the base levels
ibase	ibase = number of base-levels
rulatm	rulatm = upper limit of the atmosphere
rwmolref	rwmolref = molecular weigth of the gas that has been selected as a reference for building the levels.
dsigm0	dsigm0 = Centre frequency of the line selected as a reference for building the levels.
rhw0ref	rhw0ref = half-width of the line selected as a reference for building the levels.
rmaxtv1	rmaxtv1 = max. allowed temperature variation (K) between two neighbouring levels, when the lower level is located below rzt12.
rmaxtv2	rmaxtv2 = max. allowed temperature variation (K) between two neighbouring levels, when the lower level is located above rzt12.
rzt12	rzt12 = altitude (km) where the temperature thresholds rmaxtv1 and rmaxtv2 are exchanged.
rhwvar	rhwvar = max. allowed half-width variation of the selected reference line between two neighbouring levels.
igas	igas = total number of different gases
rexphref	rexphref = exponent for the calculation of Lorentz h-w for the line selected as a reference for building the levels.
rincz	rincz = guess altitude increment (km) used for building the levels above the highest simulated geometry.
redfact	redfact = reduction factor applied to 'rincz' when it produces not

	acceptable P levels above the highest simulated geometry.
rlat	rlat = actual latitude (degrees)
lfitgeo	lfitgeo(imxgeo) = logical vector which identifies the simulations which correspond to a fitted point in the T profile, among all the simulations to be performed.
rzmod	rzmod(imxlev) = heights of model levels used for the radiat. transf. calc.
rpmod	rpmod(imxlev) = pressure on model levels used for the radiat. transf. calc.
rtmod	rtmod(imxlev) = temperature on model levels used for the radiat. transf. calc.
rmrmod	rmrmod(imxlev,imxgas) = volume mixing ratio for each gas considered in actual retrieval on model levels used for rad. tra. calc.
ilev	ilev = number of model levels (for rad. trans. calculation)
itglev	itglev(imxgeo) number of the tangent-level for each geometry
ipar	ipar = number of altitudes where the temperature profile is fitted.

Module structure

The module proceeds along the following steps:

1. Building of the levels located between the lowest and the highest simulated geometries
2. building of the levels located above the highest simulated geometry,
3. interpolation of temperature and VMR profiles to the altitude levels generated in steps 1. and 2., determination of pressure at the generated levels,
4. calculation of itglev

Detailed description

For the description of this module, see par. 3.2.11.1.
The only difference between **mkplev_vmr** and **mkplev_fwd** modules is that the variable *iderlay* is not calculated in **mkplev_fwd**.

4.2.7.20 CHECK_VMR

For the description of this module, see Sect. 3.2.11.2.

4.2.7.21 POINT

For the description of this module, see par. 2.2.11.4.

4.2.7.22 CURGOD_FWD

```
CURGOD_FWD
    |(----DREFIND +
    |((---QSIMP5
          |-----DREFIND +
          |-----DFUNC1 >
```

```
|-----TRAPZ5
|       |(----PTNMRFROMZ
|       |       |-----DREFIND +
|       |(----DFUNC1 >
```

Description: This subroutine performs the ray-tracing for the different geometries and calculates,

1. for all the pairs geometry-layer:
- the column of all the gases (*rcol*) that have to be taken in account in the actual retrieval
- the air column (*raircol*)
- the length of the optical path (*ropath*) in the layer

2. and, only for a sub-set of the possible 'paths', the IAPT-numbers (see subroutine point):
- the equivalent pressure (in Curtis-Godson meaning) (*rpeq*) for all the gases (IAP)
- the equivalent temperature (*rteq*) for all the gases (IAT)

For some explanations of the reasons of the choices implemented in this module, refer to *T.N. on 'High Level algorithm definition and physical and mathematical optimisations' (TN-IROE-RSA9601),* sect. 6.1 and 6.2.

Variables exchanged with external modules

Name	Description
ipath	Total number of different IAPTs number
igeo	Total number of simulated geometries
ilev	Total number of atmospheric levels
itglev	Vector that associates to each geometry, the corresponding number of the tangent level.
ipoint	Matrix of IAPT-number
igas	Total number of gases in the selected MW
rpmod	rpmod(imxlev): pressure on levels used for the radiat. transf. calc.
rtmod	rtmod(imxlev): temperature on levels used for the radiat. transf. calc.
rzmod	rzmod(imxlev): heights of levels used for the radiat. tranf. calc.
rmrmod	rmrmod(imxlev,imxgas): volume mixing ratio for each gas considered in actual retrieval on levels used for rad. transf. calc.
rearad	earth radius
rlat	latitude of the actual limb-scan (deg.)
deps	degree of accuracy required for the calculation of Curtis-Godson integrals
rpeq	rpeq(imxpat,imxgas) implemented atmospheric (equivalent) pressures (IAPs)
rteq	rteq(imxpat,imxgas) implemented atmospheric (equivalent) temperatures (IATs)
rcol	rcol(imxlay,imxgeo,imxgas) columns for each layer, each geometry and each gas
raircol	raircol(imxlay,imxgeo) air-column for each layer and each geometry
ropath	ropath(imxlay,imxgeo) optical path lenght for each layer, each geometry
rtmain	rtmain(imxpat) Curtis-Godson equivalent temperature (IAT) of the main gas

Detailed description

This module performs the same operations that are made in module **curgod_pt** (see par. 2.2.11.5): the only difference is that all the perturbed quantities (*rpeqpert, rteqpert, rcolpert*) are not calculated in this module.

4.2.7.23 QSIMP5 & TRAPZ5

```
QSIMP5
    |-----DREFIND +
    |-----DFUNC1 >
    |-----TRAPZ5
    |       |(----SQRT *
    |       |(----PTNMRFROMZ
    |       |       |-----DREFIND +
    |       |(----DFUNC1 >
```

Description:
Starting from:

- the limits of integration *dxa* and *dxb*,
- the value of temperature, pressure (and consequently of refractive index) and VMR of the actual gas on the boundaries of the layer,
- the interpolation law in altitude of all these quantities inside the layer,

these two modules can calculate five different numerical integrals: *dcoll*, *dpl*, *dtl*, *daircoll* and *dopathl*. According to the value of the logical variable *lflag* some of them are not calculated.

Variables exchanged with external modules

Name	Description
dalay	altitude of the lower boundary of the layer
dxa	lower limit of integration
dblay	altitude of the higher boundary of the layer
dxb	higher limit of integration
dta	temperature corresponding to the lower boundary of the layer
dtb	temperature corresponding to the higher boundary of the layer
dpa	pressure corresponding to the lower boundary of the layer
dpb	pressure corresponding to the higher boundary of the layer
dmra	VMR corresponding to the lower boundary of the layer
dmrb	VMR corresponding to the higher boundary of the layer
dsnellc	Snell's law constant
dtan_0	tangent altitude referred to centre of the earth
rearad	earth radius
rlat	latitude
deps	required accuracy for the integrals calculation
dcoll	returned column of this path (to be moved to the choisen measurement units)
daircoll	returned air density (to be multiplied by parameter rk)
dopathl	returned path lenght (in km)
dtl	returned equivalent temperature (to be normalised)
dpl	returned equivalent pressure (to be normalised)
jgas	actual gas number (local code)

lflag	logical variable: only when it is true, the equivalent pressure and temperature have to be calculated

Detailed description:
For the description of this module, see par. 2.2.11.6.

4.2.7.24 DFUNC1

For the description of this module, see par. 2.2.11.7.

4.2.7.25 DLIM

For the description of this module, see par. 2.2.11.8.

4.2.7.26 DREFIND

For the description of this module, see par. 2.2.11.9.

4.2.7.27 PTNMRFROMZ

For the description of this module, see par. 2.2.11.10.

4.2.8 ADDNOISE

ADDNOISE
- |-----RAN3]
- |-----GASDEV >
- |-----RAN3]
- |-----GASDEV >
- |-----CONV_NOISE *

For the module description see source code in [AD7].

4.2.8.1 CONV_NOISE

Description
This module is used for performing the convolution of the noise, as a function of frequency, with the apodisation function.

Variables exchanged with external modules:

Name	Description
imw	index of the actual microwindow
igsim	index of the actual geometry
nsam	*nsam(imxmw)* = number of observed sampling points for the microwindow
rnoi	*rnoi(imxj)* = not-apodised noise spectrum
rnoic	*rnoic(imxi,imxmw,imxgeo)* = apodised noise, microwindow and geometry

	dependent
rapod_ sigma	**real *4** *rapod_sigma(imxilc)* = apodisation function in spectral domain
nailsdp	total number of points of the apodisation function

Algorithm Description
This module calculates the convolution integral between the input function *rnoi* and the apodisation function *rapod_sigma* as it comes from module **sapod**. The result of the convolution is calculated in a frequency interval that is reduced on both sides with respect to that of the input function. The frequency grid coincides with those of the observations.

Module Structure
1. Determination of the normalisation factor
2. Convolution between the input function and the apodisation function and normalisation.

Detailed Description
1. Determination of the normalisation factor
The summation *rsum* on all the spectral points *nailsdp* of the apodisation function *rapod_sigma* is calculated.

2. Convolution between the input function and the apodisation function, and normalisation.

The convolution integral is computed, at the i^{th} frequency as:

$$rnoic(i, imw, igsim) = \sum_{k=1}^{nailsdp} rnoi(k+i-1) \cdot rapod\_sigma(nailsdp-k+1)$$

where *k* is incremented by steps of 1.
The computation of *rnoic* is repeated for each value of *i* going from 1 to *nsam(imw).*
All the values of *rnoic* are normalized multiplying them by *1/rsum*.

4.2.9 ADDOFF
For the description of this module, see Sect. 2.2.28.

4.2.10 OUTESA
For the description of this module, see source code in [AD7].

4.2.11 OUTOBSERV
For this module description see source code in [AD7].

4.3 Variables and parameters used in the self-standing OFM

The parameters used in the self standing OFM are described in the following table.

Name	Description	Value
dcdop	used in Doppler broadening: sqrt(2 ln2 k avog / c^2)	3.5811737d-7
dext	extension of the (already with iadd*delta extended) microwindow where ioutin is set to 1 [cm^{-1}]	0.4
dinvpi	1/pi	0.318309886
dsqln2	sqrt(ln2)	0.832554611
dsqpi	sqrt(pi)	1.772453851
iqlclf	the quotient between coarse and fine wavenumber grid intervals	5
imxapo	maximum number of points in the apodisation function (path difference domain)	513
imxcof	max number of coefficients for the calculation of the quotient of the partition sum (=4)	4
imxcta	max number of elements in the correction table of tangent altitudes due to refraction index	50
imxept	max number of extra paths	1
imxfcs	max number of frequencies to which cross sections are provided in the look-up tables	1
imxfpg	max number of elements in the fixed P grid imposed to the retrieval	50
imxgas	max number of gas in the retrieval	10
imxgeo	max number of simulated observations	18
imxgmw	max number of gases per MW	4
imxhit	number of gases in the HITRAN 96 data base	36
imxi	maximum number of sampling points in the synthetic spectra computed at the observed frequencies	100
imxilc	max number of sampling point in the instrument line-shape function (course grid!)	1000
imxils	maximum number of sampling points in the instrument line-shape function (fine-grid!)	2400
imxism	max number of isotopes in HITRAN data base per molecule (=8)	8
imxiso	number of total isotopes in the HITRAN database	85
imxite	maximum number of macro-iterations in retrieval procedure	15
imxj	maximum dimension of J matrix ($VCMobs = J \cdot J^T$)	imxilc+imxi
imxlay	max number of layers for modelling the atmosphere (=imxlev-1)	imxlev-1
imxlev	max number of levels used for modelling the atmosphere	70
imxlin	max number of lines per microwindow	300
imxlmb	max number of parameters to be retrieved for each set of parameters (p,T,C,vmr)	18
imxmw	max number of microwindows	20
imxobs	max number of observational point (for Jacobian matrix)	2700
imxpat	max number of possible paths (be careful: imxpat*	imxlay+imxept*(i

	imxsig*4*imxgas is the number of bytes needed for the biggest field (variable rcross) in the program!)	mxgeo-1)
imxpcs	max number of P to which cross sections are provided in the look-up tables	1
imxpre	maximum number of points for the precalculated line shape	imxsig
imxpro	max number of elements in p, t profiles	100
imxri	max number of refraction indices provided in the corresponding file	50
imxsig	max number of wavenumber grid-points for a microwindow	5500
imxsl	max number of sub-levels between the pointings of the simulations	20
imxsnc	max number of sampling point for the sinc function used to interpolate the instrument line-shape function	4800
imxtcs	max number of T to which cross sections are provided in the look-up tables	1
imxtop	max number of parameters to be fitted	60
imxvt	max number of vibrational T provided in the corresponding file	20
nrepcode	HITRAN code for the gas for which the line shape is precalculated (HNO_3)	12
nrepiso	HITRAN isotope number of the gas for which the line shape is precalculated	1
rairmass	average molec. weigth of the air (kg/kmol) (US STD)	28.9644
rbc	Boltzmann constant (for density in mol/cm-3)	1.380658e-19
rc1	constant in the Planck-function (2 h c^2)	1.191043934e-3
rcn	constant in the refraction index expression (n=1.+(rcn*rt0n/rp0n)*p/T)	.000272632
rdmult	the number of Doppler half-widths from the line-centre from which the Lorentz function instead of the Voigt-function is used Error: rdmult=10 -> 1.5% ; rdmult=20 -> 0.4% ; rdmult=30 -> 0.18%	30.
refind	multiplicative constant in the expression of refraction index n: refind= rcn*rt0n/rp0n	rt0n*rcn/ rp0n
rephw0	reference half width of the line shape to be precalculated	0.11
repexph	reference half width exponent of the line to be precalculated	0.75
rg0	acceleration of gravity (m/s**2)	9.80665
rhck	h*c/k [K/cm-1]	1.4387687
rk	10^{-5}/rbc	10^{-5}/rbc
rmovr	1000 * rairmass / R(=8314.32[N.m/(kmol.K)])	3.483676
rp0h	reference pressure for pressure broadening	1013.25
rp0n	pressure on level sea for refraction index calculation	1013.25
rt0h	reference temperature for pressure broadening	296.
rt0int	reference temperature for the line intensity	296.
rt0n	temperature on level sea for refraction index calculation	288.16
rvlf	multiplier for (Doppler+Lorentz=~Voigt) half-width to determine the local fine grid	0.1
rvmult	rvmult is the number of (Doppler+Lorentz=~Voigt) half-widths from the line-centre where the transition between local coarse and	50

	local fine grid occurs (rvmult >= rdmult !!	

The variables used by the self standing OFM program and exchanged between modules are described in the follwing table.

Name	Dim-ension	Description	Modified in:
delta		distance between fine-wavenumber grid points [cm^{-1}]	input
deps		maximum relative variation for each iteration in calculation of curtis-godson variables	input
dsigm0		central frequency of the line used for testing P levels [cm^{-1}]	input
dsigma	imxsig, imxmw	wavenumber fine grid for each microwindow [cm^{-1}]	grid
dsilin	imxlin, imxmw	central wavenumber for each line of each Mw [cm^{-1}]	input
dstep		distance between coarse-wavenumber grid points [cm-1]	input
iadd		number of fine-wavenumber grid points to be added on both sides of each microwindow (due to the ils-convolution)	ails
ibase		number of base-levels	chbase
icode	imxlin, imxmw	HITRAN molecular code for each line of each Mw	input
iept		actual number of extra paths	input
ifspmw	imxmw	index of the first sampling point of each MW * NOTE: the sampling point at frequency=0 has index=1	input
igas		number of different gases for actual retrieval	inigas
igashi	imxgas	HITRAN code number for each global gas number	inigas
igasmw	imxmw	number of gases to be considered for each mw	inigas
igasnr	imxgas, imxmw	global gas number for the local gas number of each Mw	inigas
igeo		number of simulated geometries	occusim
iiso	imxlin, imxmw	isotope number for each line of each Mw.	input
ilev		number of levels for simulations	mkplev
ilimb		number of measured geometries	input
ilimbmw	imxmw	number of valid measured geometries per microwindow (number of 2 in each column of iocsim)	occusim
iline	imxmw	number of lines in each microwindow	input
imaingas		HITRAN code of the main gas of the retrieval (=2 for CO_2 in the case of p-T-retrieval)	input
imaskcont	3	imaskcont(1)=1: H_2O continuum calculation imaskcont(2)=1: O_2 continuum calculation imaskcont(3)=1: N_2 continuum calculation	input
imw		number of the actual microwindow	fwdmdl
iobs		total number of observations to be fitted	occusim
iocsim	imxgeo, imxmw	occupation matrix for the simulations to performed = 0 no simulation required, = 1 simulation required without FOV	occusim

		= 2 simulation required with FOV	
ioutin	imxlin, imxmw	flag for each line =1: line-shape has to be calculated at each wavenumber inside the microwindow =2: line is considered as nearby continuum	input
ipar		number of parameter-levels	occusim
ipath		number of different IAPT numbers in ipoint	point
ipoint	imxlay, imxgeo	matrix, which attaches to each pair of layer/geometry the IAPT number	point
ipro		number of elements contained in P, T and VMR profiles initial guess	input
irowmw	imxmw	the row of the Jacobian matrix where the actual microwindow starts	occusim
isigma	imxmw	number of general wavenumber fine grid points in each microwindow	grid
iterg		macro - iteration index (Gauss)	retr
iterm		micro - iteration index (Marquardt)	retr
itglev	imxgeo	number of the tangent-level for each geometry	mkplev
lfitgeo	imxgeo	logical vector that identify the levels where the profiles are fitted: referred to rzsi (to the simulated geometries)	occusim
lokku	imxgeo, imxmw	occupation matrix used for the selection of operational MW's for each observation geometry	input
lparbase	imxpro	logical vector that identify the levels where the profiles are fitted: referred to rzbase (to the base-levels)	chbase
nailsdp		number of AILS data points	input
napod	imxapo	number of points of rapod , the apodisation function in interferogram domain. IT HAS TO BE (2**n+1), WITH n INTEGER.	input
nils		number of elements of rils	ails
ninterpol		switch for the decision of interpolation of the absorption cross-sections for the geometries above the lowest geometry (only if the IAPT number of the path is increasing, which was decided during the calculation of ipoint) =-1: no interpolation, all cross-sections recalculated =0: all cross-sections above the lowest geometry are interpolated =1: new calculation only of the tangent-layer, all other layers interpolated =2: new calculation of the tangent-layer and the layer above, all others interpolated =3: ...	input
nll		number of basis vectors	read_look up_vmr
nrd		Ratio between general coarse grid step and general fine grid step	input
nsam	imxmw	number of sampling points in each MW (general coarse grid)	input
nselmw		total number of selected microwindows for the retrieval	input
nswh2oc		=imaskcont(1): switch for H_2O continuum	(input)

on		nswh2ocon=1: H_2O continuum considered	
nswn2con		=imaskcont(3): switch for N_2 continuum nswn2con=1: N_2 continuum considered	(input)
nswo2con		=imaskcont(2): switch for O_2 continuum nswo2con=1: O_2 continuum considered	(input)
nucl		nucl+1 = upper parameter level for continuum fit	retr
nvl		number of wavenumber points in cross-section look-up table	read_look up_vmr
rails	imxilc, imxmw	apodised instrument line shape for all selected MWs	input
raircol	imxlay, imxgeo	air-column for each layer and each geometry [moec/cm^{-2}]	curgod
rapod **real*4**	imxapo	apodisation function in path difference domain	input
rapod_sigma **real*4**	imxilc	apodisation function in spectral domain	input
rbase		greater base of trapezium of Field of View function [km]	input
rcbase	imxpro, imxmw	continuum on the base-levels for each MW [cm^2/molec]	chbase
rcol	imxlay, imxgeo, imxgas	column amounts for each layer, each geometry and each gas [molec/cm^2]	curgod
rconint	imxlmb ,imxmw	frequency range around each MW,in which the continuum can be considered as varying linearly. [cm^{-1}]	input
rconvc	3	thresholds used to check convergence criteria (see convchk-description)	input
rcprof	imxpro, imxmw	array containing continuum cross section as a function of altitude and microwindow [cm^2/molec]	input
rcross **real*4**	imxsig, imxpat, imxgmw	absorption cross sections for each general wavenumber fine grid point (1st index), each IAPT number (2nd index) and each gas (3rd index) for the actual Mw [cm^2/molec]	cross
rearad		local radius of curvature of the earth [km]	input
redfact		reduction factor applied to 'rincz' when it produces not acceptable P levels	input
relow	imxlin, imxmw	lower state energy for each line of each Mw [cm^{-1}]	input
rexph	imxlin, imxmw	exponent for temp. dependence of air-broadenedhalf width	input
rexphref		exponent for the calculation of Lorentz h-w in mkplev	input
rhw0	imxlin, imxmw	air broadened half width [cm^{-1}/atm] at 296 K	input
rhw0ref		half-width of the line used for testing P levels [cm^{-1}/atm] at 296 K	input
rhwvar		relative max. half-width variation allowed between two neighbouring P levels	input

rils	imxils, imxmw	instrument-line-shape function in the frequency fine grid	ails
rincz		trial increment given to altitude for building P levels	input
rint0	imxlin, imxmw	line intensity for each line of each Mw [$cm^{-1}/(molec*cm^{-2}$]	input
rintils		ratio between the frequency step approximating infinitesimal spectral resolution and the integral of the ILS function	ails
rlat		latitude of the actual limb-scan (deg.)	input
rlolin	imxlin, imxmw	lower limit where the line has to be considered [km]	input
rmaxtv1		max. allowed temp. variation between levels,when: 0 < altitude of level < rzt12 [K]	input
rmaxtv2		max. allowed temp. vatiation between levels, when: rzt12 < altitude of level < rulatm [K]	input
rmrmod	imxlev, imxgas	volume mixing ratio for each gas considered in actual retrieval on levels used for rad. tra. calc.	mkplev
rnoise	imxmw ,imxgeo	NESR dependent on geometry and microwindow	input
roffs	imxmw	fitted instrumental offset for each mw [r.u.]	input
ropath	imxlay, imxgeo	optical path length for each layer, each geometry [km]	curgod
rpbase	imxpro	pressure on the base-levels [hPa]	chbase
rpeq	imxpat, imxgas	equivalent pressures [hPa]	curgod
rpmod	imxlev	pressure on levels used for the radiat. transf. calc. [hPa]	mkplev
rpprof	imxpro	vector of pressure profile as a function of altitude Z. [hPa]	input
rsl		half-difference between the bases of the trapezium (1/rsl gives the slope) [km]	input
rspct	imxi, imxgeo	spectrum for each geometry on the general coarse grid [r.u.] 1st index: general wavenumber coarse grid 2nd index: geometries to be simulated for the actual Mw	spectrum
rspfov	imxi, imxgeo, imxmw	simulated spectra corresponding to the different tangent pressures and different microwindows on the general wavenumber coarse grid: (rspct * FOV) [r.u.]	fov, addoff
rtbase	imxpro	temperature of the base levels [K]	chbase
rteq	imxpat, imxgas	equivalent temperatures [K]	curgod
rtmain	imxpat	Curtis-Godson temperature of the main gas [K]	curgod
rtmod	imxlev	temperature on levels used for the radiat. transf. calc. [K]	mkplev
rtprof	imxpro	vector of temperature as a function of altitude Z. [km]	input
rulatm		upper limit of the atmosphere [km]	input
ruplin	imxlin, imxmw	upper limit where the line has to be considered [km]	input
rvmrbase	imxpro, imxgas	volume mixing ratio of the gases on the base levels [ppm]	chbase
rvmrprof	imxpro,	matrix of VMR profiles [ppm]	input

	imxgas		
rwmol	imxhit, imxism	molecular weight for each HITRAN molecular code and isotope number [g/mol]	wmol
rwmolref		molecular weight of the gas used for testing P levels [g/mol]	input
rzbase	imxpro	altitude of the base-levels [km]	chbase
rzerof		zero-filling expressed as the ratio between measured and transformed interferogram	input
rzmod	imxlev	heights of levels used for the radiat. tranf. calc. [km]	mkplev
rzmodpert	imxlev, imxlmb	perturbed altitude grids after the perturbation of temp. profiles. [km]	mkplev
rzprof	imxpro	vector of altitudes Z to which rtprof, rpprof and rvmrprof are referred [km]	input
rzsi	imxgeo	tangent altitudes of the geometries to be simulated [km]	occusim
rzt12		altitude where the temperature threshold changes from rmaxtv1 to rmaxtv2 [km]	input
rztang	imxgeo	vector containing the engineering values of tangent altitudes [km]	input
smw	imxmw	character*6 : microwindow identifier	