`tellrem`
An IDL package for removing telluric lines from VLT/X-Shooter spectra

Natascha Rudolf, Hamburger Sternwarte, Universität Hamburg
natascha.rudolf@hs.uni-hamburg.de

Version 1.1, December 2014

Contents

Chapter 1

Preface

The `tellrem` package is designed to remove telluric absorption lines from nodded VLT/X-Shooter spectra using atmospheric transmission spectra modelled by the radiative transfer code LBLRTM (Line-By-Line Radiative Transfer Model, Clough et al. 2005). The nodding already removes the telluric emission. Since the models include atmospheric extinction the spectra are also corrected for this effect.
The `tellrem` package is presented in Rudolf et al. (2014). It is based on the approach presented by Seifahrt et al. (2010). If you use this package, please also cite their paper.
While Rudolf et al. (2014) describes the general process, this document describes how to use the package. Chapter 2 explains the installation and the other steps you have to take before you can use the package. Chapter 3 introduces you to the routines in the package and explains how to use them. Chapter 4 gives some introduction to LBLRTM. Although this package was designed for use with X-Shooter spectra, its routines can in principle be used for any spectrum. Chapter 5 gives some advice for this. In Appendix A you can find the documentation of the functions and procedures included in the `tellrem` package.

The design is optimised for running on Linux with IDL in command line mode. The package was tested using Ubuntu 10.04.4 and 12.04.5 LTS, IDL Version 7.1.1., the IDL Astronomy User's Library in the version as of June 2010, MPFIT Version 1.75, and LBLRTM Version 12.0 with LNFL Version 2.6 and AER line parameter database Version 3.0.

Chapter 2

Preparations

Before you can use the package a few preparations are required. These are discussed here.
The first step is to download and prepare software and data that are copyrighted by others, so they are not included in this package:

- LBLRTM (see Sect. 2.1)
- line database for LBLRTM (see Sect. 2.2)
- MIPAS model atmosphere (see Sect. 2.3.2)
- MPFIT and IDL Astronomy User's Library (see Sect. 2.4)

The second block of steps is related to your observations, so if you want to use the package on another X-Shooter data set you just have to redo these steps:

- obtain GDAS sounding files (Sect. 2.3.1)
- data reduction (Sect. 2.5)
- adapt `tellrem` input file (Sect. 2.6)

2.1 Installation of LBLRTM

LBLRTM can be downloaded for free from the website of Atmospheric & Environmental Research (AER) (`http://rtweb.aer.com/lblrtm_frame.html` → Code & Examples) for non-commercial use. You need to build the program from the source code before using it. In the LBLRTM package you find three `.tar.gz` archives (`aer_v_XXX.tar.gz`, `aerlbl_vXXX.tar.gz`, and `aerlnfl_vXXX.tar.gz`). `aerlbl_vXXX.tar.gz` contains LBLRTM. Once you unpacked it, you find building instructions in the subdirectory *build* where you have to choose the appropriate Makefile for your system. You should choose a double precision one.
Next, you have to build LNFL using the `aerlnfl_vXXX.tar.gz` archive and the appropriate makefile in the subdirectory *build*. Also, you have to unpack the `aer_v_XXX.tar.gz`.
Add the directories containing the executables to your PATH variable.

2.2 Creating the line database

For running LBLRTM a line database in a certain format is required. This is generated by the program LNFL. It creates the line database for the wavelength region you are interested in from the AER version of the HITRAN database. Since the database is relatively large ($\sim 300\,\mathrm{MB}$) using the complete database slows LBLRTM down. Using a customised database is recommended.
The first input is the AER version of the HITRAN line database as an ASCII table, named `TAPE1`[1].

---

[1] The file naming `TAPEXX` comes from the times when there were no hard disks and everything was stored on tape drives.

You simply rename the complete line list file called `aer_v_XXX` to `TAPE1`. You find the line list file in the archive `aer_v_XXX.tar.gz` in the subdirectory *line_file*. The second input is the `TAPE5` which states the wave number region and molecules to use. The `TAPE5` looks like this:

```
$ Example TAPE5
␣␣4000.␣␣␣␣25233.␣␣␣
111111111111111111111111111111111111111␣␣␣␣LNOUT
%
```

The input `TAPEXX` files are made up of several so-called `records` which are entered line by line. `Record 1`, i.e. line 1, signals the start of input (`$`), the rest is available for user comments. `Record 2` contains in 20 characters the start and end wave number to be used in the format `F10.3`. They should be 25 cm^{-1} less/greater then the wave number intended to be used in LBLRTM. The first 39 characters of `record 3` are switches for the 39 molecules available[2]. A `1` activates the corresponding molecule, a `0` disables it. After 4 blank characters, there are 40 characters for the `HOLIND` (HOLlerinth INDicator to select LNFL options) character keywords. Using `LNOUT` creates `TAPE7` containing a list of all chosen lines in text format. Information on the other available keywords and the available molecules can be found in the `lnfl_instructions` document in the subdirectory *docs*. The `%` in line 4 signals the end of the `TAPE5` file.

Put `TAPE1` and `TAPE5` into a directory and call LNFL there from the command line with the executable you built. It will create a `TAPE3` that contains the customised database (and `TAPE7` if applicable). `TAPE3` is required to run LBLRTM.

2.3 Model atmosphere

The model atmosphere is a combination of two models. This section describes how to obtain them.

2.3.1 Sounding files from GDAS

To enable the package to model the weather conditions during your observations it requires the so-called sounding files. You obtain these in the following way:

- go to `http://ready.arl.noaa.gov/READYamet.php`
- enter the coordinates of the observatory, here Paranal (Lat -24.6272, Long. -70.4048), in the "Select a Location" field and click "continue"
- in the row "Sounding" select "GDAS" and click "Go"
- select the appropriate file (see file format explanation given on the site) and click "Next"
- choose the start time of the animation, activate it by choosing "Animation: GIF" and set the duration you need to cover all your observations. The package will search for the sounding file closest to your observation from the files provided
- set "Output Options" to "Text only" and "Graphics" to "Text Listing"
- enter the displayed access code into the text box and click "Get sounding"

From the page, you now see, you have to extract the data for pressure (PRESS), height (HGT(MSL)), temperature (TEMP), dew point temperature (DEW PT), wind direction and speed (WND DIR, WND SPD) and copy it to individual text files for the individual time steps. One file should look like the example in Figures 2.1 (see also file `GDASexample` in the package). Name the individual files `GDASYYYY_MM_DD_HH`, where YYYY is the year (e.g. 2010), MM the month (e.g. 05 for May), and DD the day (e.g. 04 for the fourth). `tellrem` expects the first 6 lines to be comments and the data to begin in line 7. Put all files you created into a directory that contains nothing else.

---

[2] This number is valid for version 3.0 of the line database.

```
YR: 2010   MON: 05   DAY: 04   HOUR: 00    AT POSITION: 290.6  66.4  LAT.:-24.63  LON.: -70.40

PRESS HGT(MSL) TEMP DEW PT  WND DIR  WND SPD
HPA      M      C     C       DEG     M/S
E = Estimated Surface Height

 915.   916.   14.2   10.8   191.7     2.7
 900.  1030.   20.3    4.3   129.3     1.3
 850.  1518.   17.8   -0.6    41.5     2.1
 800.  2033.   15.2   -6.2    27.3     4.1
 750.  2576.   12.0  -11.4    16.8     5.7
 700.  3149.    8.6  -16.4     9.5     5.0
 650.  3756.    5.1  -19.4   325.4     3.2
 600.  4405.    1.4  -21.3   251.8     4.7
 550.  5099.   -3.0  -30.0   250.4     7.1
 500.  5845.   -8.4  -32.1   256.1     8.1
 450.  6654.  -13.5  -38.9   247.9     9.9
 400.  7540.  -19.2  -40.2   250.2    14.2
 350.  8517.  -27.2  -44.7   252.2    15.5
 300.  9606.  -36.3  -50.3   245.2    18.7
 250. 10843.  -46.0  -61.0   241.9    22.4
 200. 12302.  -53.1  -70.5   233.3    28.7
 150. 14121.  -62.3  -99.4   237.8    37.7
 100. 16552.  -73.1 -273.1   248.4    18.5
  50. 20656.  -62.3 -273.1   276.0     4.9
  20. 26520.  -48.4 -273.1   126.7     7.0
```

Figure 2.1: Example of GDAS sounding data text file. Its name would be `GDAS2010_05_04_00`.

2.3.2 MIPAS model atmosphere

The equatorial MIPAS model atmosphere, constructed by John Remedios (U. Leicester), can be obtained from `http://www-atm.physics.ox.ac.uk/RFM/atm/`. From the "MIPAS Model Atmospheres (2001)" section download `equ.atm`, the equatorial, day-time model.

2.4 MPFIT and IDL Astronomy User's Library

The `tellrem` package makes use of the MPFIT package by Markwardt (2009) as well as several procedures and functions from the IDL Astronomy User's Library. If you do not already use them download them from `http://purl.com/net/mpfit` and `http://idlastro.gsfc.nasa.gov/`, respectively.

2.5 Reduction of the spectra

`tellrem` expects the spectra to be reduced but not flux-calibrated. All other steps of the data reduction should be completed, i.e. bias, sky, and dark subtraction, flat-fielding, wavelength calibration, and extraction and merging to 1D spectra. Since the flux calibration also includes an extinction correction using the package on flux-calibrated spectra would take into account the extinction twice.

2.6 `tellrem` input file

`tellrem` necessitates several pieces of information for running. These are summarised in a text file that will be read by the procedure `LOADTELLREMINFO`. An example file called `info_for_tellrem_example` is part of the package. You have to adapt it to your setup. The content of this file is shown in Table 2.1. The format needs to be adhered to. The first line contains the directory with the GDAS sounding files you created in Section 2.3.1. The second line holds the absolute path to where you saved the MIPAS model atmosphere in Section 2.3.2. In the third line you have to enter the name of the LBLRTM executable you built in Section 2.1. The fourth line is the path to where your X-Shooter data is stored. The package expects subdirectories for each individual observation within this directory. It

Table 2.1: Content of the `tellrem` input file (see also `info_for_tellrem_example` in the package)

```
/here/lie/the/GDAS/soundingdata/    ; path of directory containing GDAS files
/here/is/the/MIPAS/model/equ.atm    ; path to MIPAS model atmosphere
lblrtm_vXX.X_linux_xxx_dbl          ; name of LBLRTM executable
/here/are/the/fits/subdirectories/  ; path to directory containing reduced spectra
SCI                                 ; type of observation, i.e. SCI, FLUX, TELL
```

uses the directory name as object identifier that will be put as auxiliary information into the structure containing the final spectrum (see Chapter 3). Within the subdirectory the `.fits` files of the merged 1D spectrum are expected to have the default pipeline file name, i.e. `TYPE_SLIT_MERGE1D_ARM.fits`. While `ARM` will be replaced by the package itself, you have to specify in line 5 of the input file what kind of observation you took. This usually is one of `SCI` for science observations, `FLUX` for flux calibration observations, or `TELL` for telluric standard observations.

2.7 Testing the package

Finally, the package provides a test run to check whether the main preparatory steps have been successful. The test is performed by the procedure `TELLREM_TEST_RUN`. It needs 3 inputs: first, the absolute path to where you saved the tellrem package, second, the name of the LBLRTM executable, that you built in Section 2.1, and, third, the absolute path to where you saved the MIPAS model atmosphere in Section 2.3.2. Create a directory and put the customised line database `TAPE3`, that you created in Section 2.2, into it. Then you start IDL on the command line in this directory and start the test by typing[3]

```
TELLREM_TEST_RUN, '/the/tellrem/package/is/here/', 'lblrtm_executable',
'/MIPAS/model/is/here/equ.atm'
```

The test will take about 15 minutes on a standard desktop PC[4]. It removes the telluric lines from an observation of MP Mus that is provided with the package and creates the plot file *Tellrem_test_run_comparison.ps* comparing the just determined telluric line removed spectrum to the spectrum provided in the package. Both spectra should be very similar. They probably will not be exactly the same since creating the spectra involves fitting which might lead to slightly different results on different computers and you are probably using an updated version of LBLRTM and the line database, so there might be changes introduced through that. If the comparison plots look suspicious to you, take a look at the plot file *tellrem_test_run_MPMus.ps* that the procedure also created. Here, you see the fit of the telluric model to the data and the cleaned spectrum. For comparison with the successful run provided in the package, this plot file is also available in the directory *example_files/Test_Run*.

If the test was not successful, you will have to check and redo, if applicable, the preparatory steps to find the specific problem.

---

[3] Make sure the `tellrem` package is in your IDL search path.

[4] Everything involved, i.e. data, programs etc., should be available locally on your PC. If this is not the case, network communication will slow down the program by factors of up to 10.

Chapter 3

The program package

Once you completed the preparations in Chapter 2, create a directory where you put the customised line database `TAPE3`, that you created in Section 2.2. This is the first input for running LBLRTM. Then you start IDL on the command line in the directory the `TAPE3`[1] is in and run the procedure `TELLREM` by typing

`TELLREM, '/path/to/your/info_for_tellrem', telluricremovedspectra`

and wait for it to finish. Handling one spectrum usually takes between 15 and 20 min. The IDL variable *telluricremovedspectra* will then contain a structure (or an array of structures in case of more than one observation) with the tags explained in Table 3.1. Additionally, two IDL `.sav` files are created (`tellremparams.sav` and `spectra_tellrem.sav`) that contain the parameters and the cleaned spectra, respectively, as well as plots for each object/observation showing the fits of the model to the data and the cleaned spectrum to enable quality control.

`TELLREM` does the following: First, it calls `LOADTELLREMINFO` to set the information required for running. These are stored in the common block `TELLREM_INFO` so that every routine can access them. After calling `GETOBJECTS` to determine the "objects", i.e. the directory names, it runs `TELLREMPARAMETERS`. This function determines the parameters needed to create telluric models matching the observation. After reading the auxiliary information, `GETABUNDANCES` determines the abundances of the dominant molecules (H_2O, CO_2, CH_4, and O_2) from sufficiently isolated line groups.

The fit is done by `FITTELL`. It uses MPFIT to fit the model produced by `TELL` to the data. Free parameters are the abundance relative to the atmospheric model of the molecule to be fitted, the wavelength shift to match the observed features (to allow for uncertainties in the wavelength solution), slope and intercept of the straight line used to adjust the transmission spectrum to the observed spectrum, and the width of the Gaussian, representing the instrumental profile of X-Shooter. `TELL` calls `RUNLBLRTM` to calculate the model for a specific set of abundances and then applies the adjustments to match the model to the observed data. `RUNLBLRTM` itself first creates the second input file for LBLRTM, the so-called `TAPE5` (see Section 4.1 for how this is created), and then runs LBLRTM.

The abundances are then fixed and `TELLREMPARAMETERS` calls `FITTELL` repeatedly to fit the remaining parameters (wavelength shift, slope and intercept, width of Gaussian) for the VIS and NIR spectra. The fits are done in 300 Å long segments.

The parameters are then handed over to `TELLREMSPECTRA` which creates the telluric line removed spectra. For the UVB spectra an extinction curve is determined by `UVEXTINCTION`. It calculates an appropriate LBLRTM model in this region. As there are no strong lines in this region that would be distinguishable from noise with X-Shooter's resolution no fitting is necessary here. The model is smoothed to avoid introducing artefacts by the small lines present in the high-resolution model spectrum. Since there are no lines in the database below 3963 Å, no model can be calculated below 3672 Å. Thus, a fourth order polynomial is fitted to the region between 3700 Å and 4600 Å. This polynomial is then interpolated down to 3100 Å and used as extinction. The wavelength coverage of X-Shooter reaches down to 3000 Å, but the first 100 Å are cut off because they contain in most cases only low and noisy signal. The observed spectrum and the pipeline error are divided by this extinction to create the extinction-corrected spectrum. For the other spectral parts `TELLREMSPECTRA` runs `RECRFITTELLRES`

---

[1] LBLRTM searches for its input files in the directory it is called from and quits with an error if it does not find them.

Table 3.1: Structure tags and their content of the output of `TELLREM`

Tag name	Meaning and content
object	object name (string), i.e. the name of the subdirectory the data were read from
obsdate	Julian Date of observation (double scalar), taken from header of VIS data
obsaltitude	altitude angle the observation was taken at (double scalar), taken from header of VIS data
exptimeu	exposure time in seconds in the UVB arm (double scalar)
exptimev	exposure time in seconds in the VIS arm (double scalar)
exptimen	exposure time in seconds in the NIR arm (double scalar)
wclu	wavelength array in Å for the UVB data (3100 Å–5880 Å)
clu	flux array in ADU of UVB corrected for extinction
eclu	error[a] array in ADU of the UVB flux
wclv	wavelength array in Å for the VIS data (5500 Å–10 170 Å)
clv	flux array in ADU of VIS corrected for extinction and telluric absorption
eclv	error[a] array in ADU of the VIS flux
wcln1	wavelength array in Å for the NIR data in region 1 (9960 Å–13 400 Å)
cln1	flux array in ADU of NIR in region 1 corrected for extinction and telluric absorption
ecln1	error[a] array in ADU of the NIR flux in region 1
wcln2	wavelength array in Å for the NIR data in region 2 (14 500 Å–18 100 Å)
cln2	flux array in ADU of NIR in region 2 corrected for extinction and telluric absorption
ecln2	error[a] array in ADU of the NIR flux in region 2
wcln3	wavelength array in Å for the NIR data in region 3 (19 600 Å–24 000 Å[b])
cln3	flux array in ADU of NIR in region 3 corrected for extinction and telluric absorption
ecln3	error[a] array in ADU of the NIR flux in region 3

Notes. (a) The errors are propagated pipeline errors, there is no additional contribution to the errors by the telluric line removal considered; (b) Although X-Shooter's wavelength coverage goes in principle up to 2.5 μm the part above 2.4 μm was omitted due to low data quality in this region in the spectra on whose basis this package was designed.

to recreate the model. The observed spectrum and the pipeline error are divided by the model to remove the telluric absorption from the spectra. Finally, the structure or array of structures described in Table 3.1 is returned. For more details on the individual routines take a look at the documentation of the routines in Appendix A and the code itself.

Chapter 4

Basics of using LBLRTM

This chapter is intended to give you an introduction into the use of LBLRTM, not to explain how it works in detail. If you are interested in details and the other capabilities and applications of LBLRTM, the paper by Clough et al. (2005) is a good starting point. Some basic introduction to its attributes is also available on the AER website (`http://rtweb.aer.com/lblrtm_frame.html` → Description). The `FAQ_LBLRTM.pdf` and the `lblrtm_instructions.html` documents available in the *docs* subdirectory of the LBLRTM package provide some help and guidance. The HTML is the main source for information on the large number of input parameters available.

4.1 The `TAPE5`

The `TAPE5` is the input file for running LBLRTM. It contains all parameters that control its operation. It consists of a number of `records`. It is created by `RUNLBLRTM`. You find an example `TAPE5` in Appendix B where each `record` is indicated so that you can look it up easily in the instructions. Each `record` has a specified format given in the instructions. If you enter a blank or a zero at the position of a parameter, the default value will be chosen by the program, if there is one.

The first group of `records` (`records 1.1` to `1.4`, line 1 to 4 in uncommented `TAPE5`) set up the general information on which modules to use and for which wave number region to calculate the model. `Record 1.1` has to start with `$` which signals the start of input. Depending on the modules activated in `record 1.2` a certain set of records has to be present (see `lblrtm_instructions.html` for details) in addition to the mandatory ones. Important switches in `record 1.2` that are set to `1` are `IHIRAC` and `ILBLF4` activating the line-by-line calculation using Voigt profiles and `ICNTNM` activating calculation of all continua including Rayleigh extinction where applicable. `IEMIT` activates the calculation of transmittance and radiance. `IATM` activates automatic or user-supplied layering of the atmospheric model. The remaining switches are set to zero, except for `MPTS` and `NPTS` which are set to 5. `Record 1.3` contains the start and end of the calculation (`V1,V2`). These need to be less then 2020 cm^{-1} apart. `SAMPLE` is the number of sample points per mean halfwidth of a line and is set by default to 4.0. This should be kept to get full accuracy. All other parameters are set to their defaults by either setting to zero, entering the default value, or omitting (i.e. a blank). `Record 1.4` is required in conjunction with `IATM=1`. The only important parameter is `TBOUND` which is the temperature in Kelvin at the upper boundary of the atmosphere. It is set to zero here.

The second group of `records` (line 5 to 209, `records 3.1` to `3.6.3`) determines the model atmosphere used. The model atmosphere is composed of N layers at different heights. For each layer temperature, pressure, and the densities of all molecules are provided. One can either use an included atmospheric profile or provide one (switch `MODEL` in `record 3.1`, line 5). We provide our own, i.e. `MODEL=0`. The model uses a slant path through the atmosphere (switch `ITYPE=3`), starting at the observatory altitude (`H1` in `record 3.2`, line 6) and ending in space (defined to start at 75 km, `HSPACE`), and taking into account the zenith angle of the observation (`ANGLE` in `record 3.2`) to determine the atmospheric layers crossed by the stellar light. `NMOL` is set to 19 as the highest-ranking molecule we use is number 19 (OCS). We actually only use 13 molecules (H_2O, CO_2, O_3, N_2O, CO, CH_4, O_2, NO, SO_2, NO_2, NH_3, HNO_3, OCS). The available ones and their positioning is detailed in Table I of `lblrtm_instructions.html`. The remaining switches of `records 3.1` and `3.2` and all switches of `record 3.3A` (line 7) are set to default. `Record 3.4` (line 8) defines the number of layers in the model (`IMMAX=50`). `Record 3.5` (line 9) defines the altitude (`ZM`), pressure (`PM`), and temperature (`TM`) at the first boundary as well as information on the units the data is provided in (`JCHARP`, `JCHART`, `JLONG`, `JCHAR(M)`, see Table I of `lblrtm_instructions.html`). `Records 3.6.1` to `3.6.3` (line 10 to 12) provide the density data for each molecule (`VMOL(M)`). Since the 13 molecules used here are not the first 13, we have to "use" 19 molecules (switch `NMOL=19` in `record 3.1`) and set the non-used ones to zero. The information in `record 3.5` to `3.6.3` is then repeated for the remaining 49 layers. The `-1.` in line 209 signals the end of the repetition. The data put in these `records` is combined from the GDAS sounding data and the MIPAS model in the following way: First, the height, temperature, pressure, and dew point temperature from the GDAS sounding data are used. The dew point temperature serves as density information for water. The densities of the other molecules are interpolated from the MIPAS model to the heights of the GDAS data. After the end of the GDAS data (usually somewhere between 26 km and 27 km) the data from the MIPAS model is used in its 1 km spacing up to 48 km, followed by using it in 5 km spacing up to 88 km. The density profiles of the individual molecules can be changed by multiplying with a factor. `RUNLBLRTM` has keywords for this purpose for H_2O, CO_2, CH_4, O_2, O_3, NO, and NH_3.

In principle the `TAPE5` could end here. LBLRTM outputs `TAPE12` containing the monochromatic results for the transmittance (and the radiance). Since this file is unformatted another step is necessary to transform it into an ASCII file. The next group of `records` is doing that. Here, LBLRTM is restarted (`records 1.1` and `1.2` with only `IPLOT=1` to activate "plotting", i.e. extracting, line 210 and 211). `Record 12.2A` (line 213) tells LBLRTM to extract the wave number range determined by `V1` and `V2` from the `TAPEXX` specified by `LFILE` (`TAPE12` in our case) in standard output (`IOPT=0`). The remaining parameters are arbitrary. They were used when LBLRTM still had actual plotting capabilities. `JEMIT=0` and `JOUT=3` in `record 12.3` (line 214) state to extract the transmittance and write it to the `TAPEXX` specified by `JPLTFL`, set to `99` here. Again, the remaining parameters are arbitrary. `TAPE99` is read by `RUNLBLRTM`. The `-1.` in line 215 signals the end of the "plotting". The `%` in line 216 signals the end of the `TAPE5`.

4.2 Other points to know

LBLRTM calculates the radiative transfer line-by-line, meaning that the spectral resolution is only limited by the actual width of the Voigt profiles used in the calculations. The parameter `SAMPLE` in `record 1.3` sets the number of sample points per mean halfwidth of a line. This parameter is set to 4, the maximum it can take. For NIR the models have resolutions above 500 000. The models are calculated with an almost equidistant spacing in wave number of $\sim 0.0027\,\mathrm{cm}^{-1}$. Thus, after converting to wavelength you do not have equidistant bins.

As mentioned above, LBLRTM can be used for many more applications and thus has a lot of modules not used here. For example, it can take into account effects like aerosol contribution, clouds, and rain.

Chapter 5

Using `tellrem` for other spectra

Although this package is designed for use on X-Shooter spectra, it can in principle be used for any spectrum. This chapter intends to assist you to adapt it for your situation.

5.1 Input

The model atmosphere is optimised for the VLT site, so you might want to consider using a different model than the MIPAS equatorial model if your data was not taken there. The observatory elevation is an important input to the model. Its by default set to that of the VLT at Paranal. Use the keyword OBSELEVATION in `RUNLBLRTM` to set another one.
You should create a new `TAPE3` only encompassing the wavelength region you are interested in (see Section 2.2).

5.2 Adapting the routines

You cannot use `TELLREM` to run all the steps at once, but you have to create your own versions of `TELLREMPARAMETERS` and `TELLREMSPECTRA` adapted to your data. You might need your own routine to read the data from the `.fits` files. Reading the data is done by RDDAT. It should work with ESO data, but better check.
Most importantly, the abundance determination for H_2O, CO_2, CH_4, and O_2 done by `GETABUNDANCES` will not work if your spectra do not cover the wavelength ranges it requires. Within the X-Shooter spectral range and for X-Shooter resolution the contribution by other molecules is only marginal. Fitting CH_4 seems to get more difficult if the noise level increases (or the resolution is lower). For other spectra you may want to check the contributions of other molecules. Currently, `RUNLBLRTM` only has keywords for adjusting the abundance of H_2O, CO_2, CH_4, O_2, O_3, NO, and NH_3. If you want to check other molecules, just add keywords to the code to access the other available molecules. The model does not use all molecules available in the MIPAS model, so if you think one of them might be important for you, adjust the code to take it into account too.
To check which molecules contribute in your wavelength range, use `RUNLBLRTM` to calculate transmission spectra. Calculate one that contains all molecules, one that contains only water (by setting the others to a very small value, e.g. 10^{-20}, **`but not to zero`**), and one that contains water and the molecule in question. Setting the water abundance to a very small value crashes LBLRTM. Overplot them to see the contributions. Once you identified the important molecules either adapt `GETABUNDANCES` and the parinfo structure handed over to `FITTELL` to your situation by using different molecules and/or regions or skip the predetermination of the abundances and set them as free parameters in the fit by adjusting the parinfo structure accordingly. If you can avoid setting the abundances as free parameters you should do so because the fitting tends to get unstable with many free parameters. In the parinfo structure, also check the start values and limits. Additionally, take a look at `TELL`. You might have to adjust the factors in the parameter assignment to match your spectra so that the numerical values of the parameters MPFIT hands over to it have the same order of magnitude. This is recommended by C. Markwardt for stability. Then, you need to run `FITTELL` with your data and

the adjusted parinfo structure. Another issue to look at is the length of the segments in which the model is fitted. The choice of 300 Å for X-Shooter was mainly based on the instrument stability, so for another instrument you might be able to use longer ones, but keep in mind that the adjustment of the model to the data is done by multiplication with a straight line. This approximation is only valid on local scales.

Finally, either save the cleaned spectrum directly from the output of `FITTELL` or adjust `TELLREMSPECTRA` to feed your data to `RECRFITTELLRES`.

Chapter 6

References

S. A. Clough, M. W. Shephard, E. J. Mlawer, J. S. Delamere, M. J. Iacono, K. Cady-Pereira, S. Boukabara, and P. D. Brown. Atmospheric radiative transfer modeling: a summary of the AER codes. *J. Quant. Spec. Radiat. Transf.*, 91:233–244, March 2005. doi: 10.1016/j.jqsrt.2004.05.058.

C. B. Markwardt. Non-linear Least-squares Fitting in IDL with MPFIT. In D. A. Bohlender, D. Durand, and P. Dowler, editors, *Astronomical Data Analysis Software and Systems XVIII*, volume 411 of *Astronomical Society of the Pacific Conference Series*, page 251, September 2009.

N. Rudolf, H. M. Günther, P. C. Schneider, and J. H. M. M. Schmitt. Modelling telluric line spectra in the optical and infrared with an application to VLT/X-Shooter spectra. Submitted to A&A, 2014.

A. Seifahrt, H. U. Käufl, G. Zängl, J. L. Bean, M. J. Richter, and R. Siebenmorgen. Synthesising, using, and correcting for telluric features in high-resolution astronomical spectra . A near-infrared case study using CRIRES. *A&A*, 524:A11, December 2010. doi: 10.1051/0004-6361/200913782.

Acknowledgements

I thank Christian Lippe (Hamburger Sternwarte) for introducing me to the Seifahrt et al. (2010) paper and helping me get started with LBLRTM. Large thanks also have to go to Boris Voigt (formerly Hamburger Sternwarte) whose notes on how to use LBLRTM have been very helpful and also improved this documentation.

Appendix A

Documentation for `tellrem` routines

A.1 CREATE_STANDARD_PARINFO

Name:

CREATE_STANDARD_PARINFO

Purpose:

This function creates a parameter info structure for use with FITTELL containing appropriate values.

Calling Sequence:

Result = CREATE_STANDARD_PARINFO()

Inputs:

None.

Outputs:

This function returns a parameter info structure for use with FITTELL and in the format expected by MPFIT.

Example:

parinfo = CREATE_STANDARD_PARINFO()

Modification History:

Written by:	Natascha Rudolf, October 2013.

A.2 EXCLUSION

Name:

EXCLUSION

Purpose:

This function extracts from a wavelength array a certain range but without the regions specified to exclude.

Calling Sequence:

Result = EXCLUSION(Wavelength, Range, Exclude)

Inputs:

Wavelength:	Wavelength array from a range of which one wants to exclude certain regions.
Range:	Wavelength range to extract from above array as two-element array in the form [start,end].
Exclude:	Regions to exclude from wavelength array in the range specified as array with up to 10 entries of the form [start1,end1,...,start5,end5].

Outputs:

This function returns an index array for the wavelength array that contains the specified range but does not contain the regions stated in exclude.

Example:

indices = EXCLUSION(Wavelength, [6500.,6800.], [6558.,6568.])

Modification History:

Written by:	Natascha Rudolf, October 2013.

A.3 FITTELL

Name:

FITTELL

Purpose:

This procedure uses LBLRTM to fit a transmission model to the observed spectrum and then uses this model to remove the telluric lines from the spectrum.

Calling Sequence:

FITTELL, Wave, Data, Error, Range, Gdasdata, Obsaltitude, Params, Paramserror, Waveresult, Result, Cleaned, Cleanederror

Inputs:

Wave:	Wavelength array of observed spectrum.
Data:	Flux array of observed spectrum.
Error:	Error of flux measurements.
Range:	Wavelength range of fit as two-element array in the form [start,end].
Gdasdata:	Scalar string containing absolute path to GDAS file.
Obsaltitude:	Altitude angle of observation in deg.

Keyword Parameters:

EXCLUDE:	Regions to exclude from wavelength array in the range specified as array with up to 10 entries of the form [start1,end1,...,start5,end5].
PLOT:	Set this keyword to get a plot of the result.
SILENT:	Set this keyword to suppress informational messages.
PARINFO:	Parameter info structure to use in MPFIT. Default is a standard structure created by CREATE_STANDARD_PARINFO.
TITLEPLOT:	Title of plot.

Outputs:

Params:	Fitted parameters ([smoothing_FWHM, velocity_shift, slope_of_straight, intercept_of_straight, water_abundance, methane_abundance, carbon-dioxide_abundance, oxygen_abundance]).
Paramserror:	Formal $1\,\sigma$ errors of parameters.
Waveresult:	Wavelength array of the fitted part.
Result:	Model fitted with all parameters (to be compared to observation).
Cleaned:	Telluric line removed spectrum.
Cleanederror:	Error of flux measurements divided by transmission model.

Example:

FITTELL, Wave, Data, Error, [9000.,9300.], '/here/lies/the/Gdasdata', 87.2, Params, Paramserror, Waveresult, Result, Cleaned, Cleanederror

Modification History:

Written by:	Natascha Rudolf, October 2013.

A.4 GAUSSCONV

Name:

GAUSSCONV

Purpose:

This function smooths an array by convolving with a Gaussian profile.

Calling Sequence:

smoothed_y = GAUSSCONV(X, Y, Fwhm)

Inputs:

X:	Array (double or float) of the values of the x-axis in ascending order.
Y:	Array (double or float) of same size as x of the y-axis values.
Fwhm:	FWHM in units of the x-axis of the Gaussian profile.

Outputs:

Array (double or float) of same size as x containing smoothed y.

Example:

flux_smoothed = GAUSSCONV(wavelength,flux,0.75)

Modification History:

Written by:	Natascha Rudolf, October 2013.

A.5 GETABUNDANCES

Name:

GETABUNDANCES

Purpose:

This procedure determines the abundances of the main molecules (H_2O, CO_2, CH_4, O_2) producing telluric lines in the X-Shooter spectral range.

Calling Sequence:

GETABUNDANCES, Wavevis, Specvis, Errvis, Wavenir, Specnir, Errnir, Gdasdata, Obsaltitude, Ab_h2o, Ab_co2, Ab_ch4, Ab_o2, Ab_h2o_str, Ab_co2_str, Ab_ch4_str, Ab_o2_str

Inputs:

Wavevis:	Wavelength array of VIS data.
Specvis:	Flux array of VIS data.
Errvis:	Error array of VIS data.
Wavenir:	Wavelength array of NIR data.
Specnir:	Flux array of NIR data.
Errnir:	Error array of NIR data.
Gdasdata:	Scalar string containing absolute path to GDAS file.
Obsaltitude:	Altitude angle of observation in deg.

Keyword Parameters:

SILENT:	Set this keyword to suppress informational messages.
PLOT:	Set this keyword to get a plot of the result.

Outputs:

Ab_h2o:	Abundance of H_2O in this observation.
Ab_co2:	Abundance of CO_2 in this observation.
Ab_ch4:	Abundance of CH_4 in this observation.
Ab_o2:	Abundance of O_2 in this observation.
Ab_h2o_str:	Structure containing the fit results for the individual regions used to determine the abundance of H_2O.
Ab_co2_str:	Structure containing the fit results for the individual regions used to determine the abundance of CO_2.
Ab_ch4_str:	Structure containing the fit results for the individual regions used to determine the abundance of CH_4.
Ab_o2_str:	Structure containing the fit results for the individual regions used to determine the abundance of O_2.

Example:

GETABUNDANCES, Wavevis, Specvis, Errvis, Wavenir, Specnir, Errnir, '/here/lies/the/Gdasdata', 87.2, Ab_h2o, Ab_co2, Ab_ch4, Ab_o2, Ab_h2o_str, Ab_co2_str, Ab_ch4_str, Ab_o2_str

Modification History:

Written by:	Natascha Rudolf, October 2013.
N. Rudolf, October 2014	Adapted parameter borders.
N. Rudolf, Nov. 2014	Added output to save the fit results of the individual regions used to determine the abundances of the main telluric contributors.

A.6 GETOBJECTS

Name:

GETOBJECTS

Purpose:

This function extracts the names of the objects to use the `tellrem` package on.

Calling Sequence:

Result = GETOBJECTS()

Inputs:

None.

Outputs:

This function returns a string array of the folder names located in the path containing the reduced spectra specified in the info file provide to LOADTELLREMINFO. It expects these to be the object names which will be added as auxiliary information during the further run of TELLREM.

Common Blocks:

TELLREM_INFO:	This common block contains relevant folder names and strings for running TELLREM. It has to be initialised by running LOADTELLREMINFO.

Example:

objects = GETOBJECTS()

Modification History:

Written by:	Natascha Rudolf, October 2013.

A.7 IDLPLOTTITLECHANGER

Name:

IDLPLOTTITLECHANGER

Purpose:

This procedure replaces the IDL standard plot title "Graphics produced by IDL" with the file name in the specified postscript file.

Calling Sequence:

IDLPLOTTITLECHANGER, Filename

Inputs:

Filename:	Name of the postscript file to work on (scalar string).

Outputs:

None.

Example:

IDLPLOTTITLECHANGER,'plotfileXYZ.ps'

Modification History:

Written by:	Natascha Rudolf, October 2013.

A.8 LOADTELLREMINFO

Name:

LOADTELLREMINFO

Purpose:

This procedure loads the information necessary to run the procedures of the `tellrem` package and creates the common block TELLREM_INFO. An example file called 'info_for_tellrem', which you can adjust for your needs, is provided with the package. Run this procedure before any of the other functions or procedure of the `tellrem` package.

Calling Sequence:

LOADTELLREMINFO, filename

Inputs:

Filename:	Absolute path to file containing the information as string.

Outputs:

None. But creates the common block TELLREM_INFO. This common block contains the variables Gdasfolder, Modelatmosphere, Executeablename, Spectrafolder, and Obstype. All of them are scalar strings. Gdasfolder contains the absolute path of the folder containing the GDAS files, Modelatmosphere contains the absolute path to the MIPAS model atmosphere, Executeablename contains the name of LBLRTM executable, Spectrafolder contains the absolute path to the folder containing the reduced spectra (in individual subfolders therein), and Obstype contains the prefix specifying the type of observation (i.e. SCI, FLUX, or TELL), which the X-Shooter pipeline assigns to the .fits files.

Common Blocks:

TELLREM_INFO:	This common block contains relevant folder names and strings for running tellrem. It is created by this procedure.

Example:

LOADTELLREMINFO,'/here/lies/info_for_tellrem'

Modification History:

Written by:	Natascha Rudolf, October 2013.

A.9 RDDAT

Name:

RDDAT

Purpose:

This procedure reads VLT/X-Shooter spectra.

Calling Sequence:

RDDAT,File,Wave,Spec,Error,Header

Inputs:

File:	Scalar string containing complete path to file.

Keyword Parameters:

PHOENIX:	Set this keyword if you want to read spectra from the Göttingen Spectral Library by PHOENIX (http://phoenix.astro.physik.uni-goettingen.de/).

Outputs:

Wave:	Wavelength array in Å.
Spec:	Flux array.

Optional Outputs:

Error:	Error array.
Header:	Header of the fits file.

Example:

If you just want to have wavelength and flux use
rddat, '/here/is/the/data/SCI_SLIT_MERGE1D_VIS.fits', wave, spec
If you also want error and header use
rddat,'/here/is/the/data/SCI_SLIT_MERGE1D_VIS.fits', wave, spec, error, header

Modification History:

Written by:	Natascha Rudolf, October 2013.

A.10 READMODATM

Name:

READMODATM

Purpose:

This procedures reads the requested parameter (e.g. pressure, abundance) from the MIPAS model atmosphere.

Calling Sequence:

Result = READMODATM(Parameter)

Inputs:

Parameter:	Parameter to be read (scalar string) in format of MIPAS model atmosphere file.

Outputs:

This function returns the requested parameter in a float array.

Common Blocks:

TELLREM_INFO:	This common block contains relevant folder names and strings for running TELLREM. It has to be initialised by running LOADTELLREMINFO.

Restrictions:

This function only works for the parameters available in the MIPAS model atmosphere. These are: HGT, PRE, TEM, N2, O2, CO2, O3, H2O, CH4, N2O, HNO3, CO, NO2, N2O5, ClO, HOCl, ClONO2, NO, HNO4, HCN, NH3, F11, F12, F14, F22, CCl4, COF2, H2O2, C2H2, C2H6, OCS, SO2, SF6.

Example:

height = READMODATM('HGT')

Modification History:

Written by:	Natascha Rudolf, October 2013.

A.11 RECRFITTELLRES

Name:

RECRFITTELLRES

Purpose:

This procedure recreates the output of FITTELL for the parameters specified.

Calling Sequence:

RECRFITTELLRES, Wave, Data, Error, Range, Gdasdata, Obsaltitude, Params, Waveresult, Result, Cleaned, Cleanederror

Inputs:

Wave:	Wavelength array of observed spectrum.
Data:	Flux array of observed spectrum.
Error:	Error of flux measurements.
Range:	Wavelength range of fit as two-element array in the form [start,end].
Gdasdata:	Scalar string containing absolute path to GDAS file.
Obsaltitude:	Altitude angle of observation in deg.
Params:	Fitted parameters ([smoothing_FWHM, velocity_shift, slope_of_straight, intercept_of_straight, water_abundance, methane_abundance, carbondioxide_abundance, oxygen_abundance]).

Keyword Parameters:

SILENT:	Set this keyword to suppress informational messages.

Outputs:

Waveresult:	Wavelength array of the fitted part.
Result:	Model fitted with all parameters (to be compared to observation).
Cleaned:	Telluric line removed spectrum.
Cleanederror:	Error of flux measurements divided by transmission model.

Example:

RECRFITTELLRES, Wave, Data, Error, [9000.,9300.], '/here/lies/the/Gdasdata', 87.2, Params, Waveresult, Result, Cleaned, Cleanederror

Modification History:

Written by:	Natascha Rudolf, October 2013.

A.12 RUNLBLRTM

Name:

RUNLBLRTM

Purpose:

This procedure creates a TAPE5 using the GDAS data and the MIPAS model atmosphere and runs LBLRTM to obtain a transmission spectrum in the specified wavelength range.

Calling Sequence:

RUNLBLRTM, Wavestart, Waveend, Gdasdata, Obsaltitude, Wavetransmission, Transmission

Inputs:

Wavestart:	Start wavelength in Å (scalar float or double).
Waveend:	End wavelength in Å (scalar float or double).
Gdasdata:	Scalar string containing absolute path to GDAS file.
Obsaltitude:	Altitude angle of observation in deg (scalar float or double).

Keyword Parameters:

OBSELEVATION:	Elevation above sea level in km of observatory site, default is 2.648 for VLT at Paranal.
WATER:	Water abundance (scalar float or double) relative to model abundance.
METHANE:	Methane abundance (scalar float or double) relative to model abundance.
OXYGEN:	Oxygen abundance (scalar float or double) relative to model abundance.
CARBONDIOXIDE:	Carbondioxide abundance (scalar float or double) relative to model abundance.
AMMONIA:	Ammonia abundance (scalar float or double) relative to model abundance.
NITRICOXIDE:	Nitricoxide abundance (scalar float or double) relative to model abundance.
OZONE:	Ozone abundance (scalar float or double) relative to model abundance.
SILENT:	Set this keyword to suppress informational messages by LBLRTM.
KEEP:	Set this keyword to keep the produced TAPE files.

Outputs:

Wavetransmission:	Wavelength in vacuum (double array)
Transmission:	Calculated transmission spectrum (double array)

Common Blocks:

TELLREM_INFO:	This common block contains relevant folder names and strings for running TELLREM. It has to be initialised by running LOADTELLREMINFO.

Restrictions:

LBLRTM can only calculate a piece of less then 2020 cm^{-1} in width.

Example:

A simple call looks like this:
RUNLBLRTM, 6000., 6500., '/here/is/the/GDAS/file', 79.8, wt, t
If you want to adjust the abundance of water and suppress informational messages, use this:
RUNLBLRTM, 6000., 6500., '/here/is/the/GDAS/file', 79.8, wt, t, water=0.8, /silent
wt contains the wavelength array and t the transmission spectrum.
The abundance keywords work relative to the model abundance, i.e. setting water=1. means using the water abundance from the atmospheric model, setting water=0.8 means lowering the abundance from the atmospheric model by 20 %.

Modification History:

Written by:	Natascha Rudolf, October 2013.
N. Rudolf, Dec. 2014	Changed abundance keyword check to allow setting them to 0.

A.13 STATUSINTERPRETER

Name:

STATUSINTERPRETER

Purpose:

This function maps the status code of MPFITFUN to a text explaining its meaning.

Calling Sequence:

text = STATUSINTERPRETER(Status)

Inputs:

Status:	The integer status code returned by MPFITFUN.

Outputs:

This function returns a string explaining the meaning of the status code MPFITFUN set.

Example:

textmeaning = STATUSINTERPRETER(1)

Modification History:

Written by:	Natascha Rudolf, October 2013.

A.14 SUBZERO

Name:

SUBZERO

Purpose:

This function replaces zero or subzero elements of the input array by the mean of the neighbouring points.

Calling Sequence:

Result = SUBZERO(Inputarray)

Inputs:

Inputarray:	Array to be treated.

Outputs:

This function returns the array containing no zero or subzero elements.

Example:

outputarray = subzero(inputarray)

Modification History:

Written by:	Natascha Rudolf, October 2013.

A.15 TELL

Name:

TELL

Purpose:

This function calculates a model transmission spectrum with the abundances stated and adjusts it by convolving with a Gaussian, shifting in wavelength and multiplying by a straight line so that it can be compared to an observed spectrum.

Calling Sequence:

model = TELL(Wavelength,Parameters,Gdasdata=Gdasdata,Obsaltitude=Obsaltitude)

Inputs:

Wavelength:	Wavelength array in Å for which model is requested.
Parameters:	Array of parameters of model ([smoothing_FWHM, velocity_shift, slope_of_straight, intercept_of_straight, water_abundance, methane_abundance, carbondioxide_abundance, oxygen_abundance]).

Required Keyword Parameters:

GDASDATA:	Scalar string containing absolute path to GDAS file.
OBSALTITUDE:	Altitude angle of observation in deg.

Keyword Parameters:

SILENT:	Set to 1 if you do not want to get informational messages.

Outputs:

This function returns an array of the same size as wavelength containing the transmission spectrum modelled and adjusted according to the parameters.

Common Blocks:

MERKEN:	This common block saves the parameters and results to avoid having to run LBLRTM too often.

Example:

mod = TELL(wave_array, [0.5,0.1,-5.8,2.,1.0,1.0,1.0,1.0], gdasdata='/here/lies/the/Gdasdata', obsaltitude=82.2)

Modification History:

Written by:	Natascha Rudolf, October 2013.

A.16 TELLREM

Name:

TELLREM

Purpose:

This procedures creates telluric line removed spectra using the information specified by the info file.

Calling Sequence:

TELLREM,Infofile,Telluricremovedspectra

Inputs:

Infofile:	Absolute path to file containing the necessary information to run the procedures of the `tellrem` package. An example file called 'info_for_tellrem', which you can adjust for your needs, is provided with the package. The format must be adhered to.

Keyword Parameters:

NOSAVE:	Set this keyword if you do not want the results to be saved to an IDL .sav file.
NOPLOT:	Set this keyword if you do not want plots of the results to be created.

Outputs:

This procedure returns an array of structures 'Telluricremovedspectra' containing the telluric line removed spectra and additional information for each object. For each object the structure consists of the tags 'object' containing the object name in a string, 'obsdate' containing the JD observation date as double scalar, 'obsaltitude' containing the altitude angle of the observation as double scalar, 'exptimeu' containing the exposure time in the UVB arm as double scalar, 'exptimev' containing the exposure time in the VIS arm as double scalar, 'exptimen' containing the exposure time in the NIR arm as double scalar, 'wclu' containing the wavelength array for the UVB data, 'clu' containing the flux array of UVB, 'eclu' containing the error array of the UVB flux, 'wclv' containing the wavelength array for the VIS data, 'clv' containing the flux array of VIS, 'eclv' containing the error array of the VIS flux, 'wcln1' containing the wavelength array for the NIR data in region 1, 'cln1' containing the flux array of NIR in region 1, 'ecln1' containing the error array of the NIR flux in region 1, 'wcln2' containing the wavelength array for the NIR data in region 2, 'cln2' containing the flux array of NIR in region 2, 'ecln2' containing the error array of the NIR flux in region 2, 'wcln3' containing the wavelength array for the NIR data in region 3, 'cln3' containing the flux array of NIR in region 3, 'ecln3' containing the error array of the NIR flux in region 3. The NIR arm spectrum is divided into 3 parts omitting the heavily absorbed parts.

Example:

TELLREM,'/here/lies/info_for_tellrem',Telluricremovedspectra

Modification History:

Written by:	Natascha Rudolf, October 2013.

A.17 TELLREMPARAMETERS

Name:

TELLREMPARAMETERS

Purpose:

This function determines the parameters needed to create a telluric line model for the individual observations.

Calling Sequence:

Result = TELLREMPARAMETERS(Objects)

Inputs:

Objects:	String array containing the object names that are to be treated, i.e. the names of the folders that contain the data of the individual objects.

Keyword Parameters:

PLOTNAME:	String containing the name of the file holding the plots. It is named plotnameobjectname.ps. Default is ’tellrem’.
NOPLOT:	Set this keyword if you do not want the fit results to be plotted.
NOSAVE:	Set this keyword if you do not want the parameters to be saved into an IDL .sav file.
SAVNAME:	String containing the name of the .sav file. It will be called savname.sav, default is ’tellremparams’.
SILENT:	Set this keyword if you do not want informational message about the progress to printed to the terminal.

Outputs:

This function returns an array of structures containing the parameters fitted to the spectrum and additional information for each object. For each object the structure consists of the tags ’object’ containing the object name in a string, ’obsdate’ containing the JD observation date as double scalar, ’obsaltitude’ containing the altitude angle the observation was taken at as double scalar, ’exptimev’ containing the exposure time in the VIS arm as double scalar, ’exptimen’ containing the exposure time in the NIR arm as double scalar as well as ’abundance_h2o’, ’abundance_co2’, ’abundance_ch4’, and ’abundance_o2’, that are structures containing the fit results of the individual regions used to determine the abundances of the main telluric contributors H_2O, CO_2, CH_4, and O_2, constructed like ’para’ (see below).
Additionally, it contains the 55-element array of structures ’para’. The individual structures consist of the tags ’range’ containing the start and end of the region the parameters stored in ’p’, an 8-element double array, were determined for. ’pe’, an 8-element double array too, contains the errors of those. The parameters are [smoothing_FWHM,velocity_shift, slope_of_straight, intercept_of_straight, water_abundance, methane_abundance, carbondioxide_abundance, oxygen_abundance] used in TELL.

Common Blocks:

TELLREM_INFO:	This common block contains relevant folder names and strings for running TELLREM. It has to be initialised by running LOADTELLREMINFO.

Example:

params=TELLREMPARAMETERS([’Object1’,’Object2’,’Object3’])

Modification History:

Written by:	Natascha Rudolf, October 2013.
N. Rudolf, August 2014	Changed extraction of keyword for altitude of observation
N. Rudolf, Nov. 2014	Added tags to the tellremparameters structure to save the fit results of the individual regions used to determine the abundances of the main telluric molecules

A.18 TELLREMSPECTRA

Name:

TELLREMSPECTRA

Purpose:

This function takes the parameters determined by TELLREMPARAMETERS and uses the modelled telluric line spectrum to remove the telluric lines from the observed spectrum.

Calling Sequence:

Result = TELLREMSPECTRA(Tellremparameters)

Inputs:

Tellremparameters:	The array of structures returned by TELLREMPARAMETERS.

Keyword Parameters:

SILENT:	Set this keyword if you do not want informational message about the progress printed to the terminal.
SAVNAME:	String containing the name of the .sav file. It will be called savname.sav, default is 'spectra_tellrem'.
NOSAVE:	Set this keyword if you do not want the resulting spectra to be saved to an IDL .sav file.

Outputs:

This function returns an array of structures containing the telluric line removed spectra and additional information for each object. For each object, the structure consists of the tags 'object' containing the object name in a string, 'obsdate' containing the JD observation date as double scalar, 'obsaltitude' containing the altitude angle of the observation as double scalar, 'exptimeu' containing the exposure time in the UVB arm as double scalar, 'exptimev' containing the exposure time in the VIS arm as double scalar, 'exptimen' containing the exposure time in the NIR arm as double scalar, 'wclu' containing the wavelength array for the UVB data, 'clu' containing the flux array of UVB, 'eclu' containing the error array of the UVB flux, 'wclv' containing the wavelength array for the VIS data, 'clv' containing the flux array of VIS, 'eclv' containing the error array of the VIS flux, 'wcln1' containing the wavelength array for the NIR data in region 1, 'cln1' containing the flux array of NIR in region 1, 'ecln1' containing the error array of the NIR flux in region 1, 'wcln2' containing the wavelength array for the NIR data in region 2, 'cln2' containing the flux array of NIR in region 2, 'ecln2' containing the error array of the NIR flux in region 2, 'wcln3' containing the wavelength array for the NIR data in region 3, 'cln3' containing the flux array of NIR in region 3, 'ecln3' containing the error array of the NIR flux in region 3. The NIR arm spectrum is divided into 3 parts omitting the heavily absorbed parts.

Common Blocks:

TELLREM_INFO:	This common block contains relevant folder names and strings for running TELLREM. It has to be initialised by running LOADTELLREMINFO.

Example:

tellremspecs = TELLREMSPECTRA(Tellremparams)

Modification History:

Written by:	Natascha Rudolf, October 2013.

A.19 TELLREM_TEST_RUN

Name:

TELLREM_TEST_RUN

Purpose:

This procedure performs a test run of the `tellrem` package using exemplary data provided in the package to check whether the installation of the related software was successful.

Calling Sequence:

TELLREM_TEST_RUN, Packagefolder, Executablename, Mipasmodel

Inputs:

Packagefolder:	String containing the path to where the `tellrem` package folder is located.
Executablename:	String containing the name of LBLRTM executable.
Mipasmodel:	String containing the absolute path to the MIPAS model atmosphere.

Outputs:

None. But creates plot file Tellrem_test_run_comparison.ps comparing the freshly determined telluric line removed spectrum to the spectrum provided in the package.

Example:

TELLREM_TEST_RUN, '/the/tellrem/package/is/here/', 'lblrtm_executable', '/MIPAS/model/is/here/equ.atm'

Modification History:

Written by:	Natascha Rudolf, December 2013.

A.20 TELLWITHOUT

Name:

TELLWITHOUT

Purpose:

This function calculates a model transmission spectrum with the abundances stated and adjusts it by convolving with a Gaussian and shifting in wavelength to enable its use to remove telluric lines from an observed spectrum.

Calling Sequence:

model = TELLWITHOUT(Wavelength, Parameters, GDASDATA=gdasdata, OBSALTITUDE = obsaltitude)

Inputs:

Wavelength:	Wavelength array in Å for which model is requested.
Parameters:	Array of parameters of model ([smoothing_FWHM, velocity_shift, slope_of_straight, intercept_of_straight, water_abundance, methane_abundance, carbondioxide_abundance, oxygen_abundance]).

Required Keyword Parameters:

GDASDATA:	Scalar string containing absolute path to GDAS file.
OBSALTITUDE:	Altitude angle of observation in deg.

Keyword Parameters:

SILENT:	Set to 1 if you do not want to get informational messages.
WORIG:	Wavelength array of unconvolved model.
TORIG:	Unconvolved model.

Outputs:

This function returns an array of the same size as wavelength containing the transmission spectrum modelled and adjusted according to the parameters but without using the straight line that TELL used to fit the model to the data.

Example:

mod = TELLWITHOUT(wave_array, [0.5,0.1,-5.8,2.,1.0,1.0,1.0,1.0], gdasdata='/here/lies/the/Gdasdata', obsaltitude=82.2)

Modification History:

Written by:	Natascha Rudolf, October 2013.

A.21 UVEXTINCTION

Name:

UVEXTINCTION

Purpose:

This function calculates the UV extinction using an LBLRTM model for the stated parameters.

Calling Sequence:

Result = UVEXTINCTION(Tellremparameters,Wave,Gdasdata)

Inputs:

Tellremparameters:	Structure for the individual object from the array of structures created by TELLREMPARAMETERS.
Wave:	UVB wavelength array of the object.
Gdasdata:	Scalar string containing absolute path to GDAS file.

Outputs:

This function returns the extinction in the UVB region appropriate for the input LBLRTM model parameters at the wavelengths given by wave. Since the model only reaches down to 3672 Å, the extinction is extrapolated below that value.

Example:

uvext = UVEXTINCTION(Tellremparameters_object,Wave,'/here/lies/the/Gdasdata')

Modification History:

Written by:	Natascha Rudolf, October 2013.

A.22 WHICHGDAS

Name:

WHICHGDAS

Purpose:

This function selects the GDAS sounding data file that lies closest to the observation date.

Calling Sequence:

Result = WHICHGDAS(Obsdate)

Inputs:

Obsdate:	Julian date of the observation (scalar float or double).

Outputs:

Result:	Scalar string with absolute path of the closest GDAS sounding data file.

Common Blocks:

TELLREM_INFO:	This common block contains relevant folder names and strings for running TELLREM. It has to be initialised by running LOADTELLREMINFO.

Example:

filename = WHICHGDAS(2456590.9276736d)

Modification History:

Written by:	Natascha Rudolf, October 2013.

Appendix B

Annotated example `TAPE5`

```
; record 1.1:  $ signals LBLRTM to start, rest is comment
$ TAPE5 created by runlblrtm.pro
; record 1.2
    1    1    1    0    1    0    0    0    0    1    0    0    0    0    5    5
; record 1.3
  9089.909 10001.000     4.000 0.000E-00     0.000    0.0000 2.000E-04 1.000E-03
; record 1.4
     0.000     0.000     0.000     0.000     0.000     0.000     0.000
; record 3.1:  user supplied model
    0    3    0    0    0   19    0          0.000    75.000     0.000   000.000
; record 3.2
   2.64800   0.00000   0.00000   0.00000   0.00000    0
; record 3.3A, use defaults
     2.000     5.000     8.000     0.000     0.000
; user defined atmospheric profile
; record 3.4
   50
; record 3.5
 9.160E-01 9.150E+02 2.874E+02     AA   GAAAAAAAAAA
; record 3.6.1 to 3.6.3
 1.080E+01 3.685E+02 2.231E-02 3.170E-01 9.173E-02 1.768E+00 2.120E+05 7.956E-06
 1.000E-04 6.260E-05 1.000E-04 1.781E-04
 0.000E+00 0.000E+00 6.000E-04
; repeating records 3.5 and 3.6.1 to 3.6.3 for all boundaries
 1.030E+00 9.000E+02 2.934E+02     AA   GAAAAAAAAAA
 4.300E+00 3.685E+02 2.280E-02 3.170E-01 9.114E-02 1.768E+00 2.120E+05 7.075E-06
 1.000E-04 5.074E-05 1.000E-04 1.884E-04
 0.000E+00 0.000E+00 6.000E-04
 1.518E+00 8.500E+02 2.909E+02     AA   GAAAAAAAAAA
-6.000E-01 3.685E+02 2.498E-02 3.170E-01 8.998E-02 1.768E+00 2.120E+05 5.372E-06
 1.000E-04 3.043E-05 1.000E-04 2.260E-04
 0.000E+00 0.000E+00 6.000E-04
 2.033E+00 8.000E+02 2.884E+02     AA   GAAAAAAAAAA
-6.200E+00 3.685E+02 2.722E-02 3.170E-01 8.950E-02 1.768E+00 2.120E+05 5.418E-06
 1.000E-04 3.356E-05 1.000E-04 2.495E-04
 0.000E+00 0.000E+00 6.000E-04
...................................................
................................................................................
........................................
..............................
```

```
 7.800E+01 1.507E-02 2.069E+02     AA   AAAAAAAAAAA
 3.573E+00 3.447E+02 2.028E-01 7.810E-04 1.956E+01 7.364E-02 2.120E+05 3.000E-03
 2.000E-04 1.244E-07 1.914E-11 2.639E-06
 0.000E+00 0.000E+00 1.091E-08
 8.300E+01 6.730E-03 2.045E+02     AA   AAAAAAAAAAA
 2.963E+00 3.295E+02 2.818E-01 6.327E-04 2.879E+01 5.510E-02 2.130E+05 4.762E-03
 2.000E-04 9.353E-08 1.914E-11 1.567E-06
 0.000E+00 0.000E+00 1.091E-08
 8.800E+01 2.931E-03 1.933E+02     AA   AAAAAAAAAAA
 2.471E+00 2.996E+02 5.553E-01 5.215E-04 3.826E+01 4.086E-02 2.130E+05 1.817E-02
 2.000E-04 9.000E-08 1.914E-11 9.149E-07
 0.000E+00 0.000E+00 1.091E-08
; signalling the end of the repetition
-1
; end of user defined atmospheric profile
; restart LBLRTM and make it write ASCII file of transmittance (TAPE99)
; record 1.1 again
$ Make ASCII
; record 1.2 again only activating ‘‘plotting’’
 HI=0 F4=0 CN=0 AE=0 EM=0 SC=0 FI=0 PL=1 TS=0 AM=0 MG=0 LA=0 MS=0 XS=0    0    0
; record 12.1
# Plot title not used
; record 12.2A
  9090.909 10000.000   10.2000  001.0000    1    0   12    0     1.000 0  0    0
; record 12.3A
    0.0000    1.2000    7.0200    0.2000    4    0    1    0    0    0 1    3 99
; terminate plotting
-1.
; % signals end of TAPE5 and terminates LBLRTM, rest comment
%
```