ESA ESRIN Contract No.: 17580/03/I-OL

'Support to MIPAS Level 2 Product Validation'

Modifications in MIPAS Level 2 processor for handling the new measurement scenario

IFAC_GA_2005_15_pr

Issue 1

24 February 2006

Prepared by:

Name	Institute
Claudio Belotti	IFAC-CNR
Simone Ceccherini	IFAC-CNR
Piera Raspollini	IFAC-CNR

Checked by:

Name	Institute
Bruno Carli	IFAC-CNR

CONTENTS

1 Reference documents

[RD1] TN-IROE-RSA9601, Issue: 3
Title: High level algorithm definition and physical and mathematical optimisations

[RD2] TN-IROE-RSA9603, Issue: 3A
Title: Software Architecture and Algorithm Definition

[RD3] TN-IROE-GS98-11, 'High level description of the new functionalities to be implemented in MIPAS Level 2 processor'

[RD4] TN-IROE-GS0102, Issue: 1 - Revision: 1
Title: Pre-flight modifications to the ORM_ABC code (Apr. 2001)

[RD5] 'New functionalities implemented in ORM_ABC_1.2.3 and cloud detection algorithm', Draft (Nov 2001)

[RD6] 'After launch ORM_ABC bug correction', Draft, Revision 1, IFAC_GA_2004_07_pr, 11 March 2004.

[RD7] S. Ceccherini, 'Analytical determination of the regularization parameter in the retrieval of atmospheric vertical profiles', Optics Letters, 30, 2554-2556 (2005)

[RD8] TN IFAC_GA_2005_10_SC, 'Considerations on the different regularization methods for the retrieval of MIPAS new observation scenario measurements', Draft (July 2005)

[RD9] PO-TN-ULE-GS-0001, 'Cloud detection for MIPAS: an initial feasibility study and technical specifications for a cloud detection algorithm', November 2001

[RD10] PO-TN-ULE-GS-0002, 'Detection of cloud effects in MIPAS observations and implementation in the operational processor', 2004

2 Introduction

The Optimized Retrieval Model (ORM), developed in the frame of ESA contract 11717/95/NL/CN, is the scientific reference for the Retrieval Component Library of the MIPAS Level 2 NRT processor. In particular, Version 1.2.3 of the ORM_ABC code (described in [RD1], [RD2], [RD3], [RD4] and [RD5]) is the reference of Level 2 processor updated just after the launch.
The Level 2 processor includes also a pre-processor that does not have a corresponding scientific code.
While only minor upgrades in Level 2 processor were necessary after the launch, involving only the pre-processor (cloud filtering algorithm) and the retrieval environment given by the

auxiliary data, a more significant change in the code is now required as a consequence of the change in the measurement scenario decided after MIPAS major hardware problems that required a reduction in the spectral resolution (0.0625 cm^{-1} instead of 0.025 cm^{-1}). In order to exploit the reduction in measurement time coming from the reduced resolution measurements, a change in the measurement scenario consisting in a finer measurement grid in the troposphere and lower stratosphere (1.5 km step instead of 3 km step) was recommended by the MIPAS Science Team. The use of a measurement grid significantly finer than the FOV extension may improve the vertical resolution of the retrieved profile but, because of the resulting ill-conditioning of the retrieval, requires a regularisation of the profile.

These being the major changes, a few minor modifications are also considered. The minor modifications in the Level 2 processor involve the computation of the conditioning parameter of the matrix that has to be inverted for monitoring the stability of the inversion, some bug corrections, new parameters for the partitioning function for HNO3, modifications in Level 2 pre-processor, an additional module for a better definition of the altitude grid obtained exploiting ECMWF data and changes in the format of the output file.

In this TN the modifications are divided in 2 groups: the first group refers to the changes in the ORM that affect the Level 2 Retrieval Component Library. For these modifications a detailed description of the algorithm changes are provided, with indication of the changes of the affected routines of the scientific code. The described modifications have been included in version 2.0 of the ORM_ABC.
The second group involves modifications in the Level 2 pre-processor and some post-processing that hence do not affect the ORM.
For this second set of modifications only a high level description of the algorithm to be implemented is provided.

3 Description of the modifications in the ORM

The modifications that affect the ORM are:

1. a-posteriori regularisation
2. computation of conditioning parameter
3. bug correction in continuum derivative computation
4. bug correction in temperature/vmr derivative computation
5. update of partitioning function for HNO3

3.1 A-posteriori regularisation

Different options of regularisation were analysed in [RD8], and the conclusion was to implement an a-posteriori regularization. This choice was approved by the QWG during the QWG Meeting #8.
The a-posteriori regularization is obtained starting from the non-regularized profile $\hat{\mathbf{x}}$ produced by the original code, characterized by its variance-covariance matrix (VCM) $\mathbf{S}_{\hat{\mathbf{x}}}$ and its averaging kernel matrix (AKM) $\mathbf{A}_{\hat{\mathbf{x}}}$, and minimizing the following cost function:

$$f(\mathbf{x}) = (\mathbf{x} - \hat{\mathbf{x}})^T \mathbf{S}_{\hat{\mathbf{x}}}^{-1} (\mathbf{x} - \hat{\mathbf{x}}) + \lambda \mathbf{x}^T \mathbf{R} \mathbf{x}$$

where λ is the regularization parameter and **R** is the regularization matrix (equal to $\mathbf{L}_1^T\mathbf{L}_1$, with $\mathbf{L}_1$ the first derivative matrix).
The input quantities $\mathbf{S}_{\hat{\mathbf{x}}}$ and $\mathbf{A}_{\hat{\mathbf{x}}}$ are given by:

$$\mathbf{S}_{\hat{\mathbf{x}}} = \left(\mathbf{S}^{-1} + \alpha\mathbf{M}\right)^{-1}\mathbf{S}^{-1}\left(\mathbf{S}^{-1} + \alpha\mathbf{M}\right)^{-1}$$
$$\mathbf{A}_{\hat{\mathbf{x}}} = \left(\mathbf{S}^{-1} + \alpha\mathbf{M}\right)^{-1}\mathbf{S}^{-1}$$

where $\mathbf{S}=(\mathbf{K}^T\mathbf{S}_y^{-1}\mathbf{K})^{-1}$ (with **K** the Jacobian matrix of the forward model and $\mathbf{S}_y$ the VCM of the observation vector), α is the Marquardt parameter and **M** is the Marquardt matrix that, in the case of the MIPAS retrieval code, has all the elements equal to zero except those on the diagonal that are equal to the diagonal elements of the matrix $\mathbf{S}^{-1}$**.**
The minimum of f(**x**) corresponds to the solution:

$$\mathbf{x} = \left(\mathbf{S}_{\hat{\mathbf{x}}}^{-1} + \lambda\mathbf{R}\right)^{-1}\mathbf{S}_{\hat{\mathbf{x}}}^{-1}\hat{\mathbf{x}}$$

with the VCM and AKM given by:

$$\mathbf{S}_{\mathbf{x}} = \left(\mathbf{S}_{\hat{\mathbf{x}}}^{-1} + \lambda\mathbf{R}\right)^{-1}\mathbf{S}_{\hat{\mathbf{x}}}^{-1}\left(\mathbf{S}_{\hat{\mathbf{x}}}^{-1} + \lambda\mathbf{R}\right)^{-1}$$
$$\mathbf{A}_{\mathbf{x}} = \left(\mathbf{S}_{\hat{\mathbf{x}}}^{-1} + \lambda\mathbf{R}\right)^{-1}\mathbf{S}_{\hat{\mathbf{x}}}^{-1}\mathbf{A}_{\hat{\mathbf{x}}}$$

The value of the regularization parameter λ can be determined using the error consistency method [RD7] that provides the following analytical solution for the regularization parameter λ:

$$\lambda = \sqrt{\frac{n}{\hat{\mathbf{x}}^T\mathbf{R}\mathbf{S}_{\hat{\mathbf{x}}}\mathbf{R}\hat{\mathbf{x}}}}$$

Where n is the number of points of the retrieved profile.
In the case of water vapour the a-posteriori regularization is performed on the logarithm of the profile [RD8].
In this case the cost function is defined for the logarithm of the profile, the minimum of the cost function is obtained for:

$$\log(\mathbf{x}) = \left(\mathbf{S}_{\log(\hat{\mathbf{x}})}^{-1} + \lambda\mathbf{R}\right)^{-1}\mathbf{S}_{\log(\hat{\mathbf{x}})}^{-1}\log(\hat{\mathbf{x}})$$

where:

$$\left(\mathbf{S}_{\log(\hat{\mathbf{x}})}\right)_{i,j} = \frac{\left(\mathbf{S}_{\hat{\mathbf{x}}}\right)_{i,j}}{\hat{x}_i\hat{x}_j}$$

The regularized profile is obtained by means of the exponential function:

$$\mathbf{x} = \exp(\log(\mathbf{x})) = \exp\left[\left(\mathbf{S}_{\log(\hat{\mathbf{x}})}^{-1} + \lambda\mathbf{R}\right)^{-1}\mathbf{S}_{\log(\hat{\mathbf{x}})}^{-1}\log(\hat{\mathbf{x}})\right]$$

and is characterized by the following VCM and AKM:

$$(\mathbf{S}_{\mathbf{x}})_{i,j} = \left[\left(\mathbf{S}_{\log(\hat{\mathbf{x}})}^{-1} + \lambda\mathbf{R}\right)^{-1}\mathbf{S}_{\log(\hat{\mathbf{x}})}^{-1}\left(\mathbf{S}_{\log(\hat{\mathbf{x}})}^{-1} + \lambda\mathbf{R}\right)^{-1}\right]_{i,j} x_i x_j$$

$$\mathbf{A}_{\mathbf{x}} = \mathbf{C}\mathbf{A}_{\hat{\mathbf{x}}}$$

where **C** is defined by:

$$(\mathbf{C})_{i,j} = \left[\left(\mathbf{S}_{\log(\hat{\mathbf{x}})}^{-1} + \lambda\mathbf{R}\right)^{-1}\mathbf{S}_{\log(\hat{\mathbf{x}})}^{-1}\right]_{i,j}\frac{x_i}{\hat{x}_j}$$

The value of the regularization parameter λ can be determined by means of the error consistency method applied to the logarithm of the profile, obtaining:

$$\lambda = \sqrt{\frac{n}{\log(\hat{\mathbf{x}})^T\mathbf{R}\mathbf{S}_{\log(\hat{\mathbf{x}})}\mathbf{R}\log(\hat{\mathbf{x}})}}$$

The a-posteriori regularization is applied to the profiles of temperature, O_3, HNO_3, CH_4, N_2O, NO_2 and to the logarithm of the profile of H_2O. The a-posteriori regularization is not applied to pressure profile and to continuum and offset parameters.

3.1.1 I/O modifications

3.1.1.1 Input files

Concerning ML2PP, the integer variable used for switching the old regularisation can be used to select the type of regularization in the following way:

Value	Type of regularization
0	no regularization
1	old regularization
2	new a-posteriori regularization

Concerning ORM, the logical variable *lifreg* contained in the file settings_*.dat has been transformed into the integer variable *ireg*.
This variable is used to select the type of regularisation according to the table above.

3.1.1.2 Output files

The new product **x** takes the place of $\hat{\mathbf{x}}$. The new product $\mathbf{S}_{\mathbf{x}}$ takes the place of $\mathbf{S}_{\hat{\mathbf{x}}}$. The new product $\mathbf{A}_{\mathbf{x}}$ requires a new space in the output files.
The AKM *rak_reg* ((itop)*(itop)) of the regularized profile has to be added.

3.1.2 Code modifications

3.1.2.1 Routine retr_pt

Once that the convergence has been reached, and the program has exited from the iterations, the following additional operations are made:

- If *ireg* is equal to 2 (i. e., the a-posteriori regularization is selected in the input files)
    - A call to the subroutine regularize_ecam_pt (described in section 3.1.2.3) is performed
    - The vector of parameters *rxpar* is replaced by *rxpar_reg*:

      $$rxpar(i) = rxpar\_reg(i) \qquad i = 1,...,itop$$

    - A call to the subroutine updprof_pt is performed
- End if on *ireg*

After the call to ainvcal_pt that the code uses to re-calculate the inverse of the matrix *ra_bkp* (that is only used for reporting the noise error in the output files) the following additional operations are made:

- If *ireg* is equal to 2 (i. e., the a-posteriori regularization is selected in the input files)
    - The matrix *rainv* is replaced by *rvcm_reg*:

      $$rainv(i,j) = rvcm\_reg(i,j) \qquad i = 1,...,itop \quad j = 1,...,itop$$

- End if on *ireg*

3.1.2.2 Routine retr_vmr

After the exit from the iterations the following additional operations are made:

- If *ireg* is equal to 2 (i. e., the a-posteriori regularization is selected in the input files)
    - If *im* is equal to 1 (i. e., the code is performing the retrieval of H_2O)
        - A call to the subroutine regularize_ecam_h2o is performed

- Else (i. e., the code is performing the retrieval of a species different from H2O)
  - A call to the subroutine regularize_ecam_vmr is performed
- End if on *im*
- The vector of parameters *rxpar* is replaced by *rxpar_reg*:

  $rxpar(i) = rxpar\_reg(i) \qquad i = 1,...,itop$

- A call to the subroutine updprof_vmr is performed

- End if on *ireg*

After the call to ainvcal_vmr that the code uses to re-calculate the inverse of the matrix *ra_bkp* (that is only used for reporting the noise error in the output files) the following additional operations are made:

- If *ireg* is equal to 2 (i.e ., the a-posteriori regularization is selected in the input files)
  - The matrix *rainv* is replaced by *rvcm_reg*:

    $rainv(i,j) = rvcm\_reg(i,j) \qquad i = 1,...,itop \quad j = 1,...,itop$

- End if on *ireg*

3.1.2.3 Routine regularize_ecam_pt

This is a new subroutine.

3.1.2.3.1 Exchanged variables

Name	Description	I/O	Old/new
rxpar	vector of the fitted parameters	input	old
rzpar	altitudes where the temperature profile is retrieved	input	old
ra	matrix A	input	old
rainv	inverse of matrix A	input	old
ra_bkp	back-up of matrix A	input	old
ipar	number of fitted points of the temperature profile	input	old
ilimb	number of valid measured geometries	input	old
itop	total number of fitted parameters	input	old

rxpar_reg	regularized vector of the fitted parameters	output	new
rvcm_reg	VCM of *rxpar_reg*	output	new
rak_reg	AKM of *rxpar_reg*	output	new

3.1.2.3.2 Description of the code

- The first derivatives matrix *rl1* ((ipar-1)*(ipar)) is defined:

$$\begin{pmatrix} \frac{1}{rzpar(1)-rzpar(2)} & -\frac{1}{rzpar(1)-rzpar(2)} & \dots & 0 & 0 \\ 0 & \frac{1}{rzpar(2)-rzpar(3)} & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & \frac{1}{rzpar(ipar-2)-rzpar(ipar-1)} & 0 \\ 0 & 0 & \dots & \frac{1}{rzpar(ipar-1)-rzpar(ipar)} & -\frac{1}{rzpar(ipar-1)-rzpar(ipar)} \end{pmatrix}$$

- The regularization matrix *rr* ((ipar)*(ipar)) is calculated.

$$rr = rl1^T \cdot rl1$$

- The AKM *rak_in* ((itop)*(itop)) before of the regularization is calculated:

$$rak\_in = rainv \cdot ra\_bkp$$

- The VCM *rvcm_in* ((itop)*(itop)) before of the regularization is calculated:

$$rvcm\_in = rak\_in \cdot rainv$$

- The matrix *rs* ((ipar)*(ipar)) corresponding to the VCM of temperature is extracted by *rvcm_in*:

$$rs(i,j) = rvcm\_in(ilimb+i, ilimb+j) \qquad i = 1 \dots ipar, j = 1 \dots ipar$$

- The matrix *rrsr* ((ipar)*(ipar)) is calculated:

$$rrsr = rr \cdot rs \cdot rr$$

- The scalar *rden* is calculated:

$$rden = \sum_{i,j=1}^{ipar} rxpar(ilimb+i) * rrsr(i,j) * rxpar(ilimb+j)$$

- The value of the regularization parameter *rlambdatik* is calculated by means of the error consistency method:

$$rlambdatik = dsqrt(ipar / rden)$$

- The inverse of the matrix *rvcm_in* is calculated by means of the subroutine *ainvcal_pt* and the result is called *rvcm_ininv* ((itop)*(itop)).

- The matrix *ra_reg* ((itop)*(itop)) is defined equal to *rvcm_ininv* for all the elements except the block corresponding to the temperature for which the Tikonov term is added:

$$ra\_reg = rvcm\_ininv$$

$$ra\_reg(ilimb + i, ilimb + j) = rvcm\_ininv(ilimb + i, ilimb + j) + rlambdatik \cdot rr(i, j)$$
$$i, j = 1,2,...,ipar$$

- The inverse of the matrix *ra_reg* is calculated by means of the subroutine *ainvcal_pt* and the result is called *ra_reginv* ((itop)*(itop)).

- The matrix *rtemp* ((itop)*(itop)) is defined:

$$rtemp = ra\_reginv \cdot rvcm\_ininv$$

- The regularized profile *rxpar_reg* ((itop)*(1))is calculated:

$$rxpar\_reg = rtemp \cdot rxpar$$

- The total AKM of the regularized profile is calculated:

$$rak\_reg = rtemp \cdot rak\_in$$

- The VCM *rvcm_reg* ((itop)*(itop)) of the regularized profile is calculated:

$$rvcm\_reg = rtemp \cdot ra\_reginv$$

3.1.2.4 Routine regularize_ecam_h2o

This is a new subroutine.

3.1.2.4.1 Exchanged variables

Name	Description	I/O	Old/new
rxpar	vector of the fitted parameters	input	old
rzpar	altitudes where the VMR profile is retrieved	input	old
ra	matrix A	input	old
rainv	inverse of matrix A	input	old
ra_bkp	back-up of matrix A	input	old
ipar	number of fitted points of the VMR profile	input	old
itop	total number of fitted parameters	input	old
rxpar_reg	regularized vector of the fitted parameters	output	new
rvcm_reg	VCM of *rxpar_reg*	output	new
rak_reg	AKM of *rxpar_reg*	output	new

3.1.2.4.2 Description of the code

- The first derivatives matrix *rl1* ((ipar-1)x(ipar)) is defined:

$$\begin{pmatrix} \frac{1}{rzpar(1)-rzpar(2)} & -\frac{1}{rzpar(1)-rzpar(2)} & \dots & 0 & 0 \\ 0 & \frac{1}{rzpar(2)-rzpar(3)} & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & \frac{1}{rzpar(ipar-2)-rzpar(ipar-1)} & 0 \\ 0 & 0 & \dots & \frac{1}{rzpar(ipar-1)-rzpar(ipar)} & -\frac{1}{rzpar(ipar-1)-rzpar(ipar)} \end{pmatrix}$$

- The regularization matrix *rr* ((ipar)*(ipar)) is calculated.

$$rr = rl1^{T} \cdot rl1$$

- The AKM *rak_in* ((itop)*(itop)) before of the regularization is calculated:

$$rak\_in = rainv \cdot ra\_bkp$$

- The VCM *rvcm_in* ((itop)*(itop)) before of the regularization is calculated:

$$rvcm\_in = rak\_in \cdot rainv$$

- A back-up of *rxpar* is put in *rxpar_orig* ((itop)*(1)).

- The logarithm of the VMR profile is performed and the result is substituted in the first *ipar* components of *rxpar*:

$$rxpar(i) = \mathrm{d}\log(rxpar(i)) \qquad i = 1,2,...,ipar$$

- The new VCM of *rxpar* is calculated taking into account that the logarithm has been applied only to the first *ipar* components of *rxpar*. The result is put in *rvcm_in*:

$$rvcm\_in(i,j) = rvcm\_in(i,j) / rxpar\_orig(i) \qquad i = 1,...,ipar \quad j = 1,...,itop$$

$$rvcm\_in(i,j) = rvcm\_in(i,j) / rxpar\_orig(j) \qquad i = 1,...,itop \quad j = 1,...,ipar$$

- The matrix *rs* ((ipar)*(ipar)) corresponding to the VCM of the logarithm of the VMR is extracted by *rvcm_in*:

$$rs(i,j) = rvcm\_in(i,j) \qquad i = 1 \ldots ipar, j = 1 \ldots ipar$$

- The matrix *rrsr* ((ipar)*(ipar)) is calculated:

$$rrsr = rr \cdot rs \cdot rr$$

- The scalar *rden* is calculated:

$$rden = \sum_{i,j=1}^{ipar} rxpar(i) * rrsr(i,j) * rxpar(j)$$

- The value of the regularization parameter *rlambdatik* is calculated by means of the error consistency method:

$$rlambdatik = dsqrt(ipar / rden)$$

- The inverse of the matrix *rvcm_in* is calculated by means of the subroutine *ainvcal_vmr* and the result is called *rvcm_ininv* ((itop)*(itop)).

- The matrix *ra_reg* ((itop)*(itop)) is defined equal to *rvcm_ininv* for all the elements except the block corresponding to the logarithm of the VMR for which the Tikonov term is added:

$$ra\_reg = rvcm\_ininv$$

$$ra\_reg(i,j) = rvcm\_ininv(i,j) + rlambdatik \cdot rr(i,j)$$
$$i, j = 1,2,...,ipar$$

- The inverse of the matrix *ra_reg* is calculated by means of the subroutine *ainvcal_vmr* and the result is called *ra_reginv* ((itop)*(itop)).

- The matrix *rtemp* ((itop)*(itop)) is defined:

$$rtemp = ra\_reginv \cdot rvcm\_ininv$$

- The regularized parameter vector *rxpar_reg* ((itop)*(1))is calculated:

$$rxpar\_reg = rtemp \cdot rxpar$$

- The exp function of the first *ipar* components of *rxpar_reg* is performed in order to obtain the regularised profile of volume mixing ratio:

$$rxpar\_reg(i) = \mathrm{d}\exp(rxpar\_reg(i)) \qquad i = 1,2,...,ipar$$

- The matrix *rtemp1* ((itop)*(itop)) is calculated:

$$rtemp1(i,j) = rtemp(i,j) \qquad i = 1,...,itop \quad j = 1,...,itop$$

$$rtemp1(i,j) = rtemp1(i,j) \cdot rxpar\_reg(i) \qquad i = 1,...,ipar \quad j = 1,...,itop$$

$$rtemp1(i,j) = rtemp1(i,j) / rxpar\_orig(j) \qquad i = 1,...,itop \quad j = 1,...,ipar$$

- The AKM of the regularized profile is calculated:

$$rak\_reg = rtemp1 \cdot rak\_in$$

- The VCM *rvcm_reg* ((itop)*(itop)) of the regularized profile is calculated:

$$rvcm\_reg = rtemp \cdot ra\_reginv$$

$$rvcm\_reg(i,j) = rvcm\_reg(i,j) \cdot rxpar\_reg(i) \qquad i = 1,...,ipar \quad j = 1,...,itop$$

$$rvcm\_reg(i,j) = rvcm\_reg(i,j) \cdot rxpar\_reg(j) \qquad i = 1,...,itop \quad j = 1,...,ipar$$

3.1.2.5 Routine regularize_ecam_vmr

This is a new subroutine.

3.1.2.5.1 Exchanged variables

name	Description	I/O	Old/new
rxpar	vector of the fitted parameters	input	old
rzpar	altitudes where the VMR profile is retrieved	input	old
ra	matrix A	input	old
rainv	inverse of matrix A	input	old
ra_bkp	back-up of matrix A	input	old
ipar	number of fitted points of the VMR profile	input	old
itop	total number of fitted parameters	input	old
rxpar_reg	regularized vector of the fitted parameters	output	new
rvcm_reg	VCM of *rxpar_reg*	output	new
rak_reg	AKM of *rxpar_reg*	output	new

3.1.2.5.2 Description of the code

- The first derivatives matrix *rl1* ((ipar-1)x(ipar)) is defined:

$$\begin{pmatrix} \frac{1}{rzpar(1)-rzpar(2)} & -\frac{1}{rzpar(1)-rzpar(2)} & \dots & 0 & 0 \\ 0 & \frac{1}{rzpar(2)-rzpar(3)} & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & \frac{1}{rzpar(ipar-2)-rzpar(ipar-1)} & 0 \\ 0 & 0 & \dots & \frac{1}{rzpar(ipar-1)-rzpar(ipar)} & -\frac{1}{rzpar(ipar-1)-rzpar(ipar)} \end{pmatrix}$$

- The regularization matrix *rr* ((ipar)*(ipar)) is calculated.

$$rr = rl1^{T} \cdot rl1$$

- The AKM *rak_in* ((itop)*(itop)) before of the regularization is calculated:

$$rak\_in = rainv \cdot ra\_bkp$$

- The VCM *rvcm_in* ((itop)*(itop)) before of the regularization is calculated:

$$rvcm\_in = rak\_in \cdot rainv$$

- The matrix *rs* ((ipar)*(ipar)) corresponding to the VCM of volume mixing ratio (VMR) is extracted by *rvcm_in*:

$$rs(i,j) = rvcm\_in(i,j) \qquad i = 1 \ldots ipar, j = 1 \ldots ipar$$

- The matrix *rrsr* ((ipar)*(ipar)) is calculated:

$$rrsr = rr \cdot rs \cdot rr$$

- The scalar *rden* is calculated:

$$rden = \sum_{i,j=1}^{ipar} rxpar(i) * rrsr(i,j) * rxpar(j)$$

- The value of the regularization parameter *rlambdatik* is calculated by means of the error consistency method:

$$rlambdatik = dsqrt(ipar / rden)$$

- The inverse of the matrix *rvcm_in* is calculated by means of the subroutine *ainvcal_vmr* and the result is called *rvcm_ininv* ((itop)*(itop)).

- The matrix *ra_reg* ((itop)*(itop)) is defined equal to *rvcm_ininv* for all the elements except the block corresponding to the VMR for which the Tikonov term is added:

$$ra\_reg = rvcm\_ininv$$

$$ra\_reg(i,j) = rvcm\_ininv(i,j) + rlambdatik \cdot rr(i,j)$$
$$i, j = 1,2,\ldots,ipar$$

- The inverse of the matrix *ra_reg* is calculated by means of the subroutine *ainvcal_vmr* and the result is called *ra_reginv* ((itop)*(itop)).

- The matrix *rtemp* ((itop)*(itop)) is defined:

$$rtemp = ra\_reginv \cdot rvcm\_ininv$$

- The regularized profile *rxpar_reg* ((itop)*(1))is calculated:

$$rxpar\_reg = rtemp \cdot rxpar$$

- The AKM of the regularized profile is calculated:

$$rak\_reg = rtemp \cdot rak\_in$$

- The VCM *rvcm_reg* ((itop)*(itop))of the regularized profile is calculated:

$$rvcm\_reg = rtemp \cdot ra\_reginv$$

3.2 Conditioning parameter

The conditioning parameter of a matrix is the ratio between the largest and the smallest eigenvalue of the matrix, and it provides information on how much the matrix is close to be singular. Large values of the conditioning parameter tell us that the matrix is nearly singular and its inversion can cause instabilities in the results.

3.2.1 I/O modifications

3.2.1.1 Output files

The value of the conditioning parameter *rcond* of the matrix *ra* calculated at the last iteration has to be written in the output files.

3.2.2 Code modifications

3.2.2.1 Routine ainvcal_pt

The new variable *rcond* has been inserted between the exchanged variables.

After the call to the subroutine jacobi_pt the following additional operations are made:

- the largest and the smallest eigenvalue of *ra* have been selected
- the ratio *rcond* between the largest and the smallest eigenvalue of *ra* has been calculated

3.2.2.2 Routine ainvcal_vmr

The new variable *rcond* has been inserted between the exchanged variables.

After the call to the subroutine jacobi_vmr the following additional operations are made:

- the largest and the smallest eigenvalue of *ra* have been selected
- the ratio *rcond* between the largest and the smallest eigenvalue of *ra* has been calculated

3.2.2.3 Routine retr_pt

The calls to the subroutine ainvcal_pt have been modified inserting the variable *rcond* among the exchanged variables

3.2.2.4 Routine retr_vmr

The calls to the subroutine ainvcal_vmr have been modified inserting the variable *rcond* among the exchanged variables

3.3 Bug correction in continuum derivatives

This modification is meant to eliminate a bug in the algorithm occurring in very particular cases. The description of this modification is performed in [RD6]. However for the benefit of the reader the information needed to understand the bug and to change the code is herewith recalled.
The bug involves the computation of the derivatives of the spectrum with respect to the fitted continuum values. These are computed as the product of

a) the derivative of the spectrum with respect to the mean value of the continuum in each layer (computed in routine **spectrum_**)
b) the derivative of the mean continuum value in each layer with respect to the continuum values on the levels of the base grid (computed in routine **conlay_**)
c) the derivative of the continuum values on the base grid with respect to the fitted continuum parameters (computed by routine **ficarra_**).

The computation of these derivatives is performed with the use of some additional matrices: *igeocder(imxgeo,3)* indicating, for each simulated geometry, the highest and the lowest levels in the base grid for which continuum derivatives have to be computed; *iderlayc(imxpro,2)* indicating, for each level in the base grid, the highest and lowest layer affected by a change in the base grid level.

The range where continuum profile is fitted is defined by two settings variables:

- *rucl* : indicating the altitude below which atmospheric continuum is fitted
- *rzc0*: indicating the altitude above which atmospheric continuum is set to 0.

Between *rucl* and *rzc0* the atmospheric continuum is obtained scaling the initial guess profile by the same factor applied to the highest continuum value at the previous iteration to get the new estimate for the current iteration.

The variable *nucl1* indicates the highest level of the base grid with respect to which derivatives of type b) have to be computed. In theory, *nucl1* should be equal to *nucl2* which is the highest level of the base grid whose altitude is smaller than *rzc0*. In practice, it is sufficient to use a value of *nucl1* equal to max(*nucl2*, *nucl0-5*), where *nucl0* is the highest level of the base grid whose altitude is smaller than *rucl*. This value "5" represents the number of levels to be considered above the level corresponding to the highest fitted value for continuum derivatives calculation and is a compromise allowing good accuracy continuum derivatives without any waste in computing time.

The variable *nucl1* is computed at the beginning of the retrieval program (iteration 0) using the setting parameters *rucl* and *rzc0* and the base grid profile defined at the beginning (routine **chbase**_). This procedure is correct in the VMR retrievals, since the base grid does not change during the iterations, but could be wrong in the case of pT retrievals, since the number of the base grid levels can change from iteration to iteration.

If the number of the base grid levels changes significantly after the iteration 0, the fact that the variable *nucl1* is not re-computed at each iteration is a bug which may either be the cause of a crash (if at a given iteration the total number of base grid levels becomes smaller than *nucl1*), or can affect the accuracy of the continuum derivatives (especially the ones relating to the high altitudes of the continuum profile).
It has to be noted that the bug is more likely to cause a crash in case that the retrieval is performed excluding low tangent altitude measurements from the analysis i.e. in case of clouds or in presence of corrupted sweeps at low altitudes. In fact, in this case, since the continuum is currently fitted only up to 20 km (*rucl* = 20 km), it is also possible that no continuum values need to be fitted.
Actually, in the code no checks are done to verify if any continuum parameters are fitted before the set-up of the auxiliary variables needed for continuum derivatives calculation. Furthermore, the computation of some continuum derivatives is performed, even if they are not used. We propose to change the code in order to check if any continuum parameters are fitted or not. In the latter case no continuum derivatives are performed.

For a better understanding of the proposed modifications it is useful to recall the current code operations: *nucl1* is computed at the iteration 0 using the input parameters *rucl* and *rzc0* and the base profile relating to iteration 0.
nucl1 is used in routine **tcgeo** to define, at each iteration, the variable *igeocder*.
The variable *igeocder* is used in routine **spectrum** to compute the variable *rspctcder* (derivative of type a)). For the simulated geometry *jgeo* and for each level of the base profile *jbase* between *igeocder(jgeo,1)* and *igeocder(jgeo,2)* (computed at each iteration in pT retrieval, but with the non-updated *nucl1* value) the layers affected by *jbase* are given by *iderlayc(jbase,1→2)* (computed at each iteration in pT retrieval).
The bug occurs when *igeocder(jgeo,1)=nucl1* is greater than the total number of base levels *(ibase)*.
In this case the program looks for an *iderlayc(nucl1,1)* that has not been re-computed at the current iteration and that hence assumes an arbitrary value.

3.3.1 Code modifications

Module **updprof_pt**

- Changed interface: added variables *rucl* and *rzc0*.
- Changed interface with module **tcgeo_pt** (added variable *icontpar*).
- Added computation of *nucl1* before the call to module **tcgeo_pt**. *nucl1* indicates the highest level of the base grid with respect to which continuum derivatives of type (b) have to be computed.
  nucl1 is set equal to max(*nucl2*, *nucl0-5*), where *nucl0* is the highest level of the base grid whose altitude is smaller than *rucl*
  ****
  Definition of the variable *nucl1*:

```
do i=ibase,1,-1
if (rzbase(i) .lt. rzc0)then
        nucl2=i
end if
end do

do i=ibase,1,-1
if (rzbase(i) .le. rucl)then
        nucl1=i
end if
end do

nucl1=max(nucl1-5,nucl2)
```

****

In order to avoid unnecessary computations in case that no continuum parameters have to be fitted, the following modules need to be changed:

Module **guesspar_pt**:

- The module **ficarra_pt** is called only if some continuum parameters have to be fitted (i.e. if *icontpar* different from 0)

```
If (icontpar.ne. 0) call ficarra_pt()
```

Module **tcgeo_pt**:

- In case that no continuum parameters have to be fitted, the variable *igeocder* is set to appropriate values to skip the computation of continuum derivatives.

```
if (icontpar.eq.0) then
do jgeo=1, igeo
 igeocder(jgeo,1)=imxpro
```

```
  igeocder(jgeo,2)=imxpro-1

  igeocder(jgeo,3)=imxpro-2
  End do
 Else
  <Normal computation of variable igeocder>
End if
```

- Added *icontpar* at the interface

Module **mkplev_pt**:

- Added variable *icontpar* at the interface
- If no continuum values have to be fitted, the matrix *iderlayc* is set to appropriate values in order to skip the computation of continuum derivatives

```
If(icontpar.eq.0)then

        Do jbase=1, imxpro
                Iderlayc(jbase,1)=ilev -1
                Iderlayc(jbase,2)=ilev -2
        End do
Else
        <Normal computation of variable iderlayc>
End if
```

Module **fwdmdl_pt**:

- Changed interface of **mkplev_pt** (added variable *icontpar*)

Module **retr_pt**:

- changed interface with module **updprof_pt** (added variables *rucl* and *rzc0*)

- changed interface with module **tcgeo_pt** (added variable *icontpar*)

- Set the variable icontpar to 1 before the call to module **tcgeo_pt**

```
Icontpar=1
Call tcgeo_pt()
```

- After the call to the module **guesspar_pt**, if no continuum values have to be fitted (i.e. if icontpar=0) the variable *igeocder* is set to appropriate values to skip the computation of continuum derivatives.

```
if (icontpar.eq.0) then
do jgeo=1, igeo
  igeocder(jgeo,1)=imxpro
```

```
 igeocder(jgeo,2)=imxpro-1

 igeocder(jgeo,3)=imxpro-2
End do
End if
```

3.4 Bug correction in temperature/vmr derivatives

Another small bug was discovered in routines **tcgeo_pt.f** and **gcgeo_vmr.f** occurring only with measurements in the new measurement scenario, characterized by tangent altitudes closer than the FOV extension at low altitudes. The problem occurs because a variable is defined inside a do-loop in case that an if- condition is verified. As soon as it happens the program exits the do-loop. In case this condition is never verified, as may happen with the new measurement scenario, the program exits the do-loop without the computation of this variable. The correction consists in defining this variable correctly after the do-loop in case that the condition inside the do-loop is never verified.

3.4.1 Code modifications

tcgeo_pt.f [gcgeo_vmr.f]

During the computation of *igeotder(jgeo,3)* [*igeogder(jgeo,3)*], a line command has to be added (see highlighted line below) to define the variable *i3*, used to detemine the lowest parameter level affecting the spectrum due to FOV convolution. This operation has to be done just after the do-loop where this variable would be computed if the distance between the height corresponding to *igeotder(jgeo,2)* [*igeogder(jgeo,2)*] and the height corresponding to one of the parameters below was greater or equal to half of FOV extension.

```
Do jgeo=1,igeo
……
mpar= [# of fitted parameters between geometry 1 and geometry jgeo]
…….
If(igeogder(jgeo,2) lt. ipar) then
    Do j=mpar+1,ipar
      If(rzpar(mpar)-rzpar(j)).ge.(rintup*0.5d0))then
        I3=j-mpar
         Goto 100
      Endif
    End do
    I3=ipar-mpar      !line to be added
100 continue

  igeogder(jgeo,2)+i3
else
  igeogder(jgeo,3)=igeogder(jgeo,2)
end if
End do
```

3.5 Update of partitioning function for HNO3

This modification is meant to correct a discrepancy of about 3% between the HNO3 profiles retrieved using cross-section lookup tables (LUT) calculated by Oxford and line-by-line (LBL) calculations. The discrepancy arises from the fact that the coefficients for the internal partition function for HNO3 used in the line-by-line calculations have not been updated with the values recently calculated in the frame of a different ESA study and that have been used for the computation of the LUTs.

Currently, in the subroutines cross_pt and cross_vmr of ORM the internal partition function for HNO3 is calculated as:

Q(T) = C0 + C1*T + C2*T**2 + C3*T**3
with:
T = current temperature in [K]
C0 = -.7420795579D+04
C1 = .3498357216D+03
C2 = .8905132937D-01
C3 = .3935627923D-02

Using these values we have Q(T=296K) = .213822D+06.
The line intensity is then proportional to Q(T=296K)/Q(T).

The new values to be used, that are the ones used for generating the LUTs, are:

C0 = -4.0963D+04
C1 = 8.8476D+02
C2 = -2.8347D+00
C3 = 9.2646D-03
and Q(T=296K) = 2.12832D+05

It has to be noticed that the ESA products are calculated using LUTs, therefore they are not affected by outdated coefficients that are implemented in the IPF and this modification will only be used when LBL calculation is enabled.
For this reason it has been decided that this modification will be implemented but it will not be tested until the use of LBL calculation is required.

4 High level description of the modifications to be introduced in the Level 2 processor that do not affect the ORM

The modifications that do not affect ORM are:

1. Flexibility for adapting Level 2 pre-processor to different floating altitude commanding
2. VCM of a-priori profiles for the computation of the optimal initial guess profiles
3. Correction of ILS computation
4. Species dependent cloud filtering
5. Altitude correction using ECMWF data
6. Changes in Level 2 output file format

4.1 Flexibility for adapting Level 2 pre-processor to different floating altitude commanding

In the new MIPAS nominal scenario the measurement altitude grid has fixed steps, but the lowest tangent altitude is commanded to oscillate as a function of latitude to reflect the latitude variation of the tropopause level around the globe. The lowest tangent altitude is commanded to vary according to the laws:

1. minimum_tangent_altitude = A + B * cos (2 * tangent_point_latitude) with A=6 km and B=3 km (before the 3rd June 2005)

2. minimum_tangent_altitude = A - B * cos (90° -|tangent_point_latitude|) with A=12 km and B=7 km (after the 3rd June 2005)

The law of latitude dependence implemented in the Level 2 is different from the laws reported above and it is computed using only 2 input parameters contained in the file MIP_PS2_AX (PS_FRAME), namely tropopause at pole and increment.
These inputs are not enough to represent the 2 formulas reported above.

We propose to implement the following law of latitude dependence for the minimum tangent altitude in the Level 2 processor:

$$A + B\cos(2\vartheta) - C\cos(90° - |\vartheta|),$$

where θ is the tangent point latitude averaged on the scan, and to add an input parameter in the file MIP_PS2_AX.
After this modification the two laws used so far for the altitude commanding can be obtained setting the following parameters:

Latitude law	A	B	C
1)	6	3	0
2)	12	0	7

4.2 VCM of a-priori profiles for the computation of the optimal initial guess profiles

The initial guess profiles or the assumed profiles of each retrieval are determined as a weighted mean between the a-priori profile (appropriate merging of climatological profiles and ECMWF profiles, if available) and the retrieved profile at the previous iteration. The weight is given by the VCM of the different profiles. While the VCM of the retrieved profile is provided by the Level 2 processor, the VCM of both the ECMWF profile and the climatological profile (assumed to have only the diagonal and the first off diagonal elements different from 0) can be computed using inputs contained in the MIP_PS2_AX file, namely the standard deviation E_0 at a given pressure p_0, the gradient of standard deviation with pressure gradE and the correlation Cr_1.
The approximation of assuming a non-complete VCM matrix was proved to be inadequate, because correlations between the different points of the profile are not properly taken into account in this way.
In order to properly take into account correlations it was decided to provide in input the complete VCM of both climatological and ECMWF profile.

The modification in the auxiliary data files consists in replacing, in the file MIP_PS2_AX, all the fields used for the definition of the VCM of both the climatological profiles and the ECMWF profiles with complete VCMs on the grid of the climatological profile.

Concerning the code, only the initial part of the algorithm has to be modified. The complete VCM matrices have to be read from the MIP_PS2_AX file, and these matrices, defined on the grid of the climatological profile, have to be interpolated to the grid of the retrieved profiles. All the other operations in the algorithm for the optimal initial guess computation remain unchanged.

4.2.1 Modifications in I/O files

Modifications involve only the file MIP_PS2_AX.
To be deleted: E_0^{IG2}, p_0^{IG2}, $gradE^{IG2}$, Cr_1^{IG2}, E_0^{ECMWF}, p_0^{ECMWF}, $gradE^{ECMWF}$, Cr_1^{ECMWF}.
To be inserted:

1. complete VCM of IG2 profile on the grid of climatological profile
2. complete VCM of ECMWF profile on the grid of climatological profile

4.2.2 Algorithm description

We describe the algorithm that performs the linear interpolation of the VCM of climatological/ECMWF profile from the grid of climatological profile to the grid of retrieved profile.
Since the linear interpolation from a grid to another one is a linear operation, this can be represented by a matrix. Then the interpolation of matrix VCM from the climatological grid to the retrieval grid can be obtained as a product between matrices.

Note: the climatological grid on which the input VCM is defined is assumed to cover the complete range of the retrieval grid.

Computation of the transformation matrix B which performs the linear interpolation from the climatological grid (*zclim(1→nclim)*) to the retrieval grid (*zr(1→nr)*).

1) Initialisation of matrix B(nr*nclim) to 0.
2) For each row j of the matrix (j=1 →nr)
 2.1) Determination of the couple of contiguous levels (jj, jj+1) including height zr(j):
  for each k (1 → nclim-1)
   IF((zclim(k+1) ≤ zr(j) ≤ zclim(k)) then jj=k
 2.2) Determination of $B_{j,jj}$ and $B_{j,jj+1}$

$$\text{Alpha}=\frac{zr(j)-zclim(jj)}{zclim(jj+1)-zclim(jj)}$$

$B_{j,jj}$ = 1-alpha
$B_{j,jj+1}$ = alpha

3) Transformation of matrix VCM(nclim*nclim) from climatological to retrieval grid:
 VCM(nr*nr)=B(nr,nclim)*VCM(nclim*nclim)*B^T(nr,nclim) .

4.3 Correction of ILS computation

A modification in Level 2 pre-processor is needed for correcting an error detected by BOMEM during the Commissioning Phase. The error involves the reading of two variables related to ILS computation from the auxiliary file MIP_PS2_AX, in particular the scalar variable *lin_shear_var_z* (expected to be a vector) was swapped with the vector *lin_shear* (expected to be a scalar).
To be noticed that in order to reduce the effect of the resulting error in the computation of the ILS a temporary short-term fix in the PS2, suggested by BOMEM, was implemented in the PS2. This consists in averaging the shear variance values in order to obtain a scalar that fits the single shear variance field currently available, and duplicating the linear shear scalar in order to create a vector to fill the eleven shear fields currently available.
As soon as the bug will be removed from the Level 2 pre-processor, correct values for the variables *lin_shear_var_z* and *lin_shear* will have to be put in the file MIP_PS2_AX.

4.4 Species-dependent cloud filtering

The description of the modifications to be performed to the cloud filtering algorithm is contained in a separate document by John Remedios.

4.5 Altitude correction using ECMWF data

Two sets of tangent altitudes are available in MIPAS L2 products at the moment:

1. engineering altitudes
2. corrected altitudes

Corrected altitudes are calculated through the following formula:

$$z_i = z_1 + \Delta_i(p_i, T_i) \qquad (4.1)$$

where z_1 is the lowest engineering altitude and Δ_i is the term determined exploiting the hydrostatic equilibrium law and the information on the engineering measurement of the pointing direction ([RD1] and [RD2]).

Use of (4.1 implies that corrected altitudes are affected by the bias of the lowest engineering altitude.
For this reason we propose to calculate a new set of tangent altitudes, maintaining the use of hydrostatic equilibrium and exploiting ECMWF data to correct for this bias.

Basic definitions

•**ECMWF** MIPAS Level 2 processor already uses ECMWF profiles to compute the initial guess profiles. For the following operations use will be made of only geopotential profiles (converted to geometric altitude), given on a fixed pressure grid from 1000 to 1 hPa.
•**MIPAS Corrected Altitudes** tangent altitudes computed exploiting the hydrostatic equilibrium law and considering as fixed the lowest engineering altitude.

4.5.1 High level description of the algorithm

Three MIPAS corrected tangent altitudes (z_j, z_{j+1}, z_{j+2}) are used to calculate the new altitude set. At the 8th QWG Meeting the QWG team suggested to make use of the MIPAS corrected tangent altitudes corresponding to nominal altitudes 18, 21 and 24 km; in case that one or more of these points are not available in the current profile the lowest 3 altitudes should be used.
If ECMWF and MIPAS profiles are not superimposed in at least 3 grid points the altitude correction is not performed.
The three MIPAS altitudes, are written as an increment from the lowest altitude (z_1), are equal to:

$$z_j = \hat{z}_1 + \Delta_j \qquad (4.2)$$

For the three selected tangent altitudes the corresponding values in the retrieved pressure profile are defined as p_j, p_{j+1}, p_{j+2}.
ECMWF profile is interpolated to MIPAS pressure levels obtaining three geometric altitudes values y_j, y_{j+1}, y_{j+2}.

Description of the minimisation formula

We want to minimise the following cost function:

$$f(z_j, z_{j+1}, z_{j+2}) = (z_j - y_j)^2 + (z_{j+1} - y_{j+1})^2 + (z_{j+2} - y_{j+2})^2 \qquad (4.3)$$

Appling eq. 4.2 to the previous equation we obtain:

$$f(z_1) = (\hat{z}_1 + \Delta_j - y_j)^2 + (\hat{z}_1 + \Delta_{j+1} - y_{j+1})^2 + (\hat{z}_1 + \Delta_{j+2} - y_{j+2})^2 \qquad (4.4)$$

and minimising $\left(\frac{\partial f}{\partial \hat{z}_1} = 0\right)$ with respect to z_1 we get:

$$\hat{z}_1 = \frac{(y_j + y_{j+1} + y_{j+2} - \Delta_j - \Delta_{j+1} - \Delta_{j+2})}{3} \qquad 4.5)$$

Calculation of the altitudes corrected using ECMWF data

The new set of corrected tangent altitude is calculated substituting $z_1^{'}$ in (4.2.

$$z_i^{'} = \hat{z}_1 + \Delta_i \qquad (4.6)$$

Algorithm description

Inputs

- MIPAS corrected altitudes **z**.
- MIPAS retrieved pressure, $\mathbf{p}_{\mathbf{mipas}}$.
- ECMWF geometric altitude profile, **y**.
- ECMWF fixed pressure levels, $\mathbf{p}_{\mathbf{ecmwf}}$.

Outputs

- MIPAS tangent altitudes corrected using ECMWF data, **z'**. This is a new output vector to be added in the output files.

Algorithm

Description of the operations to be performed:

Selection of the subset of MIPAS altitudes to be processed

Select three MIPAS corrected tangent altitudes, z_j, z_{j+1}, z_{j+2}:
select z_j, z_{j+1}, z_{j+2} as the sweeps at nominal altitude 18, 21, 24 km;
if the lowest nominal altitude is >18 km select the three lowest sweeps;
if MIPAS profiles and ECMWF do not superimpose the computation stops.

Selection of the subset of MIPAS pressures

Select values of pressure profile p_j, p_{j+1}, p_{j+2} corresponding to altitude z_j, z_{j+1}, z_{j+2}.

Calculation of Δ

Calculate Δ as:

$$\Delta_i = z_i - z_1 \qquad 4.7)$$

Where z_1 is MIPAS lowest tangent altitude.

Interpolation of ECMWF profile to MIPAS pressure values

Calculate y_j, y_{j+1}, y_{j+2}, altitude values of ECMWF profiles corresponding to MIPAS pressure values. Perform semilog-interpolation of ECMWF pressure profile at MIPAS pressure levels (i.e. linear interpolation of $log(\mathbf{p}_{ecmwf})$ at $log(p_j), log(p_{j+1}), log(p_{j+2})$)
$\log(p_j)$.

Calculation of the lowest altitude

The lowest unbiased altitude is given by:

$$\hat{z}_1 = \frac{(y_j + y_{j+1} + y_{j+2} - \Delta_j - \Delta_{j+1} - \Delta_{j+2})}{3} \quad (4.8)$$

Calculation of the altitude values

Calculate the tangent altitudes corrected using ECMWF data:

$$z_i^{'} = \hat{z}_1 + \Delta_i \quad (4.9)$$

4.6 Changes in Level 2 output file format

Addition of the following fields in Level 2 output file:

- sun elevation
- complete retrieved profiles (not only on measurement grid)
- OM labels

These changes have been already described in I/O DD Issue 4A.

4.7 Second order polynomial frequency correction

The current approach for the frequency calibration is the following one: the Level 1 processor performs a correction of the frequency scale of the spectra with a linear law that introduces a stretching/shrinking of the scale. The Level 2 pre-processor introduces a further correction in which the ILS is shifted according to a second order frequency dependent polynomial (the used coefficients are contained in the MIP_PS2_AX file). The second order correction performed by Level 2 was proved to be very useful from the tests performed during the commissioning phase. However, long term drifts in the instrumental effects that make the second order correction necessary are not captured by this procedure. In the QWG meeting #9 it was proposed to calculate the coefficients of the second order frequency dependent polynomial correction in the Level 1 processor. The final decision was to modify Level 1 processor in order to compute both the coefficients for the linear correction and the ones for the second order polynomial correction. The three parameters (a = linear coefficient obtained with the linear fit, b = linear coefficient obtained with the polynomial fit, c = quadratic coefficient obtained with the polynomial fit) will be reported in Level 1 files.
The Level 2 pre-processor has to be modified in order to perform the ILS shift with either the coefficients contained in MIP_PS2_AX file, as currently made, or the ones contained in Level 1.

Modifications in I/O files

A switch has to be added in the MIP_PS2_AX file to decide which parameters have to be used for the frequency corrections: the ones from Level 1 files or the ones in MIP_PS2_AX file.

Modifications in the algorithm

The Level 2 pre-processor has to read the new switch. If the switch requires the MIP_PS2_AX coefficients the same procedure, as currently made, is used for the ILS shift. If the switch requires the use of Level 1 correction, the three coefficients (a, b, c) for the frequency corrections stored in Level 1 file have to be read and the following shift is applied:

$$\Delta = (b - a)\sigma + c\sigma^2$$

5 Summary of the modifications

Table 1 summarises the modifications to be implemented in MIPAS Level 2 processor, with indication of the related modifications in ADF2 format and contents and in Level 2 output file.

LEVEL 2 CODE MODIFICATIONS	MODIFICATIONS IN ADF2 FORMAT	MODIFICATIONS IN ADF2 CONTENTS	MODIFICATIONS IN MIP_NL_2 FILE FORMAT
A-posteriori regularization		MIP_PS2_AX: to be set regularisation flag to 2.	To be added: AKM (itop*itop) for each species. Remarks: Regularised profile replaces the non- regularised profile VCM of regularized profile replaces the VCM of the non-regularised profile
Conditioning parameter			To be added: Conditioning parameter of matrix RA at the last iteration (scalar)
Bug correction in continuum derivatives			
Bug correction in temperature/vmr derivatives			
Update of partitioning function for HNO_3			
Flexibility for adapting Level 2 pre-processor to different floating altitude commanding.	MIP_PS2_AX: to be added flag addressing the formula used for floating altitude commanding.	MIP_PS2_AX: _setting of the new flag _updating of Htrop_pole and Increment	

LEVEL 2 CODE MODIFICATIONS	MODIFICATIONS IN ADF2 FORMAT	MODIFICATIONS IN ADF2 CONTENTS	MODIFICATIONS IN MIP_NL_2 FILE FORMAT
VCM of initial guess profiles	MIP_PS2_AX: to be deleted E_0^{IG2}, p_0^{IG2}, $gradE^{IG2}$, Cr_1^{IG2}, E_0^{ECMWF}, p_0^{ECMWF}, $gradE^{ECMWF}$, Cr_1^{ECMWF} To be added complete VCMs for the both climatological and ECMWF profiles on the climatological profile grid.	MIP_PS2_AX: settings of the VCMs of climatological and ECMWF profiles.	
Correction of ILS computation		MIP_PS2_AX: settings of corrected values for ILS computation	
Species-dependent cloud filtering	Inputs from Remedios	Inputs from Remedios	To be added: species-dependent thresholds found by cloud filtering algorithm. Inputs from Remedios. In case of species-independent cloud filtering (current code), the modifications in level 2 output file for including cloud filtering thresholds have been already described in I/O DD 4A (MDS#1, 25-26-27-28).
Altitude correction using ECMWF data			To be added: Altitude grid corrected with ECMWF data

LEVEL 2 CODE MODIFICATIONS	MODIFICATIONS IN ADF2 FORMAT	MODIFICATIONS IN ADF2 CONTENTS	MODIFICATIONS IN MIP_NL_2 FILE FORMAT
Changes in Level 2 output file format			To be added: • sun elevation: already described in I/O DD 4A, ADS#2, Field #11 • complete retrieved profiles (not only on measurement grid), already described in I/O DD 4A • OM labels, already described in I/O DD 4A
Second order polynomial frequency correction	MIP_PS2_AX: to be added switch for deciding whether coefficient for frequency correction have to be read either from Level 1 file or from MIP_PS2_AX.	MIP_PS2_AX: setting of the new switch.	•