European Organisation for Astronomical Research in the Southern Hemisphere

Programme: GEN

Project/WP: Science Operation Software Department

VIPER Manual

Document Number: ESO-XXXXXX

Document Version: 1.0

Document Type: Manual (MAN)

Released on: 2024-01-08

Document Classification: Public

Prepared by: Jana Köhler, *viper* Pipeline Team

Validated by:

Approved by:

Name

Authors

Name	Affiliation
Jana Köhler	Thüringer Landessternwarte Tautenburg (TLS)

Change Record from previous Version

Affected Section(s)	Changes/Reason/Remarks
All	First draft

Contents

1 Introduction

1.1 Scope

The goal of the pipeline *viper* (Velocitiy and IP estimatoR) is to estimate high-precision Radial Velocities (RVs) for different instruments. Currently it is able to reduce spectra from seven instruments operating in the optical and NIR. *Viper* is able to deal with observations taken with a cell, including the modeling of the instrument profil (IP). Furthermore, it allows forward modeling of telluric lines, which is essentiel for NIR spectra. As part of it, *viper* is able to create telluric-free stellar templates, needed as reference for the RV estimation.

Viper is the official RV pipeline for the reduction of CRIRES+ data, wherefor the manual describes the reduction of these data in all detail. Still, a short overview of the reduction of other instruments is given.

1.2 Acknowledgements

The original idea of the *viper* project was initialized by Artie Hatzes. The development of the code was started by Mathias Zechmeister and Sireesha Chamarthi, whereby first is, together with Jana Koehler, still part of the development team.

1.3 Stylistic conventions

Throughout this document the following stylistic conventions are used:

bold	in text sections for commands and other user input which has to be typed as shown
italics	in the text and example sections for parts of the user input which have to be replaced with real contents
`teletype`	in the text for FITS keywords, program names, file paths, and terminal output, and as the general style for examples, commands, code, etc

In example sections expected user input is indicated by a leading shell prompt.

In the text **bold** and *italics* may also be used to highlight words.

1.4 Notational Conventions

Hierarchical FITS keyword names, appearing in the document, are given using the dot–notation to improve readability. This means, that the prefix “HIERARCH ESO” is left out, and the spaces separating the keyword name constituents in the actual FITS header are replaced by a single dot.

2 Definitions, Acronyms and Abbreviations

CPL	Common Pipeline Library
CCD	Charge Coupled Device
CRIRES	CRyogenic InfraRed Echelle Spectrograph
DFS	Data Flow System
DRS	Data Reduction System
ESO	European Southern Observatory
EsoRex	ESO Recipe Execution Tool
FITS	Flexible Image Transport System
FOV	Field Of View
GUI	Graphical User Interface
LSF	Line Spread Function
OB	Observation Block
pixel	picture element (of a raster image)
PSF	Point Spread Function
QC	Quality Control
RV	Radial Velocity
SDP	Science Data Product
SOF	Set Of Frames
VIPER	Velocity and IP EstimatoR
VLT	Very Large Telescope
WCS	World Coordinate System

3 Overview

For the detection and study of exoplanets, as well as for the study of stellar variabilities, RV shifts of stellar spectra are a helpful tool. During the last decades, a number of codes and techniques have been developed to archive the goal of reaching high precision RV measurements. Following this, *viper* is a user-friendly python-based code to obtain RVs from stellar spectra.

The basic purpose of *viper* is to have a software which can deal with all different kind of instruments, independent of the wavelength range or used cell. Hereby, it makes use of the algorithm already described by Butler et al. (1996), using imprinted lines from an iodine cell for the wavelength calibration. Thus, *viper* is able to deal with different cell types- like f.e. iodine or Th-Ar. A detailed description of the method and algorithms is given in Sect. 5.

Furthermore, by more and more instruments coming up in the NIR requiem, which allows to study even weak M dwarfs, the method is extended to allow the forward modeling of telluric lines. By this, *viper* is able to remove telluric lines from spectra or use them for the wavelength calibration besides of, or independently of, cell lines. Due to that, *viper* is able to reduce spectra observed in the optical, as well as spectra observed in the NIR. On top of this, it enables telluric corrections in the optical requiem if needed. It is hereby up to the user, if one wants to use just cell lines, or just telluric lines or both together in the modelling process.

Another important advantage of the telluric forward modeling is, that *viper* offers a routine to create own telluric-free stellar templates, which are needed as reference spectra for the RV estimation. The topic will be discussed in detail in Sect. 7.1.2 on the example of CRIRES+ data in K-band. By its fast and easy way of creating these templates, *viper* is a suitable alternative to other telluric correction codes. Even more, the user will be able to do the entire reduction, from the telluric-free template creation to the estimation of the final RVs, without the need of any further reduction codes.

By now, *viper* is already able to reduce data from 8 different instruments. Further instruments can easily be added, no matter if operating in the optical or NIR. And even if instruments from other wavelength ranges have not been tested yet, it is expected that *viper* also would be able to deal with this kind of data.

In Sect. 6 an overview of the demo data is given, which are offered to the user for test purposes.

A detailed description of the data reduction is given in Sect. 7. The reduction of CRIRES+ data can be found in Sect. 7.1, the reduction of the optical instruments in Sect. 7.2.

As part of that, the post-processing of the RV data and later inspection of the estimated parameters is described in Sect. 7.6.

An overview of the in- and output files can be found in Sect. 8.1. Sect. 8.2 contains tables describing all parameters used by *viper* in detail.

Additionally, in Sect. C.1 a short description of the pre-processing of CRIRES+ data is given, explaining how to reduce CRIRES+ data best using the DRS pipeline to obtain 1D spectra.

4 Installation

4.1 System Requirements

Viper is running with Python3, whereby following libaries have to be installed:

- argparse
- defaultdict
- astropy
- numpy
- scipy
- gnuplot
- tkinter (for GUI)
- PyCPL (just for CRIRES$^+$)

The PyCPL package can be found and downloaded from

https://www.eso.org/sci/software/pycpl/

4.2 Installing the *viper* Pipeline

For the installation just download *viper*, enter the directory and run

```
1> pip install -e .
```

Afterwards *viper* can be run with

```
1> viper [options]
```

For pytest, run following command from inside the *viper* directory:

```
1> python3 -m pytest -s
```

The options `-s` and `-m` are needed to avoid problems with used modules and writing the output files.

5 Algorithms

5.1 General Algorithms

The main concept of *viper* is using the forward modelling as described by Butler et al. (1996). Hereby a stellar reference template $I_{stellar}$ is multiplied with the cell FTS T_{cell} and afterwards convolved with the instrument profile *IP* as well as multiplied with a normalization k:

$$I_m = k\,[T_{cell}\,(\lambda)\;I_{stellar}\,(\lambda + \delta\lambda)] * \mathrm{IP} \tag{1}$$

Using a least-square fitting, the parameters of the normalization, wavelength solution, IP and RV shift are optimized in a way that the generated model I_m fits best the observation. This allows corrections of instrument instabilities, which is needed to archive high precision RV measurements. *viper* makes use of the function `curvefit` of the python library scipy to perform the least-square fitting.

This optimization process is performed on each spectral order, unless the user select smaller chunk sizes (`-chunk <number>`). Afterwards, a final RV value is calculated from the measurements of all orders of one observation:

$$RV = \frac{\sum rv_i w_i}{\sum w_i} \tag{2}$$

whereby the weighting is defined as the reciprocal of the square of the RV error of each order:

$$w_i = 1/e_i^2 \tag{3}$$

The uncertainty of the final RV can then be calculated by

$$e_{RV} = \sqrt{\frac{\sum w_i \cdot (RV - rv_i)^2}{(N-1)\cdot \sum w_i}} \tag{4}$$

with N being the number of selected orders.

5.1.1 Including telluric forward modelling

While for observations in the optical requiem the telluric lines of the atmosphere are mostly negligible, they are not in the infra-red (IR) or near-infra-red (NIR). One common way to deal with the tellurics is to use additional correction codes, like e.g. Molecfit (Smette et al. (2015)), to remove the tellurics from all observed spectra. While it would be still possible to do that and use the corrected data files as input, *viper* also offers the option of telluric forward modelling. To do so, eq. (1) is extended the following:

$$I_m = k\,[T_{cell}\,(\lambda)\;T_{atm}\,(\lambda + \delta\lambda_{atm})\;I_{stellar}\,(\lambda + \delta\lambda)] * \mathrm{IP} \tag{5}$$

Hereby, T_{atm} represents the telluric model, which is a product of the models from all present molecules in the given wavelength range. In case of K-band in the NIR, mainly water and methane play an important role, besides of some weaker molecules (Fig. 5.1):

$$T_{atm}\,(\lambda) = T_{H_2O}\,(\lambda)\;T_{CH_4}\,(\lambda)\;T_{CO_2}\,(\lambda)\;T_{N_2O}\,(\lambda) \tag{6}$$

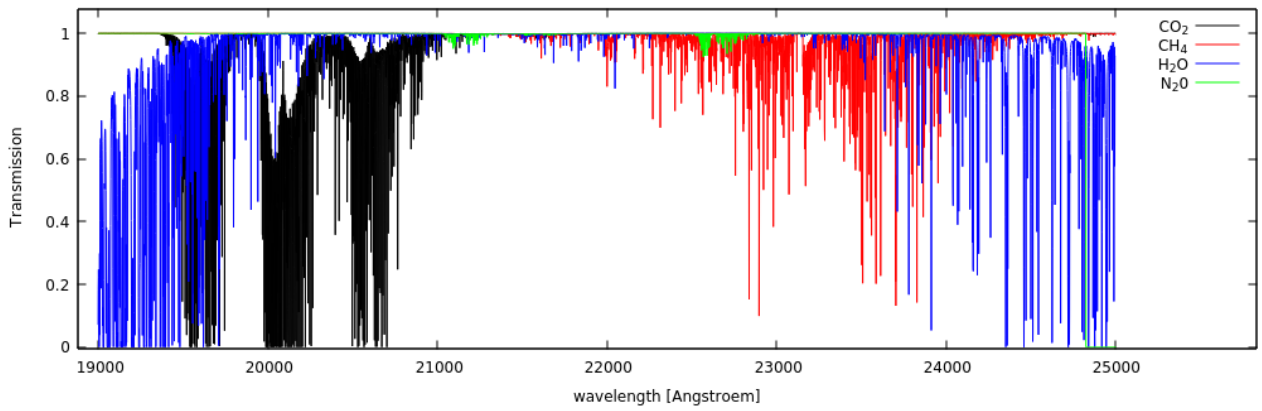

Figure 5.1: Transmission spectra of molecules in Earth atmosphere in K-band.

For the modelling, a synthetic standard model (produced with Molecfit) for each molecule with fixed airmass and PWV of 1 is used. During the optimization procedure, the scaling coefficient will be estimated for each molecule

$$T_{molecule}(\lambda) = T_{molecule}(airm = 1,\ pwv = 1)^{coeff} \tag{7}$$

The parameter $\delta\lambda_{atm}$ represent a wavelength shift of the telluric model, as the position of the lines is expected to be stable just down to 10 m/s (Figueira et al. (2010)). By using simultaneous cell lines as reference, *viper* is able to correct for these position shifts.

5.1.2 Using just telluric lines without cell

In case of weak stars, observations are often performed without the use of a gas cell to save signal. Herefore, *viper* offers the option of running the reduction without the modeling of the cell. Instead, especially for observations performed in the NIR (f.e. CRIRES+), telluric lines can be used as reference for the wavelength calibration. Also for orders in the optical, where no (iodine) cell lines are present anymore, it is possible to use telluric lines for the wavelength calibration. At least in orders, where enough telluric lines present. For this approach, eq. (9) will become:

$$I_m = k\,[T_{atm}(\lambda)\, I_{stellar}(\lambda + \delta\lambda)] * \mathrm{IP} \tag{8}$$

Note, that the position of the telluric lines should be set fixed if no additional cell lines are present and therefore `-tellshift` should NOT be used! Besides, the user should be aware that the telluric lines are just stable to around 10 m/s, which has an influence on the final RV precision, which is expected to be not better than that.

5.2 Creation of a telluric-free stellar template

For the creation of a telluric-free stellar template, several observations of one star are co-added. In best case, these observations are spread over a large time range (several months) to correct best for artefacts from the

telluric correction. First, for each observation a telluric model is created which fits best the observation. This is done by optimizing:

$$I_{atm} = k\,[T_{atm}\,(\lambda)] * \mathrm{IP} \tag{9}$$

Afterwards this telluric model is divided from the observed spectrum, leaving, in best case, nothing but the stellar lines.

For the co-adding, a weighted mean from all corrected spectra is used:

$$I_{stellar} = \frac{\sum I_i(x) w_i(x)}{\sum w_i(x)} \tag{10}$$

with the weighting of each pixel x depending on the line depth of the telluric line, present at this part of the spectrum and the errors per pixel of the observation $err_{obs}(x)$:

$$w_i = \frac{I_{atm}(x)}{[err_{obs}(x)\ /\ med(\ I_{obs}(x)\ /\ I_{atm}(x))]^2} \tag{11}$$

5.3 IP modeling

A major problem by the estimation of the best RV results, is the use of a correct IP model. *viper* offers different IP models, whereby the parameters of the IP are optimized during the modeling process. The IP models are normalized within *viper*. An overview plot of all available IP profiles is shown in Fig. 5.2. In the following, the IP profiles provided by *viper* are shortly introduced.

Gaussian `g`

A single Gaussian (`-IP g`) is the simplest version of an IP model. As the IP is in reality for most instruments more complicated, it is not recommended for every instrument as it can lead to wrong RV results (see Fig. 7.10). The Gaussian IP model is described via:

$$\mathrm{IP}(\sigma) = \exp\left(-\frac{x^2}{2\sigma^2}\right) \tag{12}$$

Super-Gaussian `sg`

Another version of a Gaussian is the super-Gaussian (`-IP sg`) with a more flat-top described by:

$$\mathrm{IP}(\sigma,\ p) = \exp\left(-\left|\frac{x}{\sigma}\right|^p\right) \tag{13}$$

Multiple, central Gaussian `mcg`

A combination of two central Gaussians can be selected by `-IP mcg`. While the positions are fixed, the width and amplitudes are variable:

$$\mathrm{IP}(\sigma,\ a) = \exp\left(-\frac{x^2}{\sigma^2}\right) + a \cdot \exp\left(-\frac{x^2}{16\sigma^2}\right) \tag{14}$$

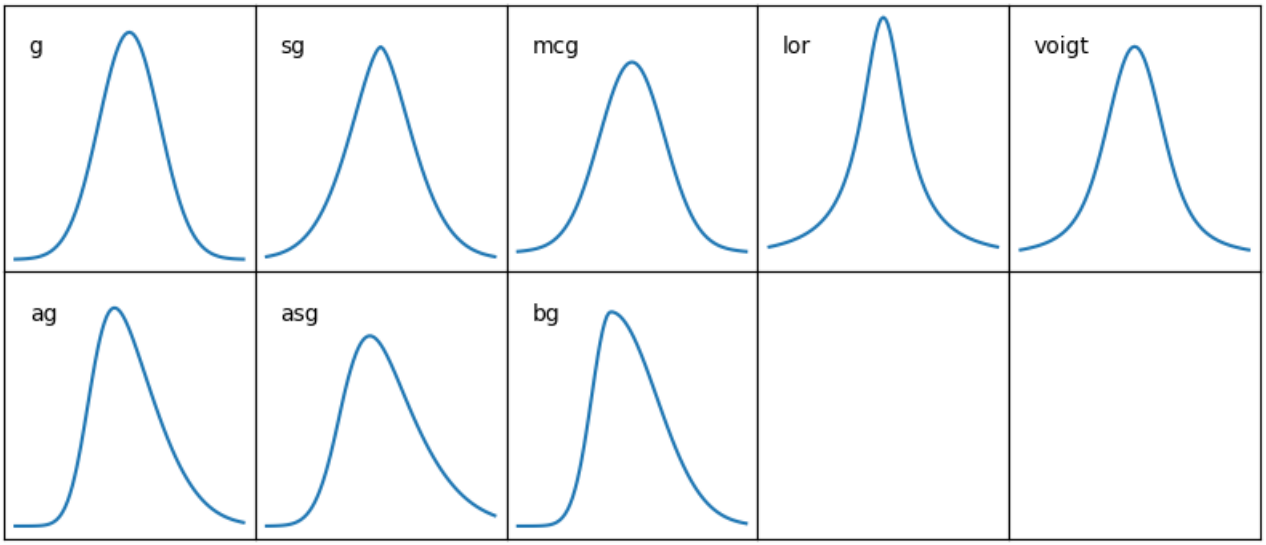

Figure 5.2: Example plots for the various IP models implemented in *viper*. Top: Symmetric IP models. Bottom: Asymmetric IP models.

Lorentzian profile `lor`

The Lorentzian profile `-IP lor` is defined by

$$\mathrm{IP}(\sigma) = \sqrt{\frac{1}{\pi}}\,\frac{|\sigma|}{x^2+\sigma^2} \tag{15}$$

Voigt profile `voigt`

The Voigt profile (`-IP voigt`) is the convolution of a Gaussian $G(\sigma)$ and a Lorentzian $L(\gamma)$:

$$\mathrm{IP}(\sigma,\,\gamma) = (G * L)\,(x) \tag{16}$$

Viper uses the Python libary scipy for the calculation.

Asymmetric (skewed) Gaussian `ag`

A single asymmetric Gaussian (`-IP ag`) is the simplest version of an asymmetric IP profile. Hereby a Gaussian is multiplied with the error function `erf` to create an assymmetry:

$$\mathrm{IP}(\sigma,\,a) = \exp\left(-\frac{(x/ss+b)^2}{2}\right)\cdot\left(1+erf\left(\frac{a\cdot x}{\sqrt{2}}\right)\right) \tag{17}$$

with

$$ss = \frac{\sigma}{\sqrt{1 - b^2}} \tag{18a}$$

$$b = \frac{a\sqrt{2/\pi}}{\sqrt{1 + a^2}} \tag{18b}$$

Asymmetric Super Gaussian `asg`

Similar, an Asymmetric Super Gaussian `-IP asg` can be generated by the multiplication with the error function:

$$\mathrm{IP}(\sigma,\, a,\, p) = \exp\left(-\left|\frac{(x + x_0)}{\sigma}\right|^p\right) \cdot \left(1 + erf\left(\frac{a \cdot (x + x_0)}{\sqrt{2}}\right)\right) \tag{19}$$

BiGaussian `bg`

Another simple way of creating an assymetric IP profile is the use of a BiGaussian (`-IP bg`). Hereby two Gaussian will be cut half and than combined to one. In this way, the width of the final profile varies for the left and for the right side of the curve:

$$\mathrm{IP}(\sigma_1,\, \sigma_2) \;=\; \begin{cases} \exp\left(-\frac{(x+x_0)^2}{2\sigma_1^2}\right) & \text{for } x + x_0 > 0 \\ \exp\left(-\frac{(x+x_0)^2}{2\sigma_2^2}\right) & \text{for } x + x_0 \leq 0 \end{cases} \tag{20}$$

with

$$x_0 = \sqrt{\frac{2}{\pi}}\,\frac{\sigma_2^2 - \sigma_1^2}{\sigma_1 + \sigma_2} \tag{21}$$

6 Demo Data

Two sets of demo data are available. One is for the TLS instrument, reperesent data reduction of optical data, the other is for the CRIRES+ instrument, representing data in the NIR.

Due to the large size of the demo data, they are stored at a seperated place. Follow the link from the website or github page for downloading the demo data.

6.1 CRIRES+

A set of demo data is given for the star GJ588 from a single setting (K2192) in `/data/CRIRES`, including following files:

- combined nodding spectra with cell (`/data/CRIRES/GJ588/withSGC/*.fits`)
- combined nodding spectra without cell (`/data/CRIRES/GJ588/noSGC/*.fits`)
- telluric-free template (`/data/CRIRES/GJ588/noSGC/GJ588_test_tpl.fits`)

The data were reduced using V1.3.0 of the DRS CR2RES pipeline.

6.2 TLS

A set of demo data is given for the star HD189733 is stored in `/data/TLS`, including following files:

- spectra with cell (`/data/TLS/HD189733/*.fits`)
- template spectrum without cell (`/data/TLS/HD189733_tpl/HD189733.model`)
- HARPS template (`/data/TLS/HD189733_tpl/HARPS*.fits`)

7 Data Reduction

Viper is able to reduce stellar spectra from different instruments. Yet, the reduction procedure does not vary much, and mainly depends on the used wavelength requiem. With demo data being available for one instrument performing in the optical (TCES/TLS) and one in the NIR (CRIRES+), this chapter shows the suggested reduction procedure for both of them.

Note: Some important information regarding the example commands used in this chapter:

- Using the commands below will show the plots as shown in the figures. To switch off all plotting remove `-lookfast` and/or `-lookctpl` from the example commands.
- The software is coming together with a default configure file setting values for a number of parameters. For some reduction procedures, the option of calling *viper* with and without the use of the configure file is shown.
- The examples in the following sections using demo data, which can be downloaded separately. In the commands it is expected that these data are stored in a data/ directory inside a directory from which viper is run.
- For CRIRES$^+$, just data in setting K2192 have been used for the examples. Some parameters or usable orders may can vary for other settings.

For more details, see chapter 8.

7.1 Reduction of CRIRES$^+$ data

As CRIRES$^+$ (Dorn et al., 2023) operates in the NIR, most of the spectral range is contaminated by strong absorptions lines of the earths atmosphere. Ignoring these telluric lines would lead to wrong RV results, as their position is not related to the ones of the stellar lines. For the RV estimation, as described in the following subsections, different options of dealing with telluric lines using *viper* are shown- whether by masking, down weighting or even forward modelling. All used data were observed in setting K2192 in K-band.

7.1.1 RV estimation

Applying no telluric correction

To give an introduction to the general RV estimation with *viper* and to show the influence of telluric lines, for the first example an observation using the short gas cell (withSGC) and one without using the cell (noSGC) are taken. In both cases, no telluric correction was applied. Here, the observation without imprinted cell lines will be used as the reference spectrum and therefore applies as the template. Two examples of the output are shown in Fig. 7.1 and can be produced with the following comment:

```
1> viper "data/CRIRES/GJ588/withSGC/22*" data/CRIRES/GJ588/noSGC/220723_1.fits
   -inst CRIRES -nset :11 -oset 12 -deg_norm 2 -deg_wave 2 -deg_bkg 1
   -kapsig 15 6 -oversampling 1 -output_format cpl -lookfast
```

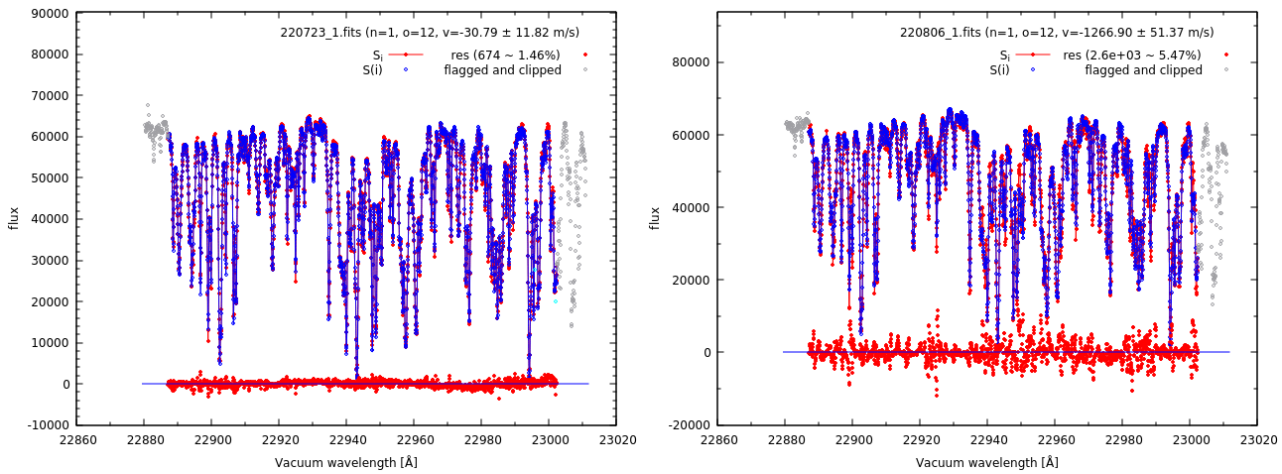

Figure 7.1: Output from *viper*. Left: Observation with and without cell are separated by just a few minutes. Right: Observation with and without cell are separated by seven days.

Depending on whether both observations are separated by just a few minutes (left plot of Fig. 7.1) or of several days (right plot Fig. 7.1), the effect of the tellurics becomes visible. If both observations were done shortly after each other, no problems or large residuals are expected for the modelling process. If both observations are separated by a longer time span, large residuals in the modelling process become visible, due to the different position shifts of stellar and telluric lines. It easily becomes clear, that ignoring the telluric lines is therefore not an option.

Masking and down-weighting telluric lines

In the optical requiem, a comment approach is to mask regions with imprinted telluric lines, which is as well one of the options offered by *viper* (`-telluric mask`). While this works fine for optical instruments, because of the low line density of tellurics, the situation is different in the NIR. As seen in the left plot of Fig. 7.2, a large part of the spectrum is flagged when using the masking of telluric lines. Besides of leaving large gaps in the spectrum, microtellurics will not be recognized with this technique, even so there will have an influence on the RV results. The left plot of Fig. 7.2 was created using:

```
1> viper "data/CRIRES/GJ588/withSGC/22*" data/CRIRES/GJ588/noSGC/220723_1.fits
   -inst CRIRES -nset :12 -oset 12 -deg_norm 0 -deg_wave 2 -deg_bkg 1
   -kapsig 15 6 -oversampling 1 -telluric mask-output_format cpl -lookfast
```

Here around 50% of the data will be masked by going down to a telluric transmission of 0.98. Using the same parameters but the option of weighting instead

```
1> viper "data/CRIRES/GJ588/withSGC/*" data/CRIRES/GJ588/noSGC/220723_1.fits
   -inst CRIRES -nset :1 -oset 12 -deg_norm 0 -deg_wave 2 -deg_bkg 1
   -kapsig 15 6 -oversampling 1 -telluric sig -tsig 0.001 -output_format cpl
   -lookfast
```

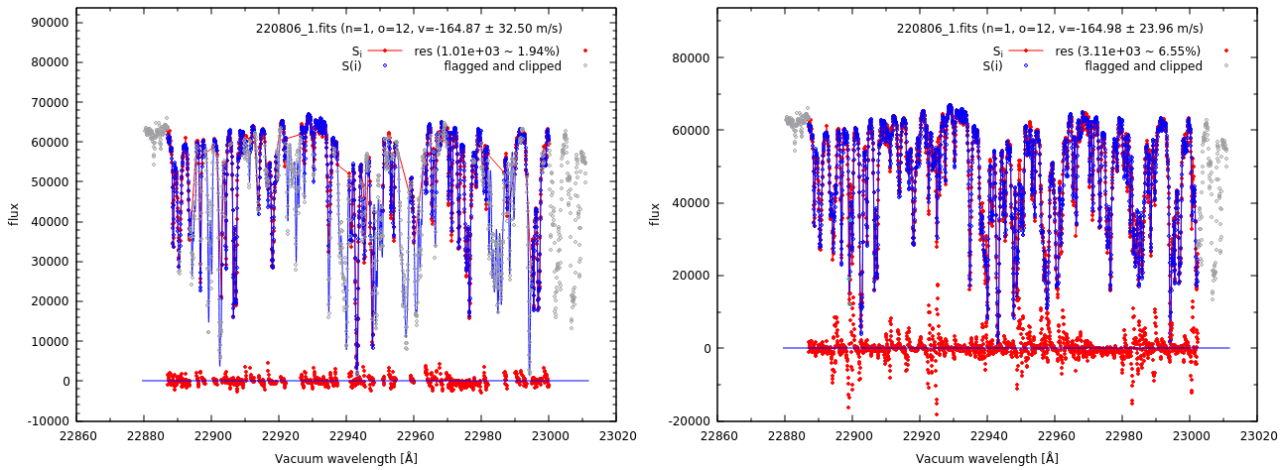

Figure 7.2: Output from *viper*. Observation with and without cell are separated by seven days. Left: Masking the tellurics. Right: Downweight the tellurics.

will end up in the right plot of Fig. 7.2. The option of weighting down the tellurics comes along with the parameter `tsig`. Hereby, a value of 1 is set as reference, meaning all values below 1 will down weight telluric lines, while values above 1 will up-weight telluric lines. For a value of one, regions with and without strong telluric lines will be weighted equally. Additionally, a value of 0.001, as used in the above example, will down weight the tellurics by a factor of 1000 in regard to telluric free regions.

Even the option of weighting (`-telluric sig`) will not solve the problem of the microtellurics, it avoids large gaps in the spectrum and therefore can lead to a slight improvement in the modelling process. Both ways would lead to similar RV results, though the errors are expected to be larger when masking the data. Nevertheless, it is easy to see that both options will not lead to optimal results, wherefor the telluric forward modelling, as described in the next subsection, is recommended over both of them.

Telluric forward modeling

If the user is in the possession of a telluric-free stellar template, but has observations where the telluric lines are still imprinted, *viper* offers the option `telluric add`. In this case, the telluric lines will be forward modelled and furthermore usable as an additional wavelength calibration besides of the present cell lines. Synthetic standard spectra from several molecules of the earths atmosphere will be used, without the need of any additional telluric correction code. The depth of the telluric lines will be scaled with a variable exponent, which is determined during the optimization process in *viper*. The procedure is described in more detail in Sect. 5.1.1. An example output from the following comment:

```
1> viper "data/CRIRES/GJ588/withSGC/*" data/CRIRES/GJ588/tpl/GJ588_test_tpl.fits
   -inst CRIRES -nset :12 -oset 7:17 -deg_norm 2 -deg_wave 2 -deg_bkg 1
   -kapsig 15 6 -oversampling 1 -telluric add -tellshift -tsig 1
   -flagfile <path to viper>/lib/CRIRES/flag_file.dat -output_format cpl -lookfast
```

can be seen in the plots of Fig. 7.3. The used telluric-free stellar template of GJ588 was generated as described

in Sect. 7.1.2. While both plots were created using the same stellar template, the observations with cell were separated by various months. Due to the forward modelling of the telluric lines, the residuals are low in both cases, demonstrating a successful modelling process.

Note that the parameter `tsig` is also available when using the telluric forward modelling. This can be especially helpful in case of an imperfect stellar template with left-over artefacts from the telluric removal. Values of `tsig` between 0.7 and 0.9 improved the results of the RV precision when telluric artefacts were present in the used stellar template. For more accurate templates, a value of 1 for `tsig` is recommended. It is up to the user, to find out the best value for the present observations.

The calculated RVs of a single spectral order (`-oset 12`) for the CRIRES+ demo data of GJ588 for all above introduced methods are plotted in Fig. 7.4. Unsurprisingly, the best precision with 11 m/s is reached using the

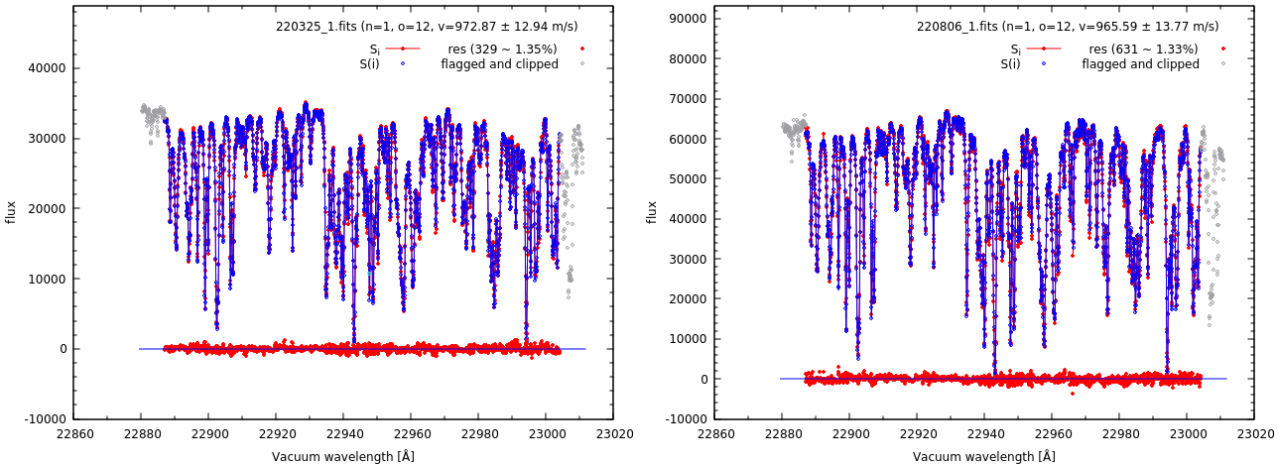

Figure 7.3: Output from *viper* using the telluric forward modelling. Both observations are separated by more than four months and are using the same telluric-free stellar template.

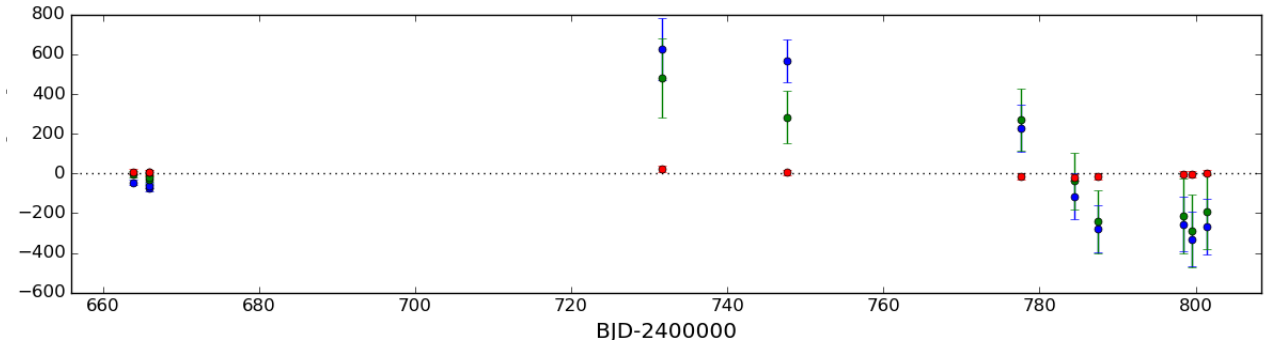

Figure 7.4: RV plot for one single spectral order for the different presented methods. The RV rms using the masking of tellurics (green) is 236 m/s, 318 m/s by downweighting tellurics with tsig = 0.001 (blue) and 11 m/s for the telluric forward modelling (red). For the masking and down-weighting an observation without cell was taken as reference template (from the first observation date). For the telluric-forward modelling a telluric-free template generated with *viper* was used instead. Besides, all other parameters have been the same for all reductions.

telluric forward modelling. In comparison, the RV precision for the same data is on the order of several 100 m/s by using the masking or down weighting of the telluric lines. Using a telluric-free stellar template and the option of masking the tellurics would lead to a rms of 22 m/s (not plotted). The present observations are spread over a time range of 5 months. By combining the RV results from several orders, the RV precision is expected to improve, as well as the error bars. By the current state, the RV estimation is most successful for `-oset 7:17` in setting K2192. It's being worked upon to improve the results for the remaining spectral orders.

It is as well possible to use just the telluric lines for the wavelength calibration, if no observations with imprinted cell lines are available. To be precise, this is exactly what we do, when using *viper* to create telluric-free templates from observations without cell, as described in the following subsection. For this, one uses the last comment above by adding the option `-nocell`. Nevertheless, the archived RV precision will never be as good as using additional cell lines, as the telluric lines are just stable down to around 10 m/s (Figueira et al. (2010).

7.1.2 Generating a telluric-free template

For the estimation of RVs with *viper*, a stellar reference template is needed. This template should contain no cell lines and, especially for data in the NIR, no telluric lines. *Viper* itself comes along with the option of creating such telluric-free templates. The possible options of template creation are explained in the following.

Using observations performed without a cell

As demonstrated in the previous subsection, the estimating RV precision is much better by using a telluric free template. If the user has not such a template to hand, *viper* offers a fast and easy option of generating such a template. The quality of the template is expected to improve by using observations distributed over the year.

As good as the telluric correction already works for weak and medium telluric lines, there are still problems for deep and strong telluric lines, which can lead to strong artefacts. By having observations over a long time range

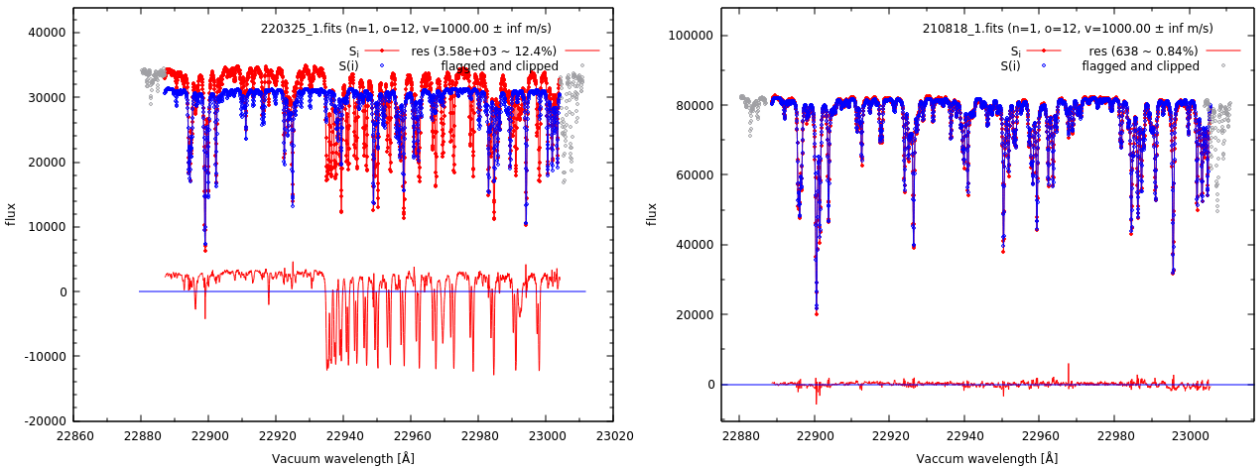

Figure 7.5: Output from *viper*. Telluric modelling and correction of one spectral order GJ588 (left) and the telluric standard star GamGru (right). Here, the residuals in the bottom represent the telluric free spectrum.

available, these artefacts will disappear in the final template by combining all observations.

For the removal of the telluric lines from the spectra, the same principle as described above and in Sect. 5.1.1 is used. After fitting for the best parameters, the modelled telluric spectrum is divided from the observed stellar spectrum. This is done for all observations and orders separately before all observations will be combined to a final template. The creation of the template can be called by:

```
1> viper "data/CRIRES/GJ588/noSGC/22*" -inst CRIRES -nset :12 -oset 7:17
   -deg_norm 0 -deg_wave 2 -deg_bkg 1 -oversampling 1 -createtpl -telluric add
   -tsig 10 -nocell -output_format cpl -lookfast -lookctpl
```

While no template is given in this case, the observations without cell will be entered as datafiles. As no cell lines are available, the option `-telluric add` is used together with `-nocell` to enable a wavelength calibration by using the imprinted telluric lines. The option `-createtpl` calls the function to create a combined template out of the selected observations. Please note that no correction for possible position shifts of the telluric spectrum is possible, as no cell lines for comparison are present. Even these shifts are expected to be just of the order of around 10 m/s, this can lead to errors in the final wavelength solution. With no other lines available to correct for instrument instabilities, this it at the current state still the best option available.

Note: For the telluric correction without using a reference template, the RV values are fixed to the value given by the `-rv_guess` parameter, which is by default 1 km/s. Due to that, the RV values printed and stored in the parameter files are all the same and the corresponding uncertainties are `inf`.

While testing, it turned out that better results are obtained by using the entire spectral order instead of flagging the noisy regions with the `-flagfile` option. This is especially the case for spectral orders with just a low number of telluric lines, where every existing telluric line in the spectral order is needed for a successful calibration. Still, for the final estimation of the RVs, it is recommended to remove the noisy regions for the reduction. Nevertheless, it is up to the user to play around with it and improving the existing flag file to resolve better results.

Fig. 7.5 demonstrate the results of the telluric removal. Testing the approach on the fast rotating B star GamGru (right plot), meaning mainly telluric lines are visible in the spectrum, to illustrate how well *viper* is able to model the telluric lines. The residuals in the bottom represent the telluric corrected stellar spectrum. The telluric corrected spectrum of the M dwarf GJ588 is shown in the left plot of the same figure. After the correction is applied on all individual observations, they will be combined to a final spectrum as shown in Fig. 7.6. The final spectrum in the bottom is calculated using the weighted mean of all above over-plotted spectra. The weighting is based on the location of the telluric lines as well as on the flux errors of the observations. In this way, artefacts and noise in the final template can be reduced. The generated fits file can afterwards be used as reference template to obtain the RVs from the observations with cell as described in Sect. 7.1.1

Note: The file name of the template generated with *viper* has to end on `_tpl.fits` to ensure *viper* is afterwards able to recognize its format and to read it in the right way.

Using observations performed with a cell

Similar to the above described method of generating a telluric-free template, it is also possible to use observations performed with a cell. This comes along with the advantage of a better wavelength calibration. Unfortunately, it also comes along with more lines that have to be removed from the observed stellar spectrum to

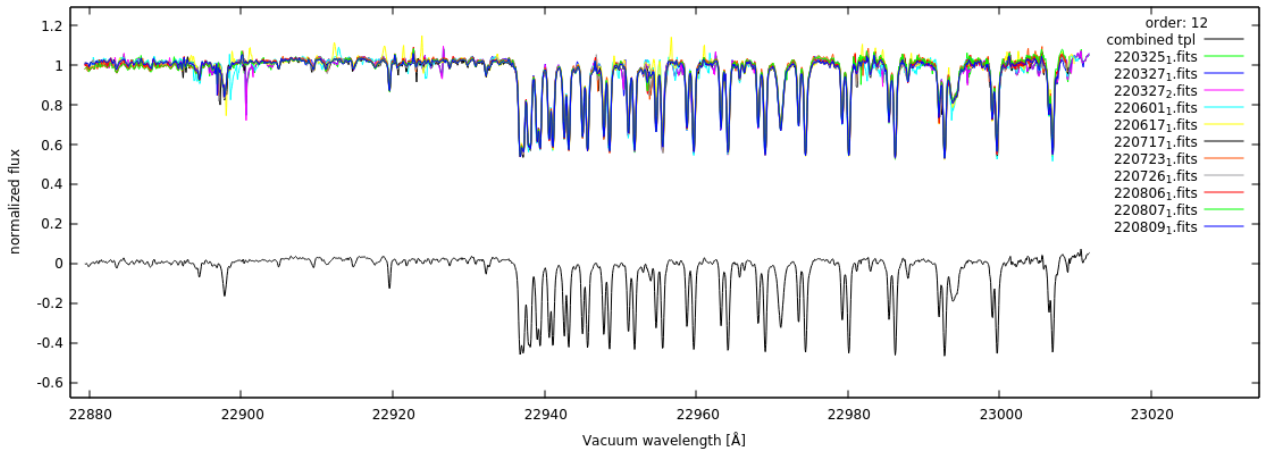

Figure 7.6: Output from *viper*. Combined template of GJ588 (black line on the bottom) created by using several telluric corrected spectra observed without cell (coloured lines in the top).

generate a template. This again can rise the number of artefacts, wherefore it is again recommended to use observations from different epochs.

The reduction with *viper* works as descriped in the previous section, just now the observations with cell are used as input:

```
1> viper "data/CRIRES/GJ588/withSGC/22*" -inst CRIRES -nset :12 -oset 7:17
   -deg_norm 0 -deg_wave 2 -deg_bkg 1 -oversampling 1 -createtpl -telluric add
   -tsig 10 -output_format cpl -lookfast -lookctpl
```

Further, the option `-nocell` is removed, as we now have to model the cell as well.

Improved telluric-free template creation - Using a stellar template

As the quality of the final RVs depends to a large amount on the quality of the used stellar template, it is worth spending time on the improvement of it. In the previous sub-section, for the creation of the telluric-free template, the command runs without the use of reference template.

In a next step we use the template, which was created in one of the previous sub-section, as an input reference template, while running again the template creation on all observations without cell via:

```
1> viper "data/CRIRES/GJ588/noSGC/22*" tmp_tpl.fits -inst CRIRES -nset :12
   -oset 7:17 -deg_norm 2 -deg_wave 2 -deg_bkg 1 -oversampling 1 -createtpl
   -telluric add -tsig 1 -kapsig 15 6 -nocell -output_format cpl -lookfast
   -lookctpl
```

Doing this comes along with two mayor benefits. First, by using a stellar reference spectrum, it is easier for the code to distinguish between stellar and telluric lines in the input spectrum. This leads in most cases to a better

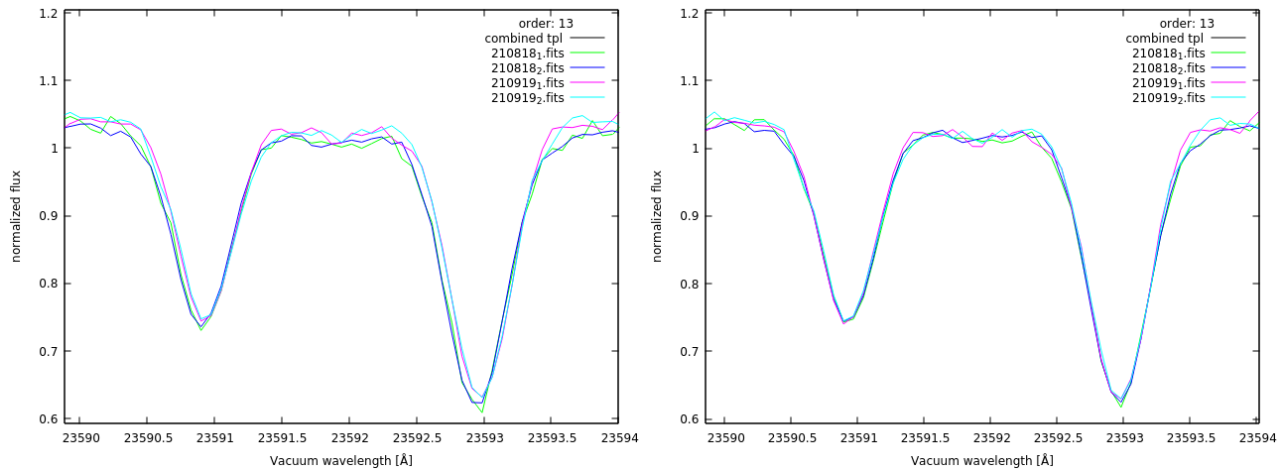

Figure 7.7: Output from *viper*. Template creation for TOI129 (star with exoplanet with K amplitude of around 750 m/s) by combining four observations. Left: No correction of stellar shifts due an existing exoplanet. Right: Improved template creation correcting for stellar shifts.

telluric modelling and therefore improves the telluric removal. The second benefit shows up when running the template creation on non RV standard stars. In case of RV standard stars, we expect no shifts in the stellar lines, which is not the case anymore for stars where we expect f.e. exoplanets. By the combination of several spectra to one, one has to take care of these shifts and has to correct for it. This is just possible by using a reference spectrum, related to which we are able to estimate line shifts. If the user wants to create a template for a star with expected strong RV shifts, it is recommended to first run the template creation of the previous section on one selected file and afterwards the improved template creation as described in this sub-section on the rest of the spectra.

By now using a reference input spectra, the changes of a few parameters can help to optimize the final results. A value larger 0 for `-deg_norm` allows correcting for small difference in the normalization of the spectra. Furthermore, using now a `-tsig` value of 1 will weight telluric and stellar lines equally and guarantees a better calculation of line shift between different observations.

While testing it turned out, that repeating the previous command several times on the data, always using the newly created template file as input, improves the quality of the final template as well as the quality of the latter archived RV data. For tests made, in most cases a number of two to three repetition gave the best results. Again, that doesn't mean that this will be the case for all data.

7.1.3 Recommended step by step reduction

In the following, a short step by step reduction to reduce CRIRES+ data with *viper* is given. This reduction procedure was giving the best results for most of the test data. Yet, it is not a guarantee and an update on several parameters can help to optimize the results for individual data. Nevertheless, it can be helpful as a first starting point.

If the user is interested to inspected the created template from several observations without using `-lookfast`,

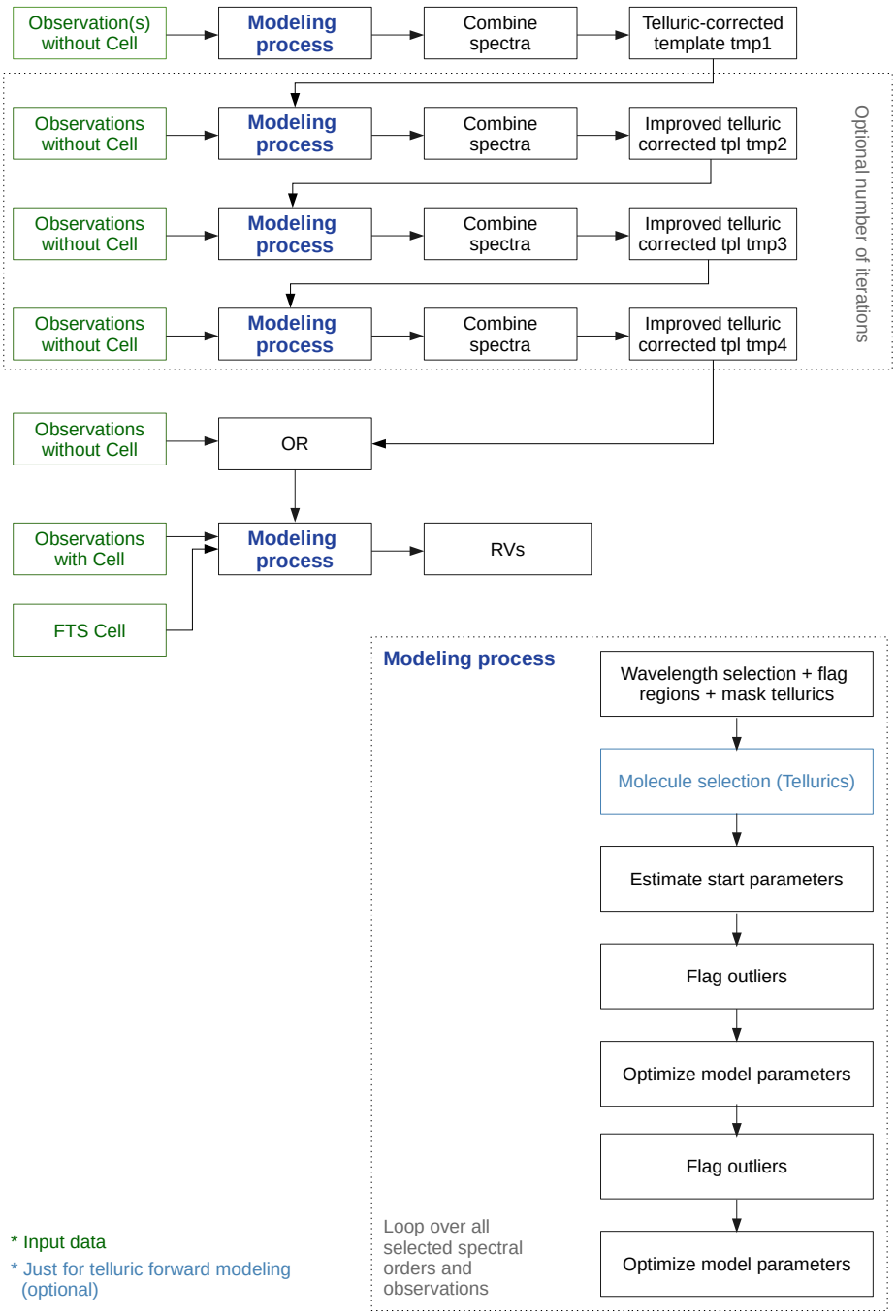

Figure 7.8: Output from *viper*.

the option `-lookctpl` can be used to show just the final results.

First template creation:

```
1> viper "data/CRIRES/GJ588/noSGC/22*" -inst CRIRES -nset :12 -oset 7:17
   -deg_norm 0 -deg_wave 2 -deg_bkg 1 -oversampling 1 -createtpl -telluric add
   -tsig 10 -vcut 10 -nocell -output_format cpl -lookfast -lookctpl - tag tmp1
```

Improved template creation- run 3 times:

```
1> viper "data/CRIRES/GJ588/noSGC/22*" tmp1_tpl.fits -inst CRIRES -nset :12
   -oset 7:17 -deg_norm 2 -deg_wave 2 -deg_bkg 1 -oversampling 1 -createtpl
   -telluric add -tsig 1 -vcut 10 -kapsig 15 6 -nocell -output_format cpl
   -lookfast -lookctpl -tag tmp2

2> viper "data/CRIRES/GJ588/noSGC/22*" tmp2_tpl.fits -inst CRIRES  -nset :12
   -oset 7:17 -deg_norm 2 -deg_wave 2 -deg_bkg 1 -oversampling 1 -createtpl
   -telluric add -tsig 1 -vcut 10 -kapsig 15 6 -nocell -output_format cpl
   -lookfast -lookctpl -tag tmp3

3> viper "data/CRIRES/GJ588/noSGC/22*" tmp3_tpl.fits -inst CRIRES -nset :12
   -oset 7:17 -deg_norm 2 -deg_wave 2 -deg_bkg 1 -oversampling 1 -createtpl
   -telluric add -tsig 1 -vcut 10 -kapsig 15 6 -nocell -output_format cpl
   -lookfast -lookctpl -tag tmp4
```

or alternatively (using CRIRES+ default values from configure file):

```
1> viper "data/CRIRES/GJ588/noSGC/22*" -inst CRIRES -output_format cpl
    -config_file CRIRES_tpl1 -nocell -nset :12 -tag tmp1

2> viper "data/CRIRES/GJ588/noSGC/22*" tmp1_tpl.fits -inst CRIRES
   -output_format cpl -config_file CRIRES_tpl2 -nocell -nset :12
   -tag tmp2

3> viper "data/CRIRES/GJ588/noSGC/22*" tmp2_tpl.fits -inst CRIRES
   -output_format cpl -config_file CRIRES_tpl2 -nocell -nset :12
   -tag tmp3

4> viper "data/CRIRES/GJ588/noSGC/22*" tmp3_tpl.fits -inst CRIRES
   -output_format cpl -config_file CRIRES_tpl2 -nocell -nset :12
   -tag tmp4
```

Once a template for a star was created, it can be saved and used for all further RV calculations.

RV estimation using the created template:

```
1> viper "data/CRIRES/GJ588/withSGC/*" tmp4_tpl.fits -inst CRIRES -nset :12
   -oset 7:17 -deg_norm 2 -deg_wave 2 -deg_bkg 1 -kapsig 15 6 -oversampling 1
   -telluric add -tellshift -tsig 1 -output_format cpl
   -flagfile <path to viper>/lib/CRIRES/flag_file.dat -lookfast
```

or alternatively (using CRIRES+ default values from configure file):

```
1> viper "data/CRIRES/GJ588/withSGC/22*" tmp2_tpl.fits -inst CRIRES
   -flagfile <path to viper>/lib/CRIRES/flag_file.dat -nset :12
   -tellshift -output_format cpl
```

Post-processing (see sub-section 7.6):

```
1> vpr tmp_rvo_par.fits -avg wmean -cen -ocen -save test.dat
```

The file test.dat contains the BJD of the observations, the weighted mean RVs and the corresponding errors (both in m/s). Further, it is possible to save it as a FITS file, f.e. `-save test.fits`.

7.2 Reduction of optical instruments using a cell - TCES

As all optical instruments supported by *viper* are following the same reduction steps with just minor differences, the description for the data reduction is summarized in this section. Thus, some comments regarding the parameter selection for various instruments is given in Sect. 7.4.

Other than in the NIR requiem, a forward modelling of the telluric lines is not necessarily needed. Even if it is possible, and tests are on-going to use it, it is sufficient to mask telluric lines. *viper* therefore offers the option `-telluric mask`, which make use of an existing mask file. Besides, the reduction of optical data is straight forward and simple with *viper*, as there is no need for the creation of a telluric free stellar template or complicated telluric treatment.

If an observation without cell, which can be used as a reference template, is available, the RVs can simply be obtained by following command:

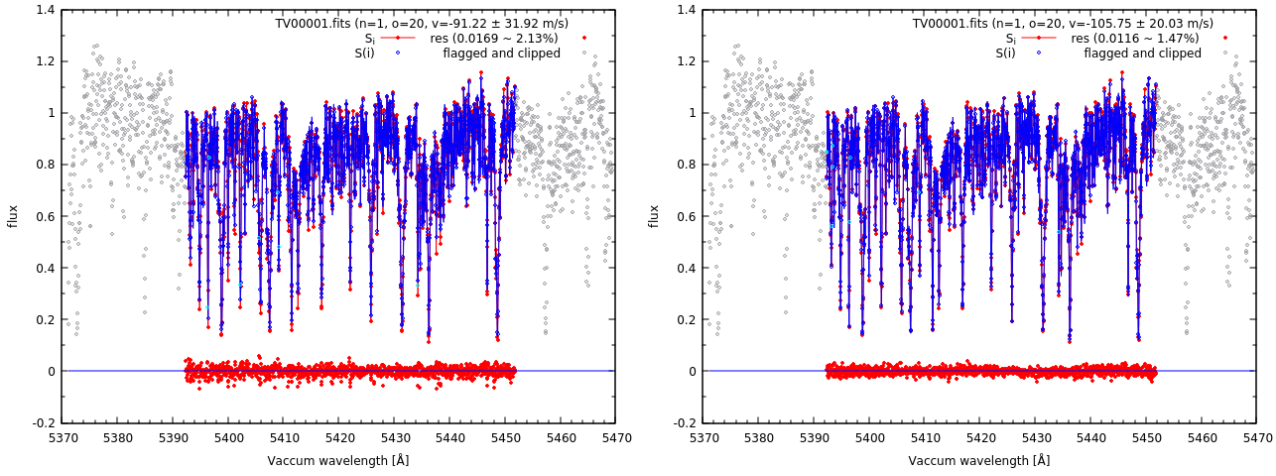

Figure 7.9: Output from *viper*. Left: Using cell free template from TLS. Right: Use HARPS data as template

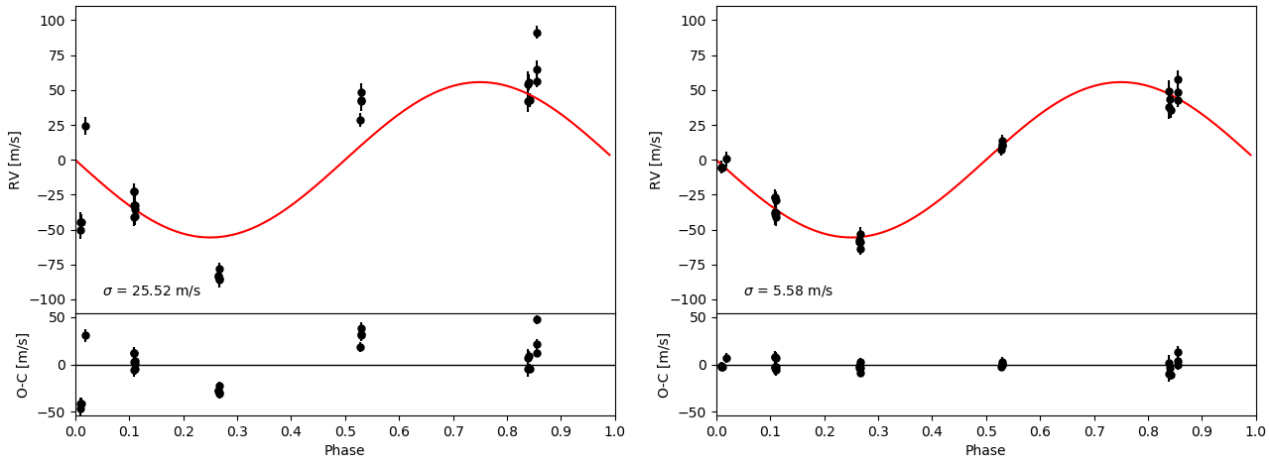

Figure 7.10: RV results from *viper* for 51Peg observed with TCES (TLS). Left: Using a single Gaussian IP model leading to a rms of 25.52 m/s. Right: Using a BiGaussian IP model resulting in an improved rms of 5.58 m/s. All other parameters stay the same.

```
1> viper "data/TLS/HD189733/TV*" data/TLS/HD189733_tpl/HD189733.model -inst TLS
    -nset :24 -oset 19:29 -deg_norm 3 -deg_wave 3 -deg_bkg 1 -ip bg -iphs 70
    -oversampling 1 -kapsig 4.5 -telluric mask -lookfast -targ HD189733
```

or alternatively (using TLS default values from configure file):

```
1> viper "data/TLS/HD189733/TV*" data/TLS/HD189733_tpl/HD189733.model -inst TLS
    -nset :24 -oset 19:29 -targ HD189733 -lookfast
```

Various tests with a number of instruments (f.e. TCES, OES) have shown that a better RV results can be reached by using the position of the stars as given by the SIMBAD catalogue (via `-targ <source>`) due to a possible wrong pointing accuracy. Furthermore, the use of a simple Gaussian IP model is not sufficient for most of the implemented optical instruments. Nevertheless, it turns out that the use of a BiGaussian (`-ip bg`) improves the final RV precision and leads to similar results than given by other codes using multi Gaussian IP models. A comparison of RV results for both IP models is shown in Fig. 7.10. Further tests on improving the implemented IP models is ongoing. The user is encouraged to find out the best IP model for the corresponding instrument. Nevertheless, a recommendation for a number of instruments based of various tests is given in Sect. 7.4. An overview of the IP models currently implemented in *viper* can be found in Sect. 5.3.

After the estimation of the RVs for all possible orders, the user can make use of the post-processing script vpr.py (Sect. 7.6) to select by hand the best orders and get rid of problematic orders to obtain the best results. Further, it is recommended to subtract the mean RV for each individual order to reduce the calculated error bars. This can be done via:

```
1> vpr tmp.rvo.dat -avg wmean -cen -ocen
```

7.3 Special remark: Order definition and selection

The precision of the obtained RV values depends largely on the line density of stellar, cell and telluric lines. *viper* has no automatic line counter and it is expected that the user is aware of which parts of the spectrum are useable and to select the corresponding spectral orders. As a help, one can use the vpr.py for the post-processing (Sect. 7.6 and right plot of Fig. 7.12) to study the quality of individual orders and optionally remove orders which are not giving reasonable results. One should be aware that this order selection can vary for different star types due to varying line density.

7.4 Recommendations for parameter selection

The choice of best parameters can vary for the different instruments available for *viper*. In the following section, some suggestions on selected parameters will be given, based on numerous tests on different data sets. The user should be aware that all of this is just a suggestion but not an assurance for getting the best results.

7.4.1 CRIRES+ data

Parameter	Suggestion
`chunks`	It is recommended to leave the value to 1 due to the low line density in most of the CRIRES+ wavelength range.
`deg_norm`	It is recommended to use a value of 0 for telluric forward modelling if no template file is given. With the use of a stellar template file, a value of 2 or 3 is sufficient.
`deg_wave`	It is recommended to use a value of 2. By now *viper* has problems modelling deg_wave larger that 2 for CRIRES+ data.
`deg_bkg`	It is recommended to leave that value always at 1.
`flagfile`	It is recommended to use a flagfile as some parts of the CRIRES+ spectrum still show noisy structures which can have an influence on the final results. *viper* comes along with one flagfile in the lib directory. This file may has to be modified by the user depending on the reduced data.
`iphs`	A value of 50 is sufficient.
`kapsig`	It is recommended to use a larger value for the first kapsig value when using telluric forward modelling, around 15. Using values from 4.5 to 6 for the second kapsig value were leading to the best results for most of the test data. When using no stellar template, kapsig should be set to 0.
`oversampling`	It is recommended and sufficient to use a value of 1.
`telluric`	It is recommended to use add or add2.
`tsig`	It is recommended to use values >= 1 for the template creation using telluric forward modelling.

continued on next page

	For the RV estimation using telluric forward modelling in most cases a value of 1 gives the best results. If the quality of the stellar template is poor and a lot of telluric artefacts are still present, values of 0.7 to 0.9 can improve the final results.

7.4.2 TCES data (TLS)

Parameter	Suggestion
`chunks`	Can be left to 1. A higher number of chunks does not seem to improve results. For low SNR data (f.e. TESS) a value of 1 is recommended.
`deg_norm`	Depending on the quality of normalization performed by the pre-processing, a value of 3 is sufficient for most data. In case of poor pre-normalization a value of 8 is recommended.
`deg_wave`	It is recommended to use a value of 3. Larger values may cause errors in *viper*.
`deg_bkg`	It is recommended to leave that value always at 1.
`ip`	Using bg (BiGaussian) seems to give best results for most data.
`iphs`	A value of 70 or 80 is recommended.
`iset`	A setting from 380:1700 is recommended due to high noise at the edge of the orders. It is deposited as default value for TLS data.
`kapsig`	In most cases a value of 4.5 leads to reasonable results.
`oversampling`	It is recommended and sufficient to use a value of 1.
`telluric`	It is recommended to use mask.

7.4.3 OES data (Ondrejov)

Parameter	Suggestion
`chunks`	Can be left to 1. A higher number of chunks does not seem to improve results. For low SNR data (f.e. TESS) a value of 1 is recommended.
`deg_norm`	Depending on the quality of normalization performed by the pre-processing, a value of 3 is sufficient for most data. In case of poor pre-normalization a value of 8 is recommended.
`deg_wave`	It is recommended to use a value of 3. Larger values may cause errors in *viper*.
`deg_bkg`	It is recommended to leave that value always at 1.
`ip`	Tests are on-going. Using g (Gaussian) or bg (BiGaussian) seems to give best results for most data.
`iphs`	A value of 80 is recommended.

continued on next page

kapsig	In most cases a value of 4.5 leads to reasonable results.
oversampling	It is recommended and sufficient to use a value of 1.
telluric	It is recommended to use mask.

7.4.4 Tull data (McDonald)

Parameter	Suggestion
chunks	Can be left to 1. A higher number of chunks does not seem to improve results. For low SNR data (f.e. TESS) a value of 1 is recommended.
deg_norm	Depending on the quality of normalization performed by the pre-processing, a value of 3 is sufficient for most data. In case of poor pre-normalization a value of 8 is recommended.
deg_wave	It is recommended to use a value of 3. Larger values may cause errors in *viper*.
deg_bkg	It is recommended to leave that value always at 1.
ip	Using bg (BiGaussian) seems to give best results for most data.
iphs	A value of 80 is recommended.
kapsig	In most cases a value of 4.5 leads to reasonable results.
oversampling	It is recommended and sufficient to use a value of 1.
telluric	It is recommended to use mask.

7.5 Graphical User Interface (GUI)

Viper is coming along with a GUI which can be used instead of running commands from the console. It can be started directly via

```
1> GUI_viper
```

or by using parameter values from configure file with

```
1> GUI_viper <configure file> <configure section>
```

After the window shown in Fig. 7.11 opens, the user can change the values of all parameters. Be aware that the appearance of some parameters depend on the setting of other parameters. So will the options of `tsig` and `tellshift` just show up after `telluric` has been set to `add` or `add2`. Also the additional options of the template creation will just show up after selecting `create tpl`. Some parameters also will be set automatically after selecting a specific instrument, like cell FTS file (if present) and the atmosphere molecules.

Running the mouse cursor over the parameters, a help text is printed. These help texts are the same as printed when running

```
1> viper -help
```

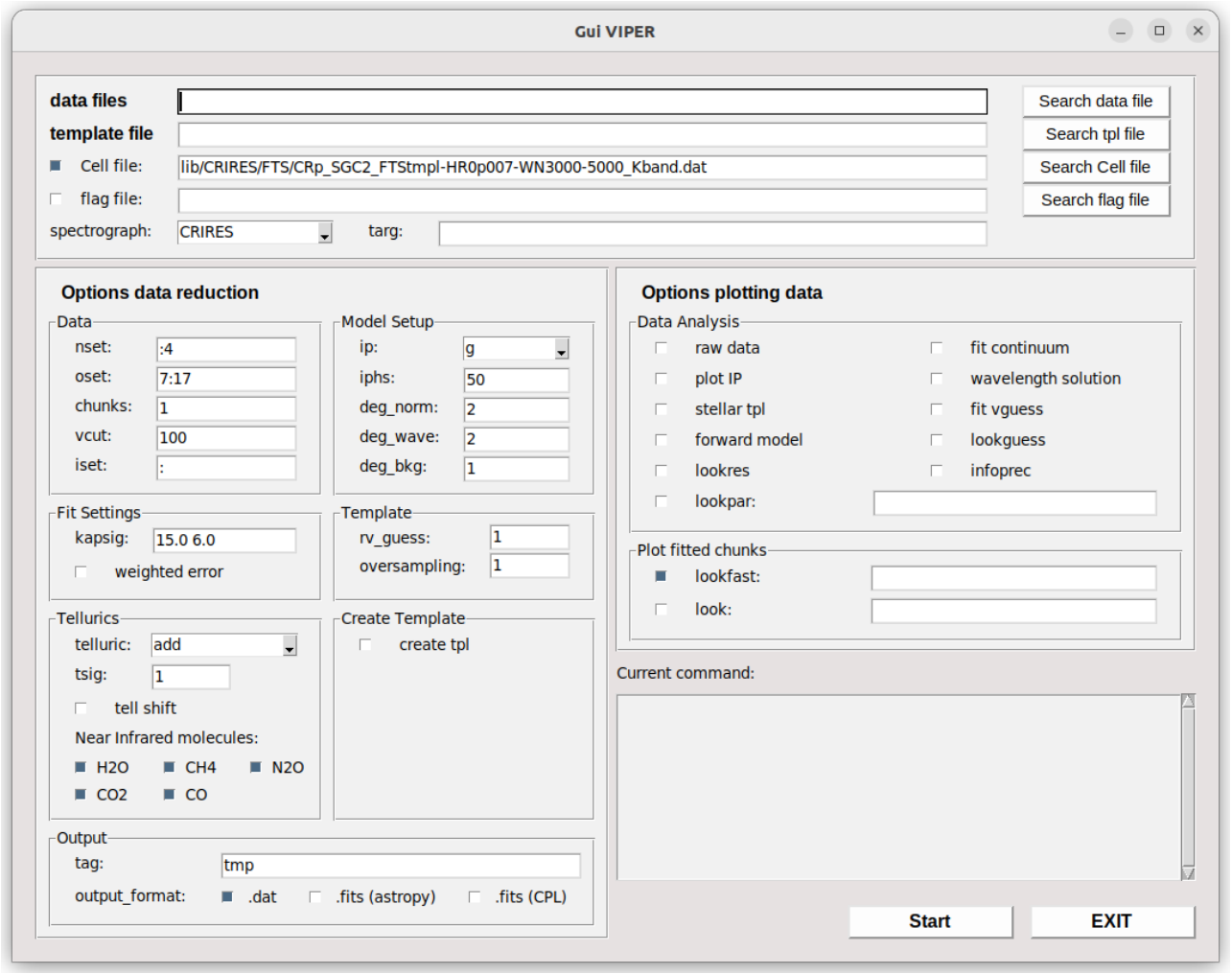

Figure 7.11: The *viper* GUI using `GUI_viper config_viper.ini CRIRES`.

To start *viper* with the selected parameters press the `start` button. While it runs, the GUI is not accessable until the processing is finished. The current command with parameter settings is printed in the GUI. This command can be directly copied to run *viper* from the console or from a bash script.

,

7.6 Post-processing using vpr.py

The *viper* package is coming along with the additional script vpr.py, which allows studing the output. It easily can be used like

```
1> vpr tmp.rvo.dat -avg mean
```

together with additional options, described in detail in Sec. 8.2.3. The outputs of Fig. 7.12 were generated using the following command:

```
1> vpr GJ588_test_jana.rvo.dat -avg mean -cen -ocen
```

whereby the left plot appears first. By pressing enter in the console the right plot appears. If the user is interested in just one of the plot instead of both, the additional option `-plot` can be used. Typing `-plot rv` will result in the left plot, while `-plot rvo` will end up in the right plot.

The option `-ocen` subtract a mean value from each individual order. This is helpful in case of shifts between orders caused by the instrument or calibration errors. Due to this option, the errors of the combined RVs will be reduced, as it is calculated using the standard deviation from all selected orders (Eq. 4). Additionally, the option `-cen` centres the final RVs by subtracting the mean value of the final RVs.

vpr.py offers by now two weighting options called by the parameter `-avg`. The default is `-avg wmean` using a weighted mean based on the errors of the RVs of the single orders. This helps suppressing the influence of problematic orders with large errors. The user should be aware that using `wmean` together with `-ocen` will change the calculated RVs. By using `-avg mean` the RVs from all orders are weighted equally and `-ocen` will have no influence on the final RVs, but just on the errors.

For a faster and more intuitive study of the data, the package comes along with a GUI (GUI_vpr.py). When starting the GUI, Fig. 7.13 will appear to the user, showing all options to plot the RV data. After the corresponding file is selected, the user can choose which orders can be combined, if the RV values should be centered and which kind of weighting is wanted. The buttons `Plot BJD-RV` and `Plot o-rv` allow an easy switch between plotting options as already introduced in Fig. 7.12. The `Save` button will save the revised RV values in an additional file, by printing BJD, RV and e_RV for all observations.

If the user is, beside of the RVs, interested in a detailed study of the quality of the reduction, the plots of the modelled parameters can be helpful (Fig. 7.14). Here, the user can choose between all parameters stored in the `tmp_par.dat` file. These parameters can be plotted vs `BJD`, the observations number `n` or the single `orders`. Depending on the selection, the observations number `n` or the single `orders` are color coded. If the user prefers the console over the GUI, the right plot of Fig. 7.14 can also be produced via:

```
1> vpr tmp -plot par -parcolx BJD -parcoly norm2
```

Additionally, it is possible to plot the residuals of all observations and orders together (Fig. 7.15). To generate the plot as shown in Fig. 7.15 via console, type:

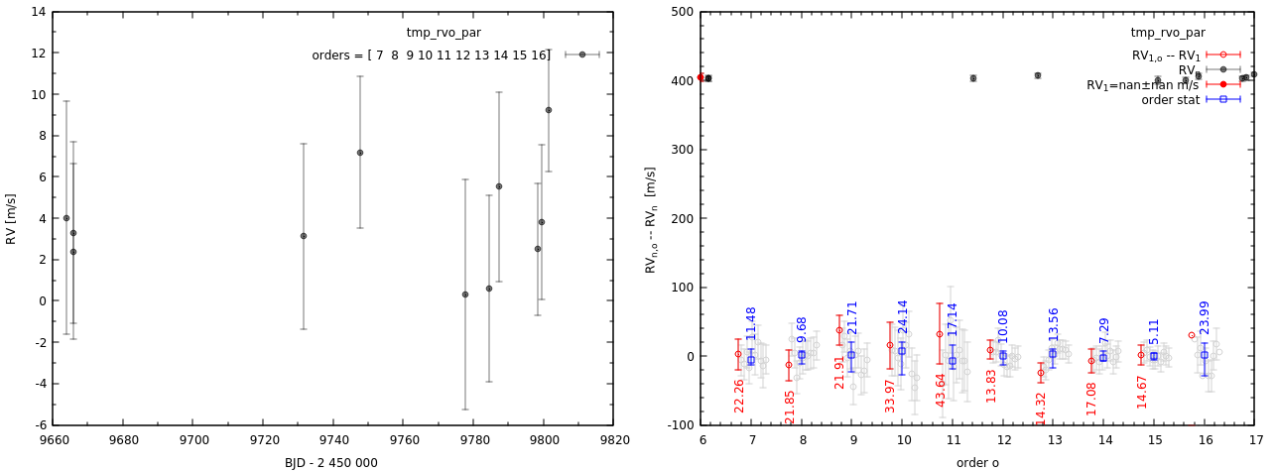

Figure 7.12: Output from vpr.py. Left: Final RVs of all observations after subtracting the mean values from each individual order. Right: RVs of all orders for all observations (bottom) together with the RV rms for each order.

```
1> vpr -res res -nset [`seq -s, 0 27`] -oset [8] -ressep 5000
```

with `-ressep` being the offset between the observations.

Moving the mouse over the parameter-names in the GUI shows the help text for the most important parameters and options.

Please be aware, that the `tmp.rvo.dat` as well as the `tmp.par.dat` and the `tmp_rvo_par.fits` will be overwritten with any new reduction, if the `-tag` option is not used. Likewise, the `res` directory will be cleared and should be stored under another name if the user wants to save the residual values of the last reduction run for later.

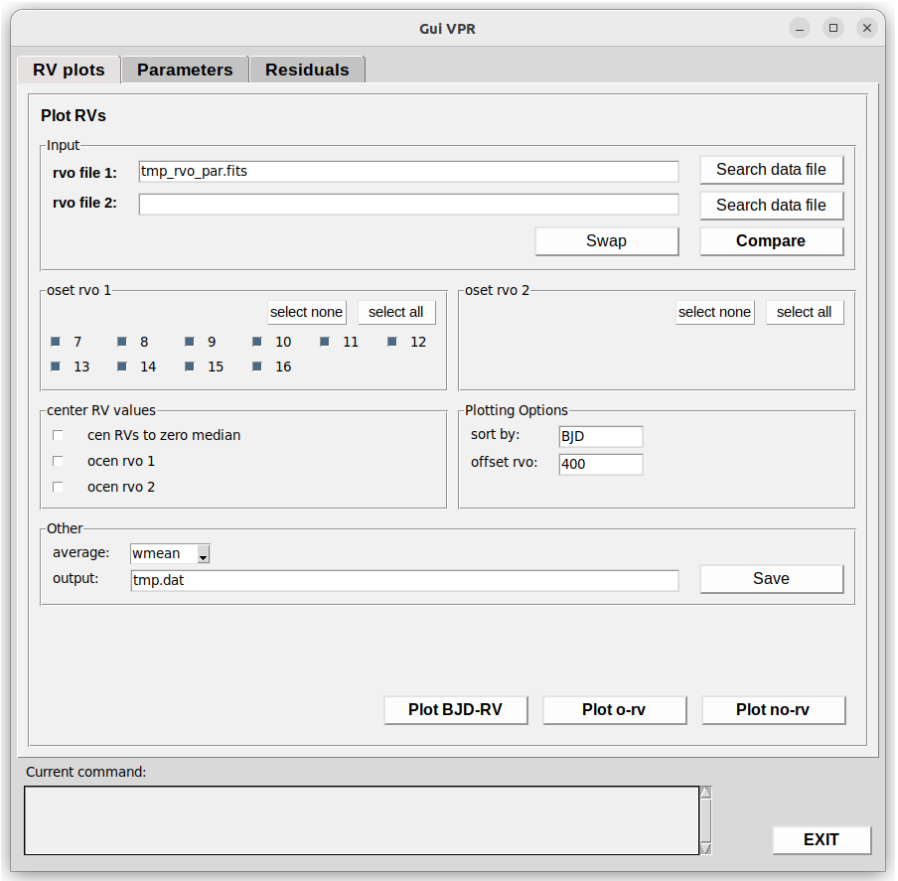

Figure 7.13: Output from vpr.py. Look of GUI showing RV plotting options.

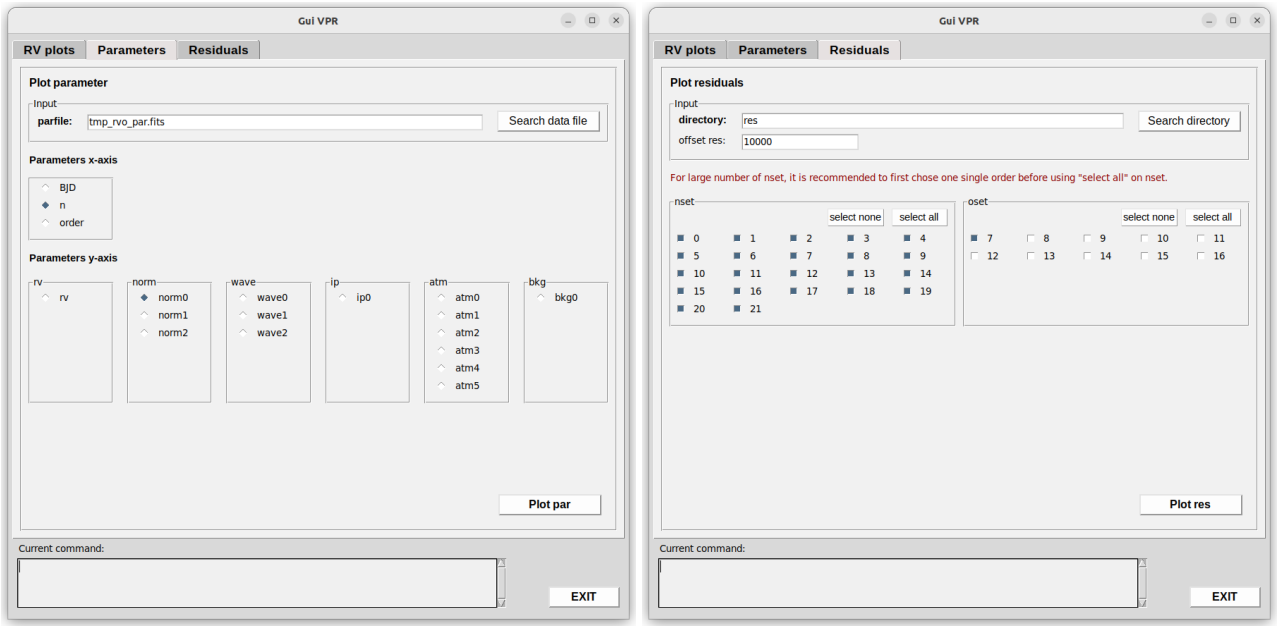

Figure 7.14: Output from vpr.py. Left: Look of GUI showing parameter plotting options. Right: Parameter plot shhowing the norm0 parameter vs BJD with colorplotted spectral orders.

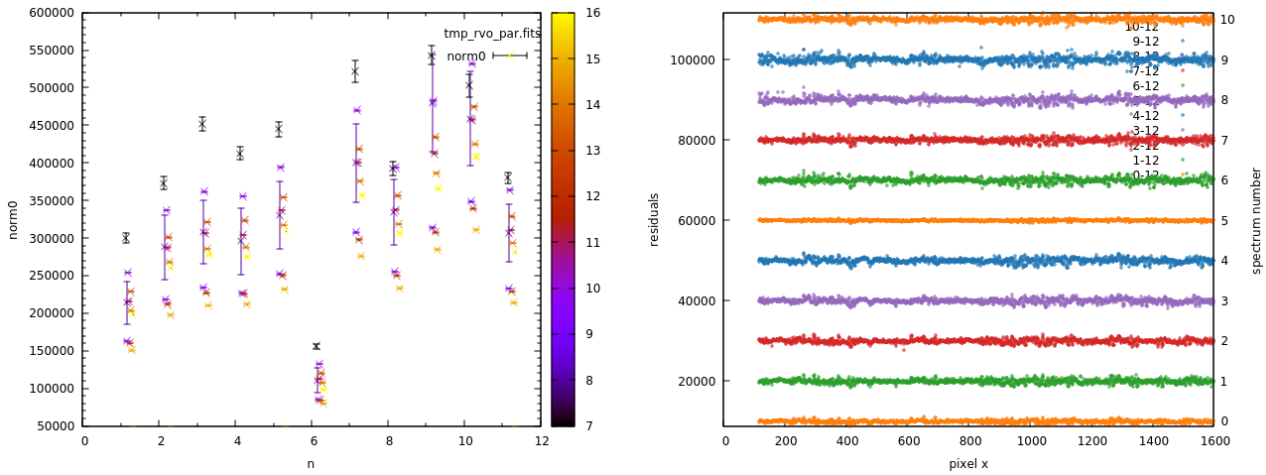

Figure 7.15: Output from vpr.py. Left: Look of GUI showing residual plotting options. Right: Residual plot shhowing the residuals of all observations for a single order.

8 Recipe Reference

viper runs via

```
1> python3 viper.py "<data files>" <template file> <parameters>
```

whereby the use of a template file is optional. Additionally, it is possible to set parameter values in a configure file and run: *viper* runs via

```
1> python3 viper.py "<data files>" <template file> -inst <instrument>
   -config_file <configure file> <configure section>
```

A default config_viper.ini file is stored inside the viper directory. If one want to use a section out of this file, it is enough to type:

```
1> python3 viper.py "<data files>" <template file> -inst <instrument>
   -config_file <configure section>
```

Otherwise the configure file always has to be mentioned. Further parameters set in the console command will given a higher priority than values set in the configure file.

8.1 File description

8.1.1 Static Input Files

All static input files are stored in the `lib/<instrument>` directory. Depeneding on the selected instrument and reduction, the following files are needed:

- FTS cell spectrum
- Synthetic telluric spectra
- blaze, wavesolution (just for CRIRES; maybe removed in the future)

The FTS cell spectrum varies for all instrument, depending on the wavelength range and present molecules in the cell. The telluric spectra are needed if the user wants to make use of the telluric forward modelling. The spectrum of each molecule is stored in a separated file. Furthermore, the telluric spectra of the optical range and NIR wavelength range are stored separately to ensure a faster read in of the corresponding files and therefore a faster data reduction. All telluric spectra, as delivered together with the *viper* source code, were generated using the Molecfit software.

8.1.2 User Input Data Files

- data file(s) (mandatory)
- template file (optional)
- configure file (optional)
- flag file (optional)

The format of the data and template files can vary for the different instruments and are handled by separated instrument files. A recommanded generation of 1D spectra for the CRIRES+ instrument is explained in the appendix C.1.1.

viper comes along with a default configure file `config_viper.ini`, containing default values for a number of instruments. Using the option `-config_file`, the user can select another configure file or section. Own configure files have to follow the structure shown in `config_viper.ini`, meaning at least one section has to be given:

```
1> cat config_viper.ini
   [CRIRES]
   inst = CRIRES
   chunks = 1
   deg_bkg = 1
   deg_norm = 2
   deg_wave = 2
   ip = g
   iphs = 50
   kapsig = 15 6
   oset = 7:17
   oversampling = 1
   rv_guess = 1
   tag = tmp
   telluric = add
   tsig = 1
   vcut = 100
   wgt = 0

   [CRIRES_tpl1]
   inst = CRIRES
   chunks = 1
   createtpl = 1
   ...
```

Hereby, the name of the section is given in the brackets [].

A flag file can be generated by the user using a single text file with the following structure:

```
1> cat flag_file.dat
   # start and end values for masked regions
```

```
# values in pixel if order is given
# values in wavelength [A] if no order
order start end
7 0 150
8 0 100
8 1948 2048
9 1648 2048
- 24623.5 24631.5
```

Hereby `order` is the spectral order as defined in *viper*. `start` and `end` are the start and end pixel from the range which should be masked. It is possible to select more than one region for one spectral order. Further, it is possible to select a region giving the wavelengths in Angstroem and setting the order to - (last line of exampe above).

8.1.3 Output Files

The following products will be generated by *viper* (depending on selected options):

- #_tpl.fits
- #.rvo.dat and #.par.dat or #_rvo_par.fits
- #.dat or #.fits (with vpr.py)
- folder res/#.dat

The #_tpl.fits is produced when choosing the create_tpl option. The format varies per instrument, as always the first input data file (observation without cell) is used as a reference for the creation of the file. The data structure stays the same as well as part of the FITS header.

The estimated RV values are stored in the #.rvo.dat file, with the result of each observation stored in a separated row. An example of running *viper* on two orders of two observations would look like:

```
1> cat tmp.rvo.dat
   BJD RV e_RV BERV rv7 e_rv7 rv8 e_rv8 filename
   2459663.910252932 -46.87194685263491 35.243705692549966 23.313759384762296
   -11.628241160084949 27.117512938308185 -82.11565254518487 22.02830277456099
   220325_1.fits
   2459665.888054387 -32.01028601640189 12.476537861916508 22.81213747615653
   -19.533748154485387 28.64347380157898 -44.486823878318404 24.82341432350439
   220327_1.fits
```

The first line gives the information about the values printed in the corresponding columns. Hereby `rv7` and `rv8` are the calculated RV values for the selected orders (7 and 8) together with the errors `e_rv7` and `e_rv8`. `RV` and `e_RV` are the mean value of the RVs and the corresponding error.

Be aware that in case of no use of a stellar reference template, the RV value will be fixed and not optimized during the modeling process and therefore all RV values in the rvo.dat file will be the same value, equal to the value of `rv_guess`.

The model parameters of the reduction are stored in the #.par.dat file. Again, the first line shows all parameters that were fitted and that are stored in the file. Depending on the selection of parameters, this can vary for each reduction. In contrast to the #.rvo.dat file, each of the following line contains the values for one order of one observation.

```
1> cat tmp.par.dat
   BJD n order chunk rv e_rv norm0 e_norm0 norm1 e_norm1 norm2 e_norm2 wave0
   e_wave0 wave1 e_wave1 wave2 e_wave2 ip0 e_ip0 atm0 e_atm0 atm1 e_atm1 atm2
   e_atm2 atm3 e_atm3 atm4 e_atm4 bkg0 e_bkg0 prms
    2459999.8959765853 1 14 0 1026.265681759598 13.423263056261177 57914.52104453276
   178.778419021483 -0.3418925011783212 0.032286312575589456 7.832952872580637e-05
   5.82570000834357e-05 23750.79815742572 0.000787445741570096 0.0710375788449673
   8.834452166127363e-07 -8.277408057085049e-07 1.8093354621102156e-09
   0.445534845191916 0.021081196461948806 3.673673135684066 0.024621255389495323
   1.119193016459838 0.00641328349127483 nan 0 nan 0 0.6320017020808397
   0.12369588754594299 0.04581836041566867 0.0033098651612129926 9.57209315236739
   ...
```

Instead of storing the estimated values in two separated .dat files, the user as well can make use the option `-output_format cpl` or `-output_format fits` to store all results together in a combined _rvo_par.fits file. The fits file follows the structures of the .dat files and uses the same column names. The rvo data being stored in the first extension of the fits file, the par data in the second extension. The post-processing with vpr.py is able to deal with all three input files, depending on the users preferences. Using the option `-save <file name>` in vpr.py produce another file, which just contains the date BJD, the combined RVs and the errors in m/s:

```
1> cat tmp.dat
   BJD RV e_RV
   2.459375668709358666e+06 1.10513582716394330 4e+03 4.576801916679109183e+00
   2.459376653419110924e+06 1.10413567257362706 2e+03 2.899356724051631584e+00
   2.459377471892267931e+06 1.10819264116116914 9e+03 3.817307235702360746e+00
   2.459377477730993647e+06 1.11034608404877076 3e+03 2.789530407263197453e+00
   2.459377607600891497e+06 1.11059146224762798 7e+03 2.965336960746724149e+00
```

It is possible to wether save it as a simple text file, using the file ending .dat, or as FITS file using the file ending .fits.

In the folder res/ the residuals for each datafile and order of the last run is stored:

```
1> cat res/000_007.dat
   150.0 32.54428425330843
   151.0 -34.977063371530676
   152.0 -50.23396281518944
   153.0 -204.92010341935384
   154.0 -134.22747191853705
   155.0 16.436064190893376
   156.0 -102.32743112541357
```

Here the first seven data points of the seventh order from the first data file are printed, with the first column corresponding to the pixel and the second to the residual value (difference between observation and model). As described in Sect. 7.6 the script vpr.py allows an easy analysis of these data.

8.2 The Parameters

8.2.1 Parameters for *viper*

The following parameters are used for the modelling process and have an influence on the final results.

Parameter	Description
`tplname`	Filename of template. Value = input_tpl.fits If no template is chosen, the template will set to 1.
`inst`	Select instrument. Type = String; Default = TLS Value = CES Value = CRIRES (CRIRES+) Value = KECK Value = GAINO Value = McDonald Value = OES (Ondrejov) Value = TLS Value = UVES
`fts`	Filename of cell FTS. e.g. Value = CRp_SGC2_FTStmpl-HR0p007-WN3000-5000_Kband.dat e.g. Value = None, if no cell file is available
`ip`	IP model used for convolution. Type = String; Default = g Value = g (Gaussian) Value = ag (asymmetric Gaussian) Value = agr (asymmetric Gaussian with restricted skewness) Value = sg (screwed Gaussian) Value = mg (multi Gaussian) Value = lor (Lorentzian) Value = voigt (Pseudo-Voigt) Value = bnd (band matrix)
`chunks`	Number of chunks within one order. Type = Integer; Default = 1 e.g. Value = 2
`config_file`	Using a selected config file and section with given parameter values. Important: A file name AND a section name is needed. Type = String; Default = config_viper.ini

continued on next page

continued from previous page

Parameter	Description
	e.g. Value = config_viper.ini CRIRES e.g. Value = CRIRES_tpl1 , then section CRIRES_tpl1 will be used from default config file, which is stored in the viper directory.
createtpl	Creation of telluric-free stellar template. Default = None If chosen, telluric lines will be removed from selected observations and orders, before generating a combined template.
deg_norm	Polynomial degree for flux normalization. Type = Integer; Default = 3 e.g. Value = 2
deg_norm_rat	Rational polynomial degree of denominator for flux normalisation. Type = Integer e.g. Value = 2
deg_wave	Polynomial degree for wavelength scale. Type = Integer; Default = 3 e.g. Value = 2
deg_bkg	Number of additional parameters (background). Type = Integer; Default = 1 e.g. Value = 2
flagfile	Using a selected flag file to remove noisy regions. Type = String; Default = None e.g. Value = lib/CRIRES/flag_file.dat
iphs	Half size of the IP. Type = Integer; Default = 50 e.g. Value = 70 Use an even number. Make sure your IP is width enough.
ipB	Factor of IP width varation. Type = Float
iset	Borders of used pixel range. Type = Integer e.g. Value = 300
kapsig	Kappa sigma clipping value to detect outliers. Type = Float; Default = 0 e.g. Value = 3.5 4.5, then use 3.5 for first and 4.5 for second clipping e.g. Value = 0 4.5, then no first clipping and 4.5 for second clipping e.g. Value = 4.5 0, then no second clipping and 4.5 for first clipping e.g. Value = 3.5, then use 3.5 for both First clipping will be performed before the modelling process, second clipping afterwards, followed by another round of modelling the data.
kapsig_ctpl	Kappa sigma values for the clipping of outliers in template creation. Type = Float; Default = 0.6

continued on next page

continued from previous page

Parameter	Description
	e.g. Value = 0.6, flag points from spectra which differ by 0.6 from mean spectrum.
molec	Molecular specifies; all: Automatic selection of all present molecules. Type = String; Default = all e.g. Value = H2O, then use model of water tellurics e.g. Value = H2O CH4, then use model of water and methan tellurics e.g. Value = all, then use models of all molecules present Use together with -telluric add.
nset	Index for used spectrum files. Type = Number of Integers; Default = 1 e.g. if nset = :2, then use the first two files e.g. if nset = 3:8, then uses file 3 to 8
nocell	No use of cell FTS for the modelling. Default = None
oset	Index for used orders. Type = Number of Integers; Default = 20 e.g. if oset = 13, then use just order 13 e.g. if oset = 13:16, then use just order 13, 14 and 15 e.g. if oset = 11,14,15 , then use just order 11, 14 and 15
output_format	Format of output files for rvo and par data (dat, fits, cpl). Type = String; Default = dat e.g. Value = dat, generates a .rvo.dat and a .par.dat file. e.g. Value = fits, generates a combined rvo_par.fits file. e.g. Value = cpl, generates a combined rvo_par.fits file using PyCPL (ESO standard).
oversampling	Value for oversampling the template data. Type = Integer; Default = 1 e.g. Value = 3
rv_guess	RV start guess in km/s. Type = Float; Default = 1 e.g. Value = 15.6 An offset of a few km/s is fine.
tag	Output tag for filename. Type = String; Default = tmp e.g. Value = cr2res_GJ588
targ	Target name requested in SIMBAD. Type = String; e.g. Value = GJ588 Name conversion as given in SIMBAD Database. If chosen RA,DEC will be taken from the database instead from the file header. Recommended to obtain an accurate barycentric correction.
telluric	Dealing with telluric lines.

continued on next page

continued from previous page

Parameter	Description
	Type = String; Default = None Value = mask, mask telluric lines Value = sig, down weight telluric lines; use tsig parameter for weighting Value = add, use telluric forward modeling Value = add2, telluric forward modelling with combined coeff for non-water molecules
`tellshift`	Allow variable position shift of tellurics. Default = None If chosen, the position of the telluric lines will be optimized during the modelling process. Otherwise the position is fixed. Just one parameter for all selected molecules.
`tsig`	(Relative) sigma value for weighting for tellurics. Type = Float; Default = 1 e.g. Value = 0.01
`vcut`	Trim the observation to a range valid for the model [km/s]. Type = Float; Default = 100 e.g. Value = 300
`wgt`	Weighted least square fit (employ data error). Default = None

8.2.2 Additional Plotting Parameters for *viper*

Additional plotting options to study and analyse the quality of the data and reduction. All a switched off by default.

Parameter	Description
`demo`	Demo plots. Value = 1, then plot raw data Value = 2, then plot the IP (start guess) Value = 4, then plot the stellar template Value = 8, then plot a simple call to the forward model Value = 16, then plot a wrapper to fit the continuum Value = 32, then plot a wrapper to fit the wavelength solution Value = 64, then plot the fit using start parameters e.g. Value = 7, then plot first three options from above (1+2+4)
`infoprec`	Prints and plots information about precision estimates for the star and the iodine.
`look`	See final fit of chunk and pause. Type = Slice; Default = None e.g. Value = 7:10, then show plots for orders 7,8,9 e.g. Value = 8,12,13, then show plots for orders 8,12,13

continued on next page

continued from previous page

Parameter	Description
	If option is used without additional value, all orders are plotted.
`lookctpl`	Show created template from several observations. Type = Slice; Default = None e.g. Value = 7:10, then show plots for orders 7,8,9 e.g. Value = 8,12,13, then show plots for orders 8,12,13 If option is used without additional value, all orders are plotted.
`lookfast`	See final fit of chunk without pause. Type = Slice; Default = None e.g. Value = 7:10, then show plots for orders 7,8,9 e.g. Value = 8,12,13, then show plots for orders 8,12,13 If option is used without additional value, all orders are plotted.
`lookguess`	Show initial model. e.g. Value = 7:10, then show plots for orders 7,8,9 e.g. Value = 8,12,13, then show plots for orders 8,12,13 If option is used without additional value, all orders are plotted.
`lookpar`	See parameter of chunk. e.g. Value = 7:10, then show plots for orders 7,8,9 e.g. Value = 8,12,13, then show plots for orders 8,12,13 If option is used without additional value, all orders are plotted.
`lookres`	Analyse the residuals. e.g. Value = 7:10, then show plots for orders 7,8,9 e.g. Value = 8,12,13, then show plots for orders 8,12,13 If option is used without additional value, all orders are plotted.

8.2.3 Parameters for vpr.py

Parameters for the addional script vpr.py, which is described in Sec. 7.6.

Parameter	Description
`avg`	Average typ (mean, weighted mean). Type = String; Default = wmean Value = mean, then use mean of selected orders Value = wmean, then use weighted mean of selected orders (by errors)
`cen`	Center RVs to zero median. Default = None
`cmp`	Compare two time series. Type = String; Default = None e.g. Value = tmp2.rvo.dat
`cmpocen`	Center orders (subtract order offset) of comparison. Default = None

continued on next page

continued from previous page

Parameter	Description
cmposet	Index for order subset of comparison. Type = String; Default = None e.g. Value = 7:10, then calculate RV mean for orders 7,8,9 e.g. Value = 8,12,13, then calculate RV mean for orders 8,12,13
ocen	Center orders (subtract order offset). Default = None
offset	RV plotting offset for rvo plot. Just for option -plot nrvo. Type = Float; Default = 400 e.g. Value = 100
oset	Index for order subset. Just for option -res. Type = String; Default = None e.g. Value = 7:10, then calculate RV mean for orders 7,8,9 e.g. Value = 8,12,13, then calculate RV mean for orders 8,12,13
parcolx	Column name for plot par x-axis. Just for option -plot par. Type = String e.g. Value = BJD
parcoly	Column name for plot par y-axis. Just for option -plot par. Type = String e.g. Value = norm0
plot	List of plot tasks. Type = String; Default = ['rv', 'rvo'] e.g. Value = rv, plots BJD vs RV e.g. Value = rvo, plots uncombined rv values for all spectral orders e.g. Value = nrvo, plots uncombined rv values for all spectral orders e.g. Value = par, plots the modelled parameters
save	Filename to save altered RVs. Type = String; Default = tmp.dat e.g. Value = tmp_new.dat
res	Plot residuals stacked (folder name) Type = String; Default = res e.g. Value = res_test
ressep	Separation between traces (tip: 5 rms) Type = Float; Default = 1. e.g. Value = 5.5

A Troubleshooting

A.1 Often-appearing problems and possible solutions

Problem with read-in of data

Make sure the right instrument is chosen.

Problem with read-in of template data

Make sure the right data format is used. Be aware that telluric-free stellar templates generated with *viper* should end on `_tpl.fits` to guarantee a successful read in.

Problem in fitting procedure

- Is a stellar template given?
- Does the stellar template fits the stellar observation? Make sure the same star is used for the template and the observations for which RVs should be obtained.
- Is the observation done with or without cell lines? Use option `-nocell` if observation was performed without any cell.
- Is the telluric forward modelling switched on? Use option `-telluric add` or `-telluric add2` if the observation is performed in the NIR and/or a telluric-free stellar template is used.
- What is the quality of the data? Check if something went wrong during the pre-processing of the data or if the SNR of the data is poor.

B Known Issues

As *viper* is still under development, there are open problems which still have to be solved. Most of them are related to the telluric modelling, which is not trivial.

B.1 Incorrect telluric modelling

As the telluric modelling and optimization in *viper* is complex, it can happen, that wrong parameters for the telluric models are obtained. This happens mainly during the telluric correction, as described in Sect. 7.1.2. In this case, it can help to use `-telluric add2` instead of `-telluric add`. By this, all non-water molecules will have the same coefficient instead of independed ones, which can reduce problems occuring by modelling microtellurics. An example of this can be seen in Fig. B.1. Running the telluric correction with `-telluric add -tsig 1` leads to similar results than shown in the right plot. Nevertheless, for most sources and observations `-telluric add` and a `-tsig` value of 10 is giving best results.

This problem is under investigation by trying to improve the telluric modelling.

B.2 Optimization failed

Related to the above problems, it is possible that *viper* is not able to find the best parameters, which causes a failure of the optimization. While testing, these problems just appeared occasionally during the telluric correction as needed for the template creation (Sect. 7.1.2). So it should be mentioned that this problem hardly appears when running the improved template correction (Sect. 7.1.3), which makes use of a reference spectrum.

To solve this problem and guarantee the optimization is successful on most parts of the spectrum, it can help to update some used parameters. In most cases, it helps to try different values for `-tsig`, varying between

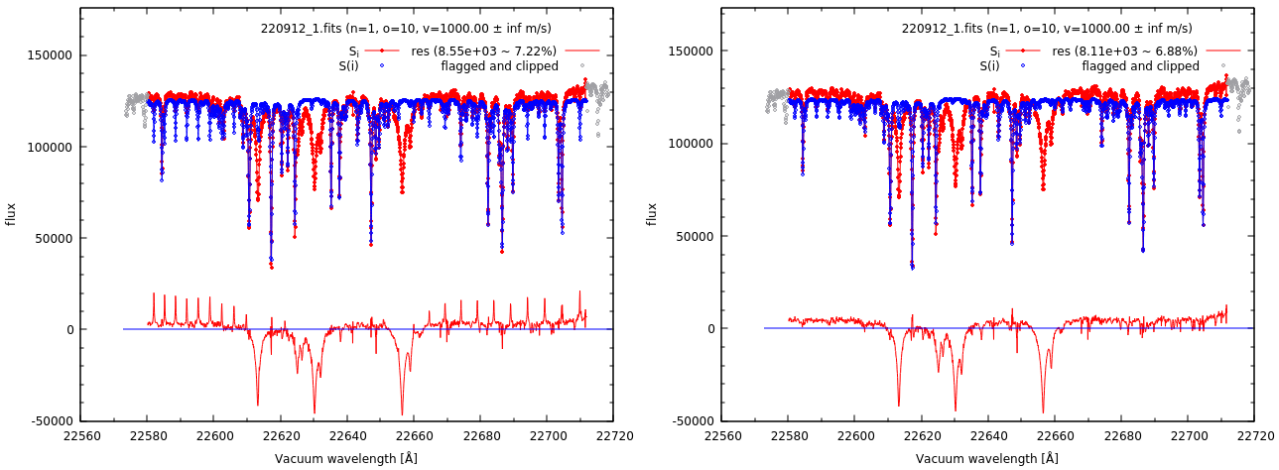

Figure B.1: Telluric correction with *viper* for GJ784 observed with CRIRES+. Left: Using `-telluric add -tsig 10` leaving strong artefacts. Right: Using `-telluric add2 -tsig 10` leading to a better telluric correction.

1 and 1000. Also, it can help to use a smaller value for `-vcut`, f.e. 0 or 10, to use a larger part of the order and therefore more available lines. In the worst case, the user can run the simple template creation first on a few spectra, for which the correction works fine and afterwards use the so created template as an input for the improved template correction (Sect. 7.1.3).

In case of an unsuccessful modelling, *viper* arises a warning without stopping the reduction. Furthermore, the template creation (Sect. 7.1.2) and the combination of all spectra to a final template, is still possible if some orders or observations fail. In this case, all successful orders will be combined.

This problem is under investigation by trying to improve the telluric modelling and making the optimization process more stable.

B.3 Problems in the IP modelling

It appears from time to time that *viper* is not able to model the IP of the instrument correctly. This results in a value close to 0 for the obtained IP parameter and a RV error of inf.

The problem mainly appears for CRIRES+ observation for orders with strong (saturated) telluric lines or orders with a low cell line density.

This problem is under investigation by trying to improve the IP modelling and making the optimization process more stable.

B.4 Available bands with CRIRES+

At the current state, *viper* has mainly been used and tested on CRIRES+ data in K-band, as by now all RV science and commissioning observations have been performed in this band. Further, most tests have been done and parameters optimized for setting K2192 and K2148 due to data availability. Neverthless, it is possible to reduce CRIRES+ data from other settings as well. If a band without existing gas cell is used, `-FTS None` has to be added to reduction commands. Tests are still ongoing.

C Additional remarks

C.1 CRIRES$^+$

C.1.1 Creation of 1D spectra for *viper* reduction

The CRIRES$^+$ raw data are available via the ESO User Portal:

http://archive.eso.org/wdb/forms/cas/eso_archive_main.html

For the reduction of the data the ESO DRS pipeline is suggested. The source code as well as the manual can be downloaded from the ESO pipeline software web page:

http://www.eso.org/sci/software/pipelines/cr2res/cr2res-pipe-recipes.html

During the data selection of a source, the user should take care of using the same number of data files for each nodding position (A and B). In case of the test data, as presented in the tutorial section, the nodding pattern is AAABBB. This means six files have to be downloaded from the archive and will be afterwards combined to a final 1D spectra using the nooding recipe of the DRS pipeline.

The presented data were reduced using the following command:

```
1> esorex --output-dir=withSGC cr2res_obs_nodding --extract_swath_width=800
-extract_oversample=12 withSGC.sof
```

with an example withSGC.sof file looking like:

```
1> cat withSGC.sof
calibs/cr2res_util_normflat_Open_master_flat.fits CAL_FLAT_MASTER
calibs/cr2res_cal_dark_bpm.fits CAL_DARK_BPM
calibs/cr2res_util_calib_calibrated_collapsed_extr1D_tw.fits UTIL_WAVE_TW
archive/CRIRE.2021-06-09T23:27:23.603.fits OBS_NODDING_OTHER
archive/CRIRE.2021-06-09T23:28:28.296.fits OBS_NODDING_OTHER
archive/CRIRE.2021-06-09T23:29:33.310.fits OBS_NODDING_OTHER
archive/CRIRE.2021-06-09T23:30:53.517.fits OBS_NODDING_OTHER
archive/CRIRE.2021-06-09T23:31:58.012.fits OBS_NODDING_OTHER
archive/CRIRE.2021-06-09T23:33:02.737.fits OBS_NODDING_OTHER
```

After a successful reduction following data products are generated:

```
1> ls withSGC
cr2res_obs_nodding_combinedA.fits
cr2res_obs_nodding_combinedB.fits
cr2res_obs_nodding_extractedA.fits
cr2res_obs_nodding_extractedB.fits
cr2res_obs_nodding_extracted_combined.fits
cr2res_obs_nodding_modelA.fits
cr2res_obs_nodding_modelB.fits
cr2res_obs_nodding_slitfuncA.fits
cr2res_obs_nodding_slitfuncB.fits
```

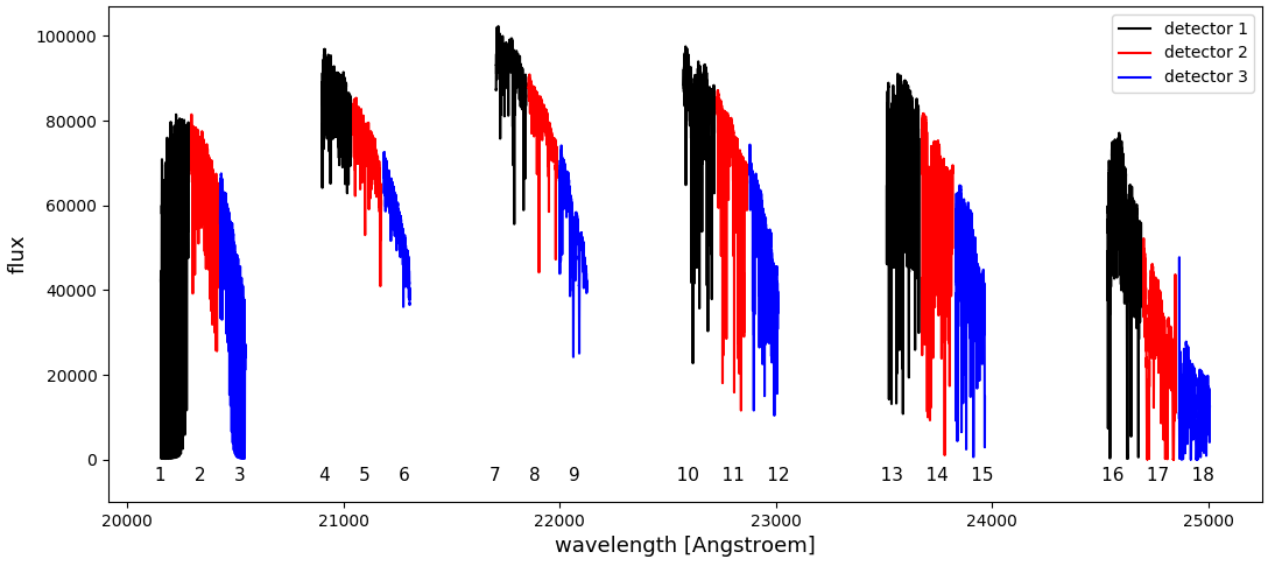

Figure C.1: Definition of orders in *viper* for CRIRES$^+$ data.

Viper is able to reduce all `cr2res_obs_nodding_extracted*` files.

Hereby `cr2res_obs_nodding_extracted_combined.fits` represents the combined spectrum of all nodding observations, whereas `cr2res_obs_nodding_extractedA/B.fits` are the spectra for the individual nodding positions. As during the commissioning of *viper* mainly the combined spectra where used, it cannot be guaranteed that the reductions of the individual nodding spectra with *viper* is always successful.

Follow the DRS Pipeline Manual for further informations.

C.1.2 CRIRES$^+$ order definition in *viper*

During the observations with CRIRES$^+$, each order is distributed over three detectors. To make it easier for the user and due to the fact that each detector of each order is reduced seperately in *viper*, an additional spectral order definition is introduced. The counting goes from 1 to 18, starting from the lowest wavelength range to the highest within one setting, as can be seen for an example observation in Fig. C.1.

D Code design

The design of the *viper* software is simply structured. The script `viper.py` collects the data/spectra from a `inst_#.py` script, whereby every instrument has its own script due to different data structures. Instruments using the IRAF data format furthermore will make use of `readmultispec.py` to get the right format. In each case, the FTS of the cell will read in via `FTS_resample.py`. If the option `-targ <source>` is used, `targ.py` will be able to read in the positions (RA, DEC) of the selected source from the SIMBAD catalogue, as well as the proper motion values. After all data are collected and, `model.py` makes use of the least-square fitting provided by the Python package `scipy` to obtain the best parameters of the optimization process. Afterwards, one can use `vpr.py` to combine estimate RV values from the selected orders or do further process and study of the obtained parameters.

Additional instrument related files are stored in the `lib` directory. Here one finds f.e. the cell FTS files, at least for the instruments for which demo data are available. In the same directory the different model spectra of the atmosphere are stored, as well as a mask file for masking telluric lines.

Besides, the modules `gplot.py` and `pause.py` are used for the plotting done with *viper*, being based on the plotting software `gnuplot`.

Furthermore, *viper* comes along with the two user interfaces `GUI_viper.py` and `GUI_vpr.py` for an easier data reduction.

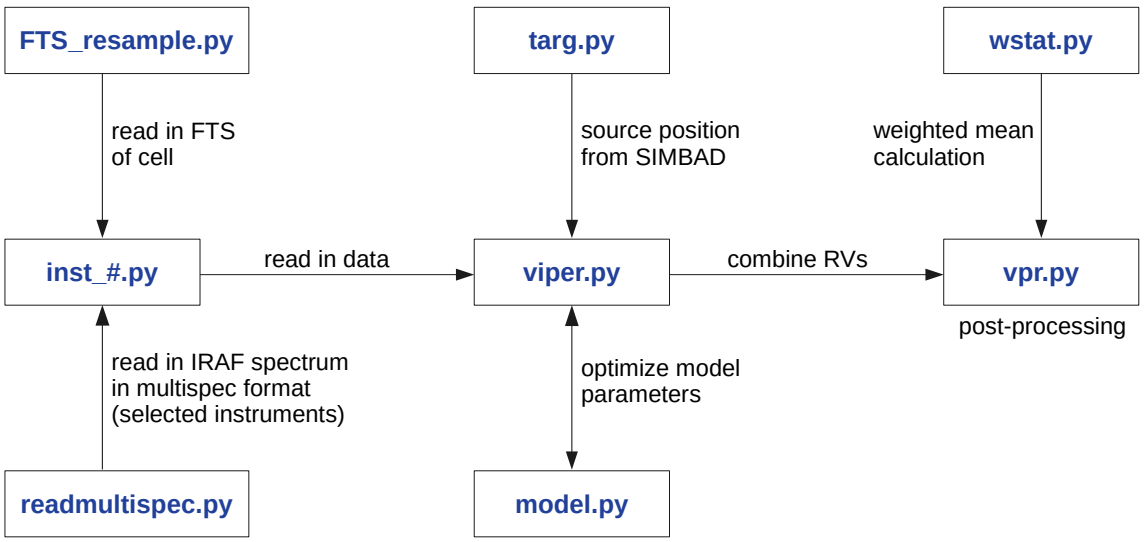

Figure D.1: Overview of the *viper* code design.