VPFIT 11.1

Bob Carswell, John Webb

August 1, 2018

Contents

1 Introduction

The VPFIT program enables you to fit multiple Voigt profiles (convolved with the instrument profiles) to spectroscopic data. If you don't want to do this, read no further. If you want to display data, play around with data and models, try "χ^2-by-eye" fits, display the result of a proper fit, do pretty plots, etc., you might prefer a program written by Joe Liske called vpguess (see https://www.hs.uni-hamburg.de/jliske/vpguess/). The vpguess program can also be used to provide initial estimates for the VPFIT program. You can also provide initial guesses and play around with the data using the RDGEN program which is included in the VPFIT tarfile.

This VPFIT description is fairly stable but some bits are evolving. Where possible we try to be backwards-compatible, but don't guarantee it! There is some information at http://www.ast.cam.ac.uk/∼rfc/vpfit.html, but if you are reading this you've probably been there already.

1.1 General terms of use

Contact: rfc(at)ast.cam.ac.uk

1.2 Reference for VPFIT

If you use the program a suitable reference is to the Astrophysics Source Code Library entry.

Carswell R. F., Webb J. K., 2014, VPFIT, Astrophysics Source Code Library, record ascl:1408.015

2 Acknowledgements

The development of VPFIT would not have been possible without major contributions from Andrew Cooke, Mike Irwin, Julian King, Joe Liske and Michael Murphy. We are very grateful for all their work. We also thank the many others too numerous to mention have also helped with suggestions, comments, questions, queries about possible bugs etc.

3 Obtaining the program

3.1 Download source

Inevitably, the VPFIT program evolves, and you should make sure the description you are using apply to the version you have. This description applies to the version which is available as vpfit11.1 as linked from http://www.ast.cam.ac.uk/∼rfc. Look there first, though some versions may be available from the anonymous ftp space pub/rfc on ftp.ast.cam.ac.uk as vpfit*nnn*.tar.gz, where *nnn* is the version number as given on the first page of this document. If it is not there, then (unusually) the documentation is ahead of the adequately working version of the program, or this is an old document and you should be looking at a more recent one. For those who still have older versions of the program an earlier description may be more appropriate. Hopefully things should generally be upward compatible through versions, so anything which works on an older one should work with a later version, and give close to the same answers. The tarfile also contains the RDGEN program for general spectral plotting, modifying, visualizing and interfacing with VPFIT. It is just a collection of routines which have been found to be useful. If you don't want it ignore it, otherwise try looking at the rudimentary documentation.

3.2 Compiling and linking

VPFIT is written in FORTRAN, so you will need an appropriate compiler. You will also need the CFITSIO library (heasarc.gsfc.nasa.gov/fitsio/fitsio.html) to allow VPFIT to read the data and, if you want to see the data, the PGPLOT graphics library (http://www.astro.caltech.edu/∼tjp/pgplot/). You need to install these before attempting to compile VPFIT.

After you gunzip and extract the VPFIT tarfile, you will need to edit the VPFIT makefile to reflect where the PGPLOT and CFITSIO libraries are held. Since different incantations are used in different places to access these the makefile shows a number of versions - the most extensively tested is the Mac one, but the linux version should also be OK. Others may be there for historical reasons.

Before compiling any of the code it is worth trying `source vppreset.cmd`, since this saves you having to set environment variables. It is important that this be done from whatever directory the source files are in - all it does is edit one line in one routine (vp_srcpath.f) to reflect the directory which contains the source files, and the ancillary files which come with the programs. You could equally well do this yourself, just editing the `' '` to become the VPFIT source directory - still in quotes, so it becomes e.g. `'/home/userx/programs/vpfit'`. If that fails, or you don't like what it gives you, then you can still use the environment variables (see Sections 6.1 and 7) for any or all of the files, since they take precedence.

At least on the machines I use 'make vpflx' makes the Linux version, and 'make vpmac' one for a Mac. You will almost certainly need to change the various system libraries in the makefile to reflect what is normally used for a Fortran compilation at your site. Apart from CFITSIO and PGPLOT, no other external libraries are needed. For RDGEN 'make rdglx' and 'make rdmac' give the corresponding versions.

Makefile will yield the executable code (after you get it right). At this stage it is worth trying to run it just to see if there are any problems, like the instant response: 'vpfit: Not enough memory'. If this happens, then increase your swap space. Or you could edit the 'vp_sizes.f' file to have smaller array sizes - in the default version they are large enough to handle most things I am aware of. In particular the data arrays are 208000 pixels long, which hopefully is enough. If you do change 'vp_sizes.f' then you should 'touch *.f' before running the makefile, to ensure that all routines are recompiled.

4 Spectral data file format

VPFIT data may be either in FITS files or as an ASCII table. The requirements are described below:

4.1 FITS files

FITS files containing spectral data for VPFIT are normally expected to provide the wavelengths, flux values, flux error estimates, and continuum values. Some of these can be omitted if appropriate FITS header information is in the data file, as detailed below. Various header items may be required, depending on the data format and how the wavelength scale has been set up.

The data files which are needed are:

- The spectrum itself may be in a FITS file, in 1-D or echelle format, as <file>.fits or, if you must, as an old-style IRAF file, <file>.imh and .pix.
- The 1-σ error array must be available, and may also be in a FITS file. It is identified by the header entry SIGFILE in the FITS header for the spectrum file, which we here call <NAME>.fits. If that is absent, then the program tries looking for <NAME>.var.fits, <NAME>.sig.fits, <NAME>.err.fits and using whichever it finds first before giving up and prompting.

  If you don't have an error file, then a carriage return (denoted <CR>) response to the prompt here results in $\sigma_i = \sqrt{\max(d_i, 0.2\bar{d})}$, where σ_i is then used as the error estimate at pixel i, d_i is the data for the ith pixel, and $\bar{d}$ is the mean value over all pixels. This is not desperately useful, but is better than nothing in the sense that not all the data will then be ignored. If the error file is absent, you will have to be very lucky (or unlucky) to get χ^2 values which look at all sensible. There is no provision for a quality array, but bad data pixels may be flagged by giving them a negative error (or variance), and the corresponding data is then ignored entirely.
- The continuum estimate specified by CONTFILE in the spectrum FITS header, or, if that is absent, <NAME>.cont.fits. The program prompts if that fails, and a <CR> response to the prompt gives a unit continuum (useful for renormalized data). If you have normalized your data to the continuum, then CONTFILE set to 'unity' (or 'UNITY') stops that prompt, and gives unit continuum (so don't call a data file 'unity', it will never find it).
- Much of the spectral data to be fitted is the sum of several exposures which have been taken at different times, and so possibly with different heliocentric corrections to the wavelength scale. In such cases it is common practice to sum the data onto a common wavelength scale, interpolating the data from each exposure on to a common set of data bins. This inevitably results in some smoothing of the data, and so the root-mean-square noise (RMS) fluctuations are usually smaller than one would expect from the the data error estimates. If the error estimates based on independent data pixels are used the resultant χ^2 is smaller than it should be.

In some sense this does not matter, since if you are aiming for a normalized $\chi^2 \sim 1$ all you are doing is assuming that the S/N is worse than it really is. However it does mean that you are not making the best use of the data you have.

A way of dealing with this problem is to provide a separate file of expected RMS values which has been created from the 1-σ error estimates from the raw data files, using knowledge of the interpolation onto the chosen wavelength scale. If such a file exists, it is specified by the data file header item RMSFILE, or, if that item is absent, <NAME>.rms.fits is used (if present). The RMS values are appropriate for the computation of the effective $\chi^2 = \sum(d_i - f_i)^2/s_i^2$, where f_i is a fit value at i, and s_i the RMS estimate there. Again this is not quite right, since neighbouring points are now correlated. However, if there are a sufficiently large number of pixels in the fit region(s) then it provides a reasonably good approximation.

The 1-σ error file is still needed - it provides the normalization for the parameter error estimates which VPFIT gives. It is also the file which contains the appropriate quantities for inverse variance weighting when combining datasets, and so is the one which is usually provided by data extraction packages.

If there is no RMS file available, then the 1-σ error estimates are used.

Most standard publicly-available packages do not provide this file, but it can be generated. See appendix B for a description of what is needed. The program has been tested in this mode using data from Michael Murphy's UVES_popler (post-pipline echelle reduction) package, which incorporates the methods described in appendix B.

For those who are concerned about correlated pixels and χ^2, rest assured that you are not the only ones. You can test that what has been here is at least reasonable by making an artificial spectrum with independent noise in each of the pixels and then rebinning e.g. with a half-pixel shift, so that each of the new pixels is an average of the old ones, so the apparent noise goes down by $\sqrt{2}$. You will then find, not surprisingly, that the normalized χ^2 for fluctuations about the continuum level with the original error array goes from ~ 1 to ~ 0.5. You can put in absorption lines to fit and find the same result of course. The RDGEN package contains some facilities for generating such spectra if you want to try it.

- The wavelengths are assumed to be in Angstroms. This is not an important restriction but the wavelengths of the transitions in the atomic data file are in Angstroms, so if you want to work in some other units then you will need to make an atomic data table using those units. If you like to work in nanometres then you should be aware that linear and loglinear wavelength coefficients are automatically converted to Angstroms *if* the FITS header item CUNIT1 is set to 'nm' (see section 4.2). If you don't want this to happen set it to something else.

  The wavelengths associated with each pixel can be specified in a variety of ways, either in the FITS header for the data as wavelength coefficients, or as a table in a separate FITS file. In more detail these are:

  - A FITS file of wavelengths, one per spectral data pixel. The header item WAVFILE in the FITS spectrum file gives the filename, or, if that is absent

then <NAME>.wav.fits is used if it is present.

– For a linear wavelength scale the base wavelength and increment can be specified in the header. See section 4.2.

- Spectral resolution information may be specified as a header item (RESVEL [fwhm in km/s]), or in a file in a subdirectory 'database/' of the data directory which is picked up from the header as well (RESFILE filename). Pixel-based instrument profiles are also catered for. The various options are described in section 8.

4.2 FITS data header

The FITS header for the spectral data file will usually contain the standard items, e.g.

```
SIMPLE          T                       file does conform to FITS standard
BITPIX          -32                     number of bits per data pixel
NAXIS           1                       number of data axes
NAXIS1          38528                   length of data axis 1
EXTEND          T                       FITS dataset may contain extensions
```

If the wavelength scale is linear then the coefficients may also be in the header

```
CRPIX1          1                       Reference pixel
CRVAL1          3050.                   Central wavelength of that pixel
CD1_1           0.05                    Wavelength increment per pixel
DC-FLAG         0                       0=linear, 1=log wavelengths
```

or, if you have equal log wavelength intervals, then you might have

```
CRPIX1          1                       Starting pixel (1-indexed)
DC-FLAG         1                       Linear=0, Log-linear=1
CRVAL1          3.51665385452534        log10 Central wavelength first pixel
CD1_1           3.62161104717198E-06    log10 increment per pixel
```

There have been various formats for wavelength information in various IRAF packages over the years. Most are recognized, and if you come across one which is not then you'll soon know, most likely when the program asks for wavelength information.

Other header items might be:

```
AIR
```

.. if this is present then air wavelengths are assumed, and a conversion is performed to vacuum within VPFIT.

```
VHELIO    value    Apply heliocentric correction (value) to wavelengths
```

These are rarely used, and have not been rechecked for a very long time. To be sure of what you are doing it is probably best to apply any such corrections to the data before entering VPFIT.

```
CUNIT1       nm     Wavelength units
```

If this is present and set to `nm`, then the wavelengths are converted to Angstroms **if** they are either linear or loglinear as specified above. Higher order polynomials or wavelength tables are **not** converted.

4.3 ASCII file format

The data may be in an ASCII table consisting of up to five columns of ascending wavelength, spectrum, error estimate, continuum and fluctuation estimate. It is assumed that the wavelengths are vacuum, heliocentric corrected, and any scaling of the error estimate has been applied. You can specify the spectral resolution by inserting a line 'RESVEL nnn' anywhere in the file, where nnn is the FWHM in km/s. The input terminates on detecting the end-of-file. Sample input might look like:

```
# Pks0237-233 subset
RESVEL 6.6
4990.  0.946703 0.03102309
4990.04  0.9681857 0.03154853
4990.08  0.9858243 0.03178264
4990.12  0.9543419 0.03165093
4990.16  0.918863 0.03112572
4990.2  0.9460486 0.03140076
4990.24  0.9262011 0.03116661
4990.28  0.9371514 0.03115548
4990.32  0.9425622 0.03101928
........
```

This example has only three columns. If the continuum column is blank, then the continuum is taken to be unity, and fluctuations = error. The input format is free, with one or more spaces separating the variables. You can also use comma separated variables if you wish.

Comment lines can be present also. Any lines starting with '!' or '#' are ignored, so any comments you want may be included in the file. Blank lines are also ignored.

IMPORTANT: If you are using this option, it is vital to make sure that the wavelength array length is large enough to hold a table of wavelengths. Under normal circumstances this will be true, but if the wavelength space has been restricted you may have to change it. If it has, edit vp_sizes.f before compiling so that the parameter `maxwco` is at least as large as the data size parameter `maxfis`.

FURTHER WARNING: With wavelength tables like this you could in principle store things in any order. Please don't - the program assumes an ordered sequence of increasing wavelengths. The other thing you could do is have large gaps in the data, like finish at 5000 A and restart at 8000 A because you don't see any point in keeping information in between. This is not handled at all either, and it is better to keep the data as separate files. VPFIT is fine with files from all over, but does not handle gaps at all well (since in generating narrow profiles it resamples the pixels to make the Voigt profile, and assumes slow changes in pixel width as you go from one pixel to the next, estimating the width by looking at the wavelength midpoints between pixel centres).

4.4 Special data formats

FITS data files written by Michael Murphy's UVES_popler (post-pipline echelle reduction) package are recognized and read without you having to do anything further. These

files contain the spectrum (normalized to a unit continuum), error estimate and RMS fluctuation estimate.

See http://astronomy.swin.edu.au/∼mmurphy/UVES_popler for more details.

5 Dealing with data problems

5.1 Changing data error estimates

A feature (?) of the current UVES pipeline is that the error estimates in the data at the bottoms of saturated absorption lines may be too low by a factor of up to about 2. Consequently, even if the zero level is correct, or corrected (it can be too high by something up to about 2% of the continuum, but it is wavelength dependent), a satisfactory χ^2 will never be reached when using VPFIT on these features because the fluctuations in the bases of the lines are significantly larger than the error estimates.

VPFIT does not allow changes to the data directly, but RDGEN has an option for modifying the error arrays to avoid this problem, and writing out the results for use with VPFIT. See the RDGEN description for details. The approach is not based on any analysis of the expected errors - all it does is take a function which is roughly 2 where the signal is near zero, and roughly 1 where the data is close to the continuum, and multiply the error arrays by this function.

The cursor mode plotting package in RDGEN also has options for modifying the error estimates (and data, continuum...) on a general or pixel-by-pixel basis. Again the details are given in the RDGEN document.

5.2 Rescaling error estimates

Unless you have access to some package which produces estimates for the fluctuations in the rebinned data as well as estimates for the S/N based on photon counts, you will probably need to adjust the error array associated with the data array if you want to obtain normalized χ^2 values ~ 1 for an appropriate fit. You can do this by (in order of precedence):

- Set up a file with the expected data fluctuations, pixel by pixel, as described in section 4.1. UVES_popler (Section 4.4) does this automatically.
- Set up an ASCII file of wavelengths and data σ/fluctuation values in the subdirectory 'database' relative to the location of the data (but not in the same directory as the data - since then it won't find the file) e.g. 'sigscales.scl'. Insert into the FITS header for the data

  ```
  SCLFILE sigscales.scl
  ```

  and VPFIT will pick up the values and adjust the data σ array by interpolating on these factors.

  If you have ASCII data then you can put the same line anywhere in an ASCII data file and get the same result. If you put it in twice (or more) by mistake then

it will use the last file named in that way. The scale file can be in either the data directory or the database/ directory.

The format is wavelength, scalefactor, anything you like (since only the first two numbers are read) on each line, with fields separated by one or more spaces. Leading spaces are ignored. So the contents might look like

```
3300.0 1.35
   3600.00     1.3
 3888.0 1.25
 4500.10    1.3   0.996812    1.66707E-02    2.167191E-02
6800.0  1.25
```

VPFIT then wavelength interpolates these values and divides them into the data 1-σ estimates to generate an array of expected fluctuations, pixel by pixel, and uses these to generate the χ^2 for the difference between the data and the Voigt profile fit. For wavelengths outside the range in the file, the nearest scale value is used.

If the FITS header contains the SCLFILE field, and the file itself is not present, then a single scalefactor is prompted for and used for the whole wavelength range.

If a data fluctuations file (datafilename).rms.fits exists, or a file of a different name is indicated by RMSFILE in the FITS data header, then the SCLFILE pointer is ignored and the fluctuations file is used.

- In the VPFSETUP file (see section 7) the line

  ```
  sigscale 1.3
  ```

  will result in a single scalefactor (1.3) being used to generate estimated fluctuations from the σ values by *dividing* by 1.3 for the whole wavelength range. This applies ONLY if neither of the options described above are used.

- You can apply a global correction to the error array in RDGEN using

  ```
  em 1.25
  ```

  which results in the whole error array being *multiplied* by 1.25. You will then need to write the results out as a .fits or ascii file for use by VPFIT. This last one is useful particularly for datasets where the error estimates provided by the extraction package are wrong, though you do need to know what they should be!

5.3 Bad pixels

These are flagged by negative or zero values of either the data error or, if present, the fluctuation estimates (i.e. expected RMS values). You will need to edit the σ-values if there are bad pixels. All pixels flagged in this way are ignored. This can be done using RDGEN.

6 Getting started

6.1 Before starting

Before starting VPFIT needs to know where the atomic data is held. This is done via an environment variable ATOMDIR, which gives the path and filename to the relevant file (see Section 14) e.g. by

```
setenv ATOMDIR '/home/userid/vpfit11.1/atom.dat'
```

or wherever you've put it, and whatever you've called it. You should do this before starting VPFIT if you are not using the default option.

You might like to change the plot colours so that data, error and continuum may be distinguished as well. This can be done at display time, but if you

```
setenv VPFPLOTS '/home/userid/vpfit11.1/vp_splot.dat'
```

before starting, then for every plot this file is read and used to reset colours. The options for the `pl` command in RDGEN are supported (see RDGEN description), though since VPFIT holds only the subsets of the data it needs for fitting not the range options are not generally useful as presets. It uses PGPLOT colour indices, so (against a black background) for white data, a green continuum, red error, light blue tick marks and the data plotted as histograms rather than curves it could contain

```
at co co 3
at er co 2
at ti co 5
at da type hist
```

You can also reset e.g. the continuum style to dot-dash lines by `at co st 3` there if you want to. With these you should be able to get a reasonably clear display of the fit to the data.

There are various other parameters to VPFIT which can be changed by using a setup file associated with the environment variable VPFSETUP. These are probably best left alone if you are an absolute novice in the use of VPFIT, but you will probably want to change some things later. They are described in Section 7.

6.2 Interactive mode

There is an interactive mode for setting up initial guesses, which can (and probably should) be bypassed if you have selected fit regions and some rough guesses from some other program. This can be done much more flexibly using e.g. RDGEN, which comes with the VPFIT tarfile, or by J. Liske's VPGUESS (https://www.hs.uni-hamburg.de/jliske/vpguess/). While either of these options does involve learning how to use another program, the gain is that setting up the fit regions and initial guesses is much much less primitive than the interactive mode (which has not changed, or been seriously tested, in decades!) described below. A description of RDGEN is provided as a separate document.

6.2.1 Single region

After starting VPFIT and its general preamble you will see a list of options, followed by a > where the program awaits a response. As anywhere in the program, defaults are given in square brackets - these are the values taken if a carriage return is used.

```
  options:   <CR> for previous value
  I - interactive setup and fit
  F - run from an input file
  D - display profiles from input file
  ? for help
  option (key) (key)...
> i

 Column density (n), logN (l) & scalefactor [l, 1.0]
>
  Filename for data, order number [required, 1]?
> sample.txt [for the demo file, or e.g. sum.ec 26 if sum.ec is a multispec file]
> Column density (n), logN (l) or emission (e), scalefactor [l, 1.0]
>

...
```

The `i` is for entry into interactive setup mode, a carriage return at the second > means it will work in log column densities, and the filename for the data can be anything. If you want to try it, use `sample.txt`. An ASCII data file which is provided in gzipped form with this description (in .../docs), so you can try a few things for yourself if you want to. Note that you are unlikely to get exactly the same results since you will almost certainly choose slightly different wavelength region limits. You are also are unlikely to provide the same initial guesses, so the convergence history will also be a bit different. This spectrum is read in, and then you have to isolate fitting regions and provide starting estimates:

```
 Plot? [y]
>
 plot parameter? (type he for options list)
>

PGPLOT device? [/xwindow]
>
 Expand plot if needed:
 Cursor ("e" or left button at edges,
  "q" or right button when OK)
 Cursor ("e" edges, "q" OK)
```

In interactive mode it would be bizarre indeed to answer 'no' to the first question, but in multiple cases it sometimes saves replotting.

The first 1000 or so pixels are now plotted in a separate window, so you should see something like that shown in Fig 1.

Control is now transferred to the plot window. There are more commands than the short list given for isolating the part(s) of the spectrum you are interested in - just type '?' to get a list. They are:

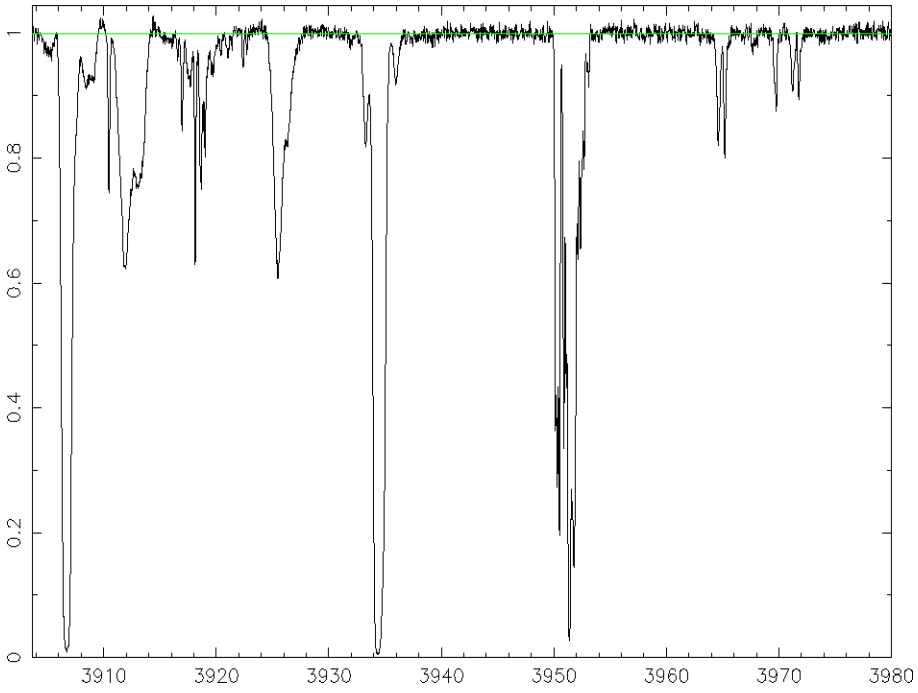

Figure 1: A sample spectrum using VPFIT interactive mode. The green line is the continuum level.

```
Left mouse button, or "e", expands plot
Center button, or "r" replots
Right button, or "q" to exit
"." shift range up
"," shift range down
"a" plot whole array
"d" demagnify by factor 2
"y" max y from cursor
"Q" abandon this region
.. any other lower case letter for command line prompt
```

You can now isolate the feature of interest. For example, the redshifted Lyα at 3926 A could be isolated by clicking the left mouse button with the cursor at 3923 A, and then again at 3929 A. The y-position does not matter (provided of course that the cursor is in the data plot window!). The dialogue is then

```
Cursor ("e" edges, "q" OK)
 Channel 590
right edge..
 Channel 775
Cursor ("e" edges, "q" OK)
```

When you are happy that the region displayed contains the region you want to fit over, click on the right mouse button (or type 'q'), and then you get

```
Mark region in which data is to be fitted:
left limit (space, or left button, when ready)
```

```
 right limit
 Region limits:  3923.500A (channel   605) -   3928.508A (   757)
  Channels  605 757
  Vacuum wavelength for start of chunk is   3923.50933
  In the graphics window
  Line   1 : ion, lamda0? or <CR> to end list
```

This gives you the fitting region limits, and asks what lines you want to put there. You can enter parameters in the order listed by hand if you wish, but any information which is missing may be entered by using the cursor. So, still with the cursor in the data window, enter e.g.

```
HI 1215
```

and the response is

```
 Wavelength used:  1215.6701
 set cursor x- wavelength, y- base of feature
```

and click the left button on the mouse. The x-position of the cursor is taken as the observed wavelength for the line center so the (initial) redshift is determined, and the y-position is provides a line center optical depth estimate when its value is compared with the continuum value for that point. The cursor then moves up to an estimated half optical depth level, with

```
  .. and now half width at half (optical) depth
```

and you should then go to either one side of the line or the other and click again to give it an estimate for the line width. You then get log column density, Doppler parameter and redshift estimates, and a chance to put in more lines, so the text screen output with line IDs etc from the plot window might look like:

```
  Estimated vac wavelength & z are  3925.49121  2.22907606
  13.3632225  36.1756785  2.22907606
  Line  2 :ion, lamda0, N,b,z? <CR> to end
  .. in the graphics window
HI
 Wavelength used:  1215.6701
 set cursor x- wavelength, y- base of feature
  .. and now half width at half (optical) depth
  Estimated vac wavelength & z are  3926.31006  2.22974963
  12.8688193  28.9873807  2.22974963
  Line  3 :ion, lamda0, N,b,z? <CR> to end
  .. in the graphics window
HI
 Wavelength used:  1215.6701
 set cursor x- wavelength, y- base of feature
  .. and now half width at half (optical) depth
  Estimated vac wavelength & z are  3926.97363  2.23029548
  12.1335024  26.5826797  2.23029548
  Line  4 :ion, lamda0, N,b,z? <CR> to end
  .. in the graphics window
```

```
  Include other data? y, n (def),
   (or sf for more from the same file)
>
```

The carriage return terminates the process, and returns control to the text window.

The wavelengths need not be entered accurately - VPFIT chooses the nearest wavelength in the atomic data table (see section 14). If you neglect to give a wavelength at all, it chooses the first one in the list for the given ion. Also, elements with single letter names, like hydrogen, need not have a space between the element and the ionization - 'HI' and 'H I' are equally acceptable.

There is another minor short-cut. Entering '*' instead of 'HI' just repeats the last one entered, so the second and third ion/wavelength entries could equally well have been just `*`.

Note also that if you try to type the line parameters or ID's in the VPFIT command window, as opposed to the PGPLOT window, the program will ignore them as line ID's and try to save them as commands. This is not useful, and can be confusing. It was done this way so that you don't have to oscillate between windows, as well as keyboard/mouse, when entering in a number of lines. What you type is reflected in the command window so you can see what you are typing. The backspace or del keys can be used to correct mistakes, though if you do use these then the ID line being typed goes on to a new line in the command window .. so the last line you see there is the one which is being entered as the line ID, preceded by a history of your mistakes which are ignored! However, this 'feature' can be a real pain if you have a system (like a Mac) where the window focus following the mouse pointer is not an option.

Carrying on with this example, if you don't want to include other regions in the fit just type 'n' to the 'Include other data?' question, and the program will now iterate to determine the best fit Voigt profile parameters for the three components if you tell it to:

```
> n

  Continue (c), fit (f), or stop (s)? [f]
> f
  no. of ions for fitting is  3

   ion          N            z             b         bturb   temp

 iteration   :   0 ( 1 )
 chi-squared :          2.358 (       339.5488,   144 )

   H I      13.36322     2.2290761     36.1757        0.00   0.00E+00  0 !    1
   H I      12.86882     2.2297496     28.9874        0.00   0.00E+00  0 !    2
   H I      12.13350     2.2302955     26.5827        0.00   0.00E+00  0 !    3


 iteration   :   1 ( 1 )
 chi-squared :          1.067 (       153.6667,   144 )

   H I      13.35395     2.2290686     35.9745        0.00   0.00E+00  0 !    1
   H I      12.82776     2.2297712     30.4385        0.00   0.00E+00  0 !    2
```

```
   H I      12.10340     2.2303048     28.7790        0.00   0.00E+00  0 !    3

.......

 iteration   :   8 ( 1 )
 chi-squared :          0.971 (       139.7969,   144 )

   H I      13.36824     2.2290807     37.1597        0.00   0.00E+00  0 !    1
   H I      12.68014     2.2297723     25.4604        0.00   0.00E+00  0 !    2
   H I      12.41261     2.2301257     41.7090        0.00   0.00E+00  0 !    3

 Parameter errors:
   H I       0.00619     0.0000046      0.5395        0.00   0.00E+00    !    1
   H I       0.29051     0.0000221      4.5144        0.00   0.00E+00    !    2
   H I       0.56100     0.0004240     24.5479        0.00   0.00E+00    !    3

 statistics for whole fit:
  Runs test  K-S test Chi-squared  Chans ndf   APr   Xp(.68) Xp(.95) Xp(.99)
   0.00044   0.04271    139.80     153  144   0.583  151.57  172.72  185.69

 Statistics for each region :
    Start      End    Chi-squared  Chans df?
   3923.18   3928.81    139.80     153  144   0.583 < Prob < 0.770 g= 0.583   1
   3923.18   maxdev       1.4340              0.033   0.956   1.358   1.627

  Plot? y,n, or c=change device [y]
>
  Line, system number: (<CR> or * for internal guesses)
>
 plot parameter? (type he for options list)
>

  Plot? y,n, or c=change device [y]
>
```

At this stage you should have a plot showing you what the fit looks like, as in Fig. 2. The data is black, the fit green, and the positions of the fitted lines are shown in light blue. The numbering gives the system number, as in the output above, on the top, and the line ID number, in the ordering given in the atomic data file, on the row below. Lyα is #1 in the list which was used here.

At this stage responding 'n' to the 'Plot?' question, then 'n' to 'Fit more lines?' closes the program, and the displayed fit will disappear.

The results are written to several places, which are by default:

- fort.13 contains the first guesses so you can run the fit again from file, if you wish to.
- fort.18 contains the full fit history, with the parameter values at all the iteration steps as seen on the screen.
- fort.26 contains a summary of the final results of the fit. This is in a different

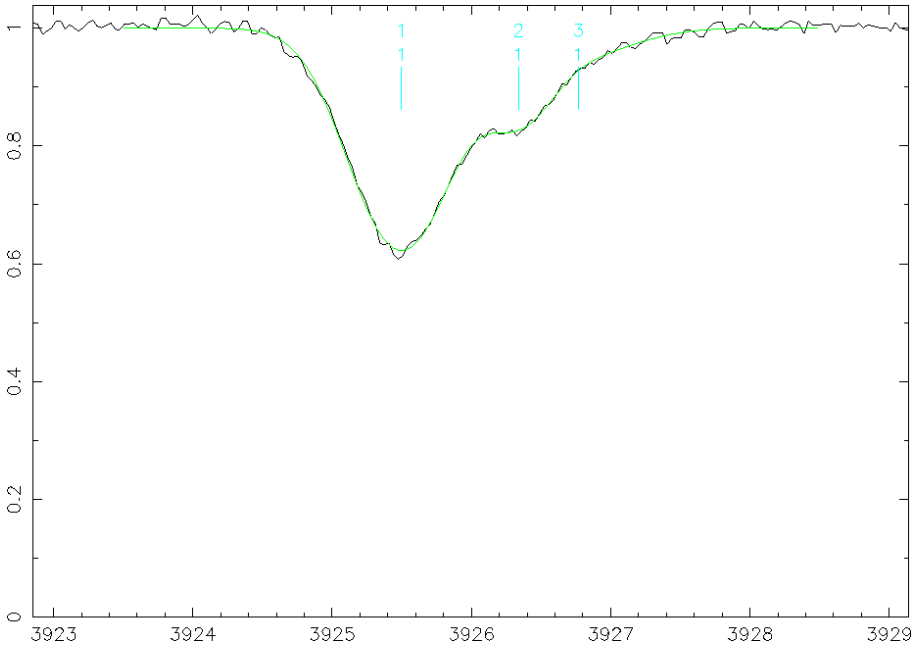

Figure 2: A three component Lyα fit to a region of the sample spectrum, obtained using the methods described in the text.

format than fort.13, but is read equally well by VPFIT if you wish to use it to restart the program. For some of us it has become the default format.

Both fort.13 & fort.26 are ASCII files which can be edited before restarting. Note that if the program is re-run then fort.18 & fort.26 are over-written, and if re-run in interactive mode then fort.13 is over-written as well. If this is a nuisance then you can use the setup file (see section 7) to change where the summary output is written.

Why were three lines chosen for this example? Because two were tried and found not to be enough. The fit statistics given in the screen (and fort.18) output are the things to look at, and the main one is the χ^2 value and the number of degrees of freedom (ndf). A probability that the data matches the fit given these values is computed (APr), and χ^2 values for three probabilities is listed. For two systems, $\chi^2 = 207.82$ for ndf= 148, which gives APr= 0.001.

There is a region-by region statistics breakdown, so you can see the bad regions, but here the probabilities are given as possible ranges because the degrees of freedom per region is not calculable. There is a runs test and K-S test probability given as well, but these are more prone to suffer from smoothing effects in rebinned data so are usually ignored.

6.2.2 Multiple regions

If there are several transitions for an ion available then the system parameters are better constrained by using all of the available lines. You can do this by specifying multiple wavelength regions. For example, in the sample data you might note that the close pairs

of lines at 3965 A and 3971.5 A look like a two component system seen in CIV 1548 & 1550. For these the process is similar to above. Mark off the boundaries of the lines at 3965 A, with now two `CIV 1548` instead of three `HI 1215`. Having done this you should then respond 's' to the 'Include other data' prompt, and you'll get something like

```
  Include other data? y, n (def),
   (or sf for more from the same file)
> s
 Adding the previous lines to the display
     1 C IV    12.807  1.560792   14.31
     2 C IV    12.622  1.561157    8.38
 Plot? [y]
>
 plot parameter? (type he for options list)
>
```

The whole available spectrum is then plotted, and you need to navigate to the CIV 1550 lines which are now helpfully marked. Expand the region, and mark off the edges of this new CIV 1550 wavelength zone in the same way as before (don't include the line near 3970 A - it is Galactic CaII). Do NOT put in CIV 1550 as lines though. VPFIT knows they are there, and putting them in a second time will only cause confusion or line rejection.
So you might for this region have the sequence

```
 Region limits:  3970.541A (channel  2034) -   3972.366A (  2089)
  Channels  2034 2089
  Vacuum wavelength for start of chunk is   3970.54363
  Line  3 :ion, lamda0, N,b,z? <CR> to end
  .. in the graphics window

  Include other data? y, n (def),
   (or sf for more from the same file)
>

  no. of ions for fitting is  2



   ion          N            z             b         bturb   temp

 iteration   :   0 ( 1 )
 chi-squared :          1.886 (       252.6578,   134 )

   C IV     12.77243     1.5607976     13.5745        0.00   0.00E+00  0 !    1
   C IV     12.62013     1.5611554      8.5531        0.00   0.00E+00  0 !    2

.....

 iteration   :   3 ( 1 )
 chi-squared :          1.263 (       169.2334,   134 )
```

```
   C IV     12.80507     1.5607889     14.4107        0.00   0.00E+00  0 !    1
   C IV     12.67798     1.5611530      8.7871        0.00   0.00E+00  0 !    2

 Parameter errors:
   C IV      0.01211     0.0000029      0.5163        0.00   0.00E+00    !    1
   C IV      0.01309     0.0000020      0.3863        0.00   0.00E+00    !    2

 statistics for whole fit:
  Runs test  K-S test Chi-squared  Chans ndf   APr   Xp(.68) Xp(.95) Xp(.99)
   0.00000   0.01309    169.23     140  134   0.021  141.29  161.73  174.29

 Statistics for each region :
    Start      End    Chi-squared  Chans df?
   3963.27   3966.51    108.09      79   73   0.005 < Prob < 0.017 g= 0.005   1
   3963.27   maxdev       1.0796             0.194   0.956   1.358   1.627

   3970.18   3972.83     61.15      61   55   0.265 < Prob < 0.471 g= 0.265   2
   3970.18   maxdev       1.2073             0.108   0.956   1.358   1.627

.....
```

You are now asked which region you want to plot, and can have a look at each in turn. This can be extended to multiple regions, or multiple files. It is just a matter of how much patience you have in setting things up.

One thing which is worth stressing again is that it is important NOT to enter each transition of each ion individually. Once an ion is in the system, with a start redshift normally determined from one line, then VPFIT will put in all lines of that ion which fall in any of the wavelength regions specified.

It is not necessary to have the same ion list either. For example, if the second region were extended to include the Galactic CaII 3970 line then the sequence for the second region could be

```
 right limit
 Region limits:  3968.892A (channel  1985) -   3972.598A (  2096)
  Channels  1985 2096
  Vacuum wavelength for start of chunk is   3968.92154
  Line  3 :ion, lamda0, N,b,z? <CR> to end
  .. in the graphics window
CaII 3970
 Wavelength used:  3969.5901
 set cursor x- wavelength, y- base of feature
  .. and now half width at half (optical) depth
  Estimated vac wavelength & z are  3969.69849  2.73041612E-05
  11.8807063  12.9534595  2.73041612E-05
  Line  4 :ion, lamda0, N,b,z? <CR> to end
  .. in the graphics window

  Include other data? y, n (def),
   (or sf for more from the same file)
>
```

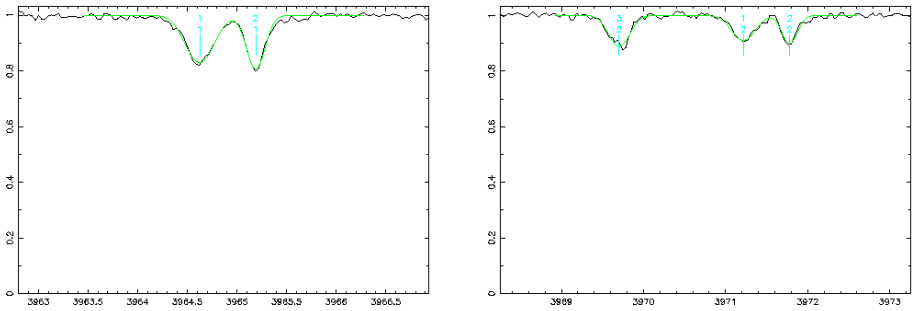

Figure 3: A three component Lyα fit to a region of the sample spectrum, obtained using the methods described in the text. Fits at redshift $z = 1.561$ to CIV 1548 (left) and CIV 1550, with low redshift to Galactic CaII 3970 (right).

```
.......
iteration   :   3 ( 1 )
 chi-squared :          1.250 (       236.2565,   189 )

   C IV     12.80493     1.5607889     14.3879        0.00   0.00E+00  0 !    1
   C IV     12.67798     1.5611531      8.7672        0.00   0.00E+00  0 !    2
   CaII     11.92909     0.0000286     12.5837        0.00   0.00E+00  0 !    3

 Parameter errors:
   C IV      0.01204     0.0000029      0.5169        0.00   0.00E+00    !    1
   C IV      0.01296     0.0000020      0.3870        0.00   0.00E+00    !    2
   CaII      0.02196     0.0000019      0.8401        0.00   0.00E+00    !    3

 statistics for whole fit:
  Runs test  K-S test Chi-squared  Chans ndf   APr   Xp(.68) Xp(.95) Xp(.99)
   0.00000   0.05532    236.26     198  189   0.011  197.74  221.79  236.45

 Statistics for each region :
    Start      End    Chi-squared  Chans df?
   3963.13   3966.61    110.44      86   80   0.014 < Prob < 0.039 g= 0.014   1
   3963.13   maxdev       1.1861              0.120   0.956   1.358   1.627

   3968.59   3972.93    125.81     112  103   0.063 < Prob < 0.176 g= 0.063   2
   3968.59   maxdev       0.8735              0.430   0.956   1.358   1.627

.....
```

The CaII could be better constrained by including CaII 3934, and under normal circumstances one would include that as well. However it is totally swamped by a strong redshifted Lyα transition, so it is not worth including it.

A feature of this mode of operation is that you get best fitting parameters for exactly those components you have put in. No more and usually no less (sometimes lines become refined away to very low column densities, and they are then automatically removed). You can add keywords to the initial response line to tell VPFIT to add lines and try

again if the best-fit χ^2 is unsatisfactory, iterating until either the χ^2 value is OK or it increases again with the addition of more ions. How to do this is described in section 6.5.

6.3 Running from file

While the interactive mode is fine for learning what the program does, it is tedious having to use it every time. If you have first guesses then it is almost invariably more convenient to run directly from a file. The first guesses need not come from interactive runs of the program, but could come from anywhere. Joe Liske's program vpguess (https://www.hs.uni-hamburg.de/jliske/vpguess/) provides a more convenient way of setting up an initial file. Another way is to use the accompanying RDGEN program to set up VPFIT start files - see the RDGEN description for this.

The results are the same as for the interactive mode, but now without the hassle of having to remark the region and enter guesses for something you tried before and somehow things were not quite right. The output file(s) from the program giving the results, which have default names fort.13 and/or fort.26 unless otherwise specified, can be used as input for a subsequent fit, after editing if you wish. This can have the format as for the output file written by the program, but a freer format is accepted. Each field in an input line can have any number of spcaes between it and the next. Thus an acceptable input file for a spectrum held in HE0515rc.fits is:

```
  *
HE0515rc  1 3691.378 3693.014
HE0515rc.fits    1  3697.606  3699.173
  *
 CIV    12.63     1.3848243      6.5544
   C IV     13.08     1.384894      6.67
```

Note that a blank line is a terminator for a dataset.

Each line between the '*' separators give the filename, spectrum number (normally 1 unless it is in a multispectrum format), and the fitting region start and end wavelengths. Where several regions are involved in a simultaneous fit, the filename etc has to be specified for each.

Following the '*' separator below the file and region information comes the initial values for the fit parameters (first guesses). Here ther is one line per system, giving the ion, log column density, redshift and Doppler parameter for each redshift component.

So, in the example here CIV 1548 & 1550 are fitted simultaneously, with two redshift components. CIV 1548 falls in the first region (3691.378 - 3693.014 A) and CIV 1550 in the second (3697.606 - 3699.17 A). Note that you don't need to specify line wavelengths - the program looks to see if any lines from ions in the list at the given initial redshifts fall in the specified wavelength regions and puts them in automatically. The rest wavelengths for each ion come from the atomic data file, see section 14.

This enables you to write an input file by hand if you wish to, rather than the easier method of using the program to generate it. Note that a space between element and ionization is allowed, or may be omitted.

Two input file formats are supported. The one above, which dates from the original version of the program, and another which follows the output summary file format (see section 9.2) so these can be edited and used to restart the program if desired. It will be interpreted correctly unless you have removed the %% markers before the filenames. The version given below is completely equivalent to the one above:

```
%% HE0515rc.fits     1   3691.3777   3693.0135
%% HE0515rc.fits     1   3697.6064   3699.1730
 C IV     1.3848243    0     6.5544      0.64 12.63    0.0  0 !
 C IV     1.384894    0.000004     6.67      0  13.08    0.0  0 !
```

The '%%' lines now given the file and fitting region information, and others the ion, redshift Doppler parameter and column density. Note that the order of variables giving the ions has changed, and there are some spare numbers after the redshift, Doppler parameter and column density. They are there for compatibility with the summary output file, where they are the parameter error estimates. On input here they are ignored, but some quantity must be there (a single 0 is fine). As above, it does not matter whether or not there is a space between the 'C' and the 'IV', and free input format is allowed.
To use either of these files just type 'F' (from file) instead of 'I' (interactive) as the option on startup. So when following the preamble you see, with entries against the `>` symbol:

```
  options:   <CR> for previous value
  I - interactive setup and fit
  F - run from an input file
  D - display profiles from input file
  ? for help
  option (key) (key)...
> f
 Column density (n), logN (l) or emission (e), scalefactor [l, 1.0]
>
 Parameter input file, # entries? [fort.13,1]
>
```

The 'f' (or 'F') in reponse to the first `>` prompt results in the program prompting for a filename. The following prompt asks if you want to work in log or linear column density space [logs are the default, i.e. the variables used if you just hit carriage return], and the third `>` prompt asks for the input filename. If you have called it `fort.13` then a carriage return will do here as well.

6.4 Multiple data files

The format of the input parameter files above suggests that all the data need not be in the same spectral data file. So if the two CIV lines are in separate files HE0515c1.fits and HE0515c2.fits then both will be used, with the wavelength regions specified against the filename.

```
%% HE0515c1.fits     1   3691.3777   3693.0135
```

```
%% HE0515c2.fits     1   3697.6064   3699.1730
 C IV     1.3848243    0     6.5544      0.64 12.63    0.0  0 !
 C IV     1.384894    0.000004     6.67      0  13.08    0.0  0 !
```

Here CIV 1548 is fitted to the data in HE0515c1.fits, and CIV 1550 to the data in HE0515c2.fits, and the program finds the best fit parameters using those single regions from both files.

You can take this further. If you have datasets taken with different resolutions say sp1.fits with instrument FWHM 6 km/s, s2.fits (FWHM 7 km/s) and s3.fits (FWHM=12 km/s) covering both lines (though they need not), then

```
%% s1     1   3691.3777   3693.0135 vfwhm=6.0
%% s1     1   3697.6064   3699.1730 vfwhm=6.0
%% s2     1   3691.3777   3693.0135 vfwhm=7.0
%% s2     1   3697.6064   3699.1730 vfwhm=7.0
%% s3     1   3691.3777   3693.0135 vfwhm=12.0
%% s3     1   3697.6064   3699.1730 vfwhm=12.0
 C IV     1.3848243    0     6.5544      0.64 12.63    0.0  0 !
 C IV     1.384894    0.000004     6.67      0  13.08    0.0  0 !
```

will yield a best fit to all the data together (we've dropped the '.fits' since it is appended automatically if there is no other type specified).

There are dangers in this approach unfortunately, the main one being that the continuum estimates for each of the different data files almost certainly won't agree precisely. This means that as far as the program is concerned the data are not consistent with each other, so you may not find a satisfactory minimum χ^2 solution. You can get around this to some extent by using continuum adjustments (see section 14.2.3) for each region for each file, and using the number before the ! in the ion list to say which region in the list it should be used for (where 0=all). So e.g. could have:

```
%% s1     1   3691.3777   3693.0135 vfwhm=6.0
%% s1     1   3697.6064   3699.1730 vfwhm=6.0
%% s2     1   3691.3777   3693.0135 vfwhm=7.0
%% s2     1   3697.6064   3699.1730 vfwhm=7.0
%% s3     1   3691.3777   3693.0135 vfwhm=12.0
%% s3     1   3697.6064   3699.1730 vfwhm=12.0
 C IV     1.3848243    0     6.5544      0.64 12.63    0.0  0 !
 C IV     1.384894    0.000004     6.67      0  13.08    0.0  0 !
 <>       2.037008  0 0.00 0 1.00 0 1 ! s1, first region
 <>       2.042355  0 0.00 0 1.00 0 2 ! s1, second region
 <>       2.037008  0 0.00 0 1.00 0 3 ! s2, first region
 <>       2.042355  0 0.00 0 1.00 0 4 ! s2, second region
 <>       2.037008  0 0.00 0 1.00 0 5 ! s3, first region
 <>       2.042355  0 0.00 0 1.00 0 6 ! s3, second region
```

You can, of course, miss out the continuum adjustment for e.g. s1 if you believe you have determined the continuum accurately - the others will then adjust for the best match given that continuum. Doing this for lots of regions is (computer) time-consuming and when we tried it ages ago it was not wonderfully satisfactory. So we have chosen normally

to combine the data into a single spectrum and built up a sub-pixel instrument profile as a weighted mean of those which have gone into the single spectrum (see section 8.2). This is what was done for Carswell et al., MNRAS, 422, 1700, 2012, where we did run tests to make sure the results were insensitive to the adopted instrument profile model.

All this is fine if you know the component structure of the complexes you are looking at, and almost invariably you don't. However, there is a facility to add lines automatically until some convergence criterion is satisfied (such as the probability that the fit is acceptable). This is best done from an input file of first guesses. Details are given in the next section (6.5).

6.5 Optional parameters

When starting VPFIT you are given a short list of options

```
  options:   <CR> for previous value
  I - interactive setup and fit
  F - run from an input file
  D - display profiles from input file
  ? for help
  option (key) (key)...
>
```

The response to this has a number of options in addition to the basic few listed. If you type '?' you get a brief list, which looks like

```
  D  Display fit from file
  E  Display and compute errors from file
  F  File start, fit parameters
  I  Interactive start, fit parameters
  G  Provide initial guesses and fit

  Key words checked are:
  add (p) (p)  -  add new lines if needed to
       get an acceptable fit, the values (p)
       are threshold probabilities for CHI^2
       and KS test respectively [0.01,1E-8]
       OR, threshold normalized CHI^2 (if >1)
  Ly-a all added lines are assumed to be Ly-a
  unknown, or ??, all added lines ?? (1215.67)
  zup (p1) add Ly-a if its redshift < p1, nearest otherwise
  ecol (p1) (p2) - remove lines at end if
       log col < (p1) AND log col err > (p2)
  fixb (p) - added lines have fixed b [20]
  diagnostics  -  write diagnostic o/p
  inc  filename - include earlier data
  cum  filename - cumulatively include earlier data
```

These options apply to all regions covered in a fit, and are given on a single line following the basic option.

- **add (p1 (p2))** When the program has converged to a best fit with the number of ions it has, check to see if the fit is acceptable, and if it is not, add a new ion and

start again. This is done iteratively until either an accptable fit is attained or the χ^2 for the fit fails to decrease when lines are added. The acceptance criteria are if the χ^2 probability is greater than **p1** and the K-S test probability is greater than **p2**. If both of these are satisfied then the process of adding components stops. The defaults are **p1**=0.01 and **p2** = 10^{-8} (so the K-S test is normally ignored).

Where there are different ions in a complex the added ion is the same as the one closest to the position of a line which the program determines should be added. Occasionally also lines need to be divided into two, because the interim best fit goes signifcantly below the data. I such cases both new components have the same ions as the original divided line. An exception to this is where the nearest line is a high order Lyman line, where HI at a redshift which assumes Lyα is added instead.

The added ion can also be specified by giving the option `Ly-a` or `??` (or `unknown`).

You might prefer to use e.g. the Akaike criterion to tell the program when to stop adding components. The overall Akaike fit statistic is printed out (labelled AIC) to enable you to do this after the event if you wish, but for individual fits the number components added in tends to be rather more than is required for a satisfactory fit. For this reason we chose not to build it in as the default option. If you want to use it then include a line with just 'Akaike' in the setup file (section 7, and the program will terminate when the AIC variable starts to increase, and write out the previous iteration. Do not use 'nopchan 2' in the setup file, since it interferes with the process - 'nopchan 1' is fine. If you look at the parameter error estimates after the event you will probably see why we believe it is not a good choice. Also, if using this option, please don't start with a good fit - the program looks for a minimum AIC, so if it starts with one you are not likely to get anything convincing .. the best AIC fit may be one with even fewer parameters!

- **Ly-a** All added systems assume that new lines placed in badly fitted parts of the regions are Lyα, independent of the nearest system.

  If you want to restrict the redshift range for which Lyα will be added, you can set an upper limit of e.g $z = 3.63$ by including **zu 3.63**,

- **??** or **unknown** All added systems have the `??` line as their basis, as specified in the atomic data file.

- **ecol p1 p2** End of iteration check of column densities. If an ion has a log column density $<$ `p1` and the error in the column density is $>$ `p2` then, when the program has converged to a satisfactory χ^2 (or given up because it failed to do so), these ions will be removed, worst first, to see if a satisfactory fit can still be obtained. It carries on doing this until the fit becomes unacceptable or there are no further systems satisfying the trial rejection criteria.

- **fixb p1** All added lines have a fixed Doppler parameter, value `p1`. You have to be pretty desperate to want this, but it can be useful if there is a messy blend where the Doppler widths are poorly constrained, provided that you have some reason to believe that the value you have entered is correct, or does not matter.

- **inc filename** Add lines from the ions with parameters in a file to the spectrum to modify the effective continuum. These are taken to have fixed redshifts, Doppler parameters and column densities and are not refitted. The input format for this file is as for the summary file (see section 9.2), and typically could be used to include weak lines where the parameters are known from elsewhere in a region. A better alternative is to fit all regions together, but if there are e.g. many high order Lyman lines from many systems going into a region of interest this may be unnecessarily time-consuming.

- **cum filename** This has the same action as the `inc` option (so don't use both together!) but for each fit the summary ouput is appended to `filename`, and used for subsequent fits if there is a series.

So if you wanted to fit with starting guesses from a file, add components until the χ^2 probability exceeded 0.02, with added components being unknown, and attempt to remove lines for the log column density was < 14.3 and error estimate > 0.3, modifying the continuum with system paramters contained in a file called f26.old, then use

```
f ad 0.02 ?? ec 14.3 0.3 inc f26.old
```

Caution: added 26.01.2015 If you included continuum (or zero level) adjustments in the fits these are written to the accumulated results file as well. In the version of VPFIT before this date these presets were also included in the preset lines if the wavelength of the 'line' corresponding to the adjustment falls within the region you are fitting, and not otherwise. So you might or might not be fitting this region against an adjusted continuum, it would be hard to tell without some extra work. With the new version the continuum and zero level adjustments from the accumulated results file are NOT included in the presets for the new fit, so all fits are against the originally assumed continuum.

6.6 Use for fine structure constant

The input file can have the form

```
   *
J012417-374423    1  6272.250  6275.880  vsig=2.810
J012417-374423    1  6288.800  6290.060  vsig=2.810
J012417-374423    1  5344.700  5346.300  vsig=2.810
J012417-374423    1  5326.200  5327.610  vsig=2.810
J012417-374423    1  5258.180  5259.780  vsig=2.810
  *
   MgII     11.12085     1.2431860AX    6.0400BX  -5.41E-06QA
   MgII     12.46181     1.2432293aa    1.3145ba  -5.41E-06QA
   MgII     13.57020     1.2432912ab    6.0504bb  -5.41E-06qa
   MgII     13.31861     1.2433628ac    6.4206bc  -5.41E-06QA
   MgII     12.71755     1.2434859ad    4.8919bd  -5.41E-06QA
   MgII     12.20853     1.2435642ae    7.7863be  -5.41E-06QA
   MgII     11.58066     1.2437463      4.1546    -5.41E-06QA
   MgII     11.67876     1.2440402      4.9295    -5.41E-06QA
```

```
FeII     12.50575     1.2432293AA    0.8672BA  -5.41E-06QA
FeII     14.18841     1.2432912AB    3.9915BB  -5.41E-06QA
FeII     13.27883     1.2433628AC    4.2357BC  -5.41E-06QA
FeII     12.20504     1.2434859AD    3.2272BD  -5.41E-06QA
FeII     11.55266     1.2435642AE    5.1367BE  -5.41E-06QA
<>        0.99988     3.3971637SZ    0.0000SB   0.00E+00SE
```

where the extra column is an estimate for the change in the fine structure constant $\Delta\alpha/\alpha$ appropriate for the systems it covers.

Note that everything now has a fourth variable attached to it, so it is vital to mark it as fixed if there is no fourth variable (Q or K) associated with it in the atomic data table (see section 14.3). So, in the example above, the 'SE' fixes the value at zero for the continuum adjustment <>.

To use the program in this mode the VPFSETUP file **MUST** contain

```
NOVARS 4
```

(see section 7), and the atomic data table (section 14) **MUST** contain the appropriate q-coefficients for each transition (see section 14.3). For this to work at all, the ions used must be flagged with at one or two character alphabetic identifier, and the first letter **MUST** be 'q' or 'Q' (to tell the program to use the q-values from the atomic data file for all transitions from this ion at this redshift). The scale of the variable $\Delta\alpha/\alpha$ as printed can be changed as well - see section 14.3.

Different tags on the fourth parameter will cause different actions (at some stage). It is hard to imagine anybody wanting more than 26 different fine structure constants, so this should not be a significant restriction. As usual, the lower case one is the reference one, and the upper case flags (of which there may be zero) are tied to that.

If instead you are interested in the electron/proton mass ratio using e.g. H_2, use 'm' or 'M' instead of 'q' or 'Q' as tied tags. If you want to use this for something other than H_2 you'll need to set up you own atomic/molecular data file.

It is anticipated that this mode will be used after a preliminary fit has been obtained with $\Delta\alpha/\alpha = 0$ (with NOVARS=3). I have absolutely no idea what might happen if you attempt to start interactively in this mode, but doubt that it will be good.

You should be able to leave the fourth parameter column blank for an ion which has lines which are not subject to fine structure constant or electron/proton mass ratio shifts, or you don't care about. The column should then become a fixed zero - as in the example above.

6.7 Velocity precision from a portion of spectrum

Murphy, Webb & Flambaum (astro-ph/0611080 v3) have given a prescription for estimating the velocity precision one can obtain by using all the pixels in a spectral region. If NOVARS=4 (or more), and the prescription above is followed, then the $\sigma_{\rm v}$ defined by their equation (4) is computed and written to both the full output file (usually fort.18) and the summary output file (fort.26 or whatever you have changed that to in the VPFSETUP file). The (fort.26, for five regions) results have the form

```
...
%% J012417-374423.fits     1   5258.1763   5259.7773 vsig=2.810
! chunk  1  sigma_v=      0.04504709
! chunk  2  sigma_v=      0.04621412
! chunk  3  sigma_v=      0.03821856
! chunk  4  sigma_v=      0.05051567
! chunk  5  sigma_v=      0.04580045
.....
```

where the sigma_v values are in km s^{-1}. It is then up to the user to associate q-coefficients with each spectral region based on the transition appropriate for that region, and calculate the uncertainty in $\Delta\alpha/\alpha$ using their equation (2). This could in principle be done within the program here, but it isn't (yet).

7 Setup parameter file

VPFIT setup parameters are read from a file which is pointed to by the environment variable VPFSETUP, e.g by

setenv VPFSETUP /home/whoever/wherever/vp_setup.dat,

or if you want to use one in the directory you are working in then simply

setenv VPFSETUP vp_setup.dat.

The second one allows you to switch setup files as you switch working directories without having to change the environment variable.

Some of the possible contents of vp_setup.dat are described below. It gives various optional presets for the running of the program. An example file could be:

```
lastchtied v
! ---- parameter limits ----
bvalmax 50000.0        Maximum b value (can have two - metals, HI)
bvalmin 0.2            Minimum b-value (up to two) (should be << that expected)
! cvalmin 8.0          minimum column density (log)
! cvalmax 30.0         maximum column density (log)
bltdrop 0.101          drop system if b value less than this AND ..
clogltdrop 14.3        if log column density less than this.
bgtdrop 50001.         Drop any system with b .gt. this (> bvalmax, so none).
fcollallzn 4.162e22 1.0E-4   Fik*col for inclusion everywhere, if optical depth in line there > 1.0E-4.
! zerolevels -0.3 0.3  limits to zero level adjustment (default - none)
! ----  miscellaneous things
! chisqthres 0.001 4.0 0.01 0  relative chi^2 decrement for stopping
! setspecchar H        internally fixed parameter flag
! dots 100             Put dots on screen print.
! ----  add/remove line parameters ----
adsplit 2.5            Line splitting bias
maxadrem 75            Maximum number add/remove cycles before stopping
absigp 10.0            line search lower level for splitting
adcontf 2.2 10 5.0 12.5
adzerof 4.5 5.0d-3 5   Add zero level if 5 pixels or more < 0.005*continuum
! colglo 0             column density below which guesses are ignored
! guessline CIV 1548   Line guesses are given feature (default is Ly-a)
! ----  output control
date                   datestamp the '26' summary output
! wr13s f13.w          end summary root filename (fort.13 format)
wr26s f26.w            fort.26 summary root filename
! wr26prob             include chunk probability estimates in summary output
! nopchan 0            output (0=not much, 1=iterations to 6, 2=6+fort.18)
! vform                use variable format O/P depending on error estimate
! wrcitn               current iteration to junk.dat, remove on clean exit
! ----  internal substepping ----
! nsubmin 1            minimum number of subpixels per pixel
! nsubmax 11           maximum number of subpixels per data pixel
! nfwhmp 7             require at least 7 pixels per instrument fwhm
! ----  development tools ----
! pcvals               prompt for controls (il, stepsize etc)
! verbose nn           information printout control
! DEBUG 0
```

Note that comment lines are ones which start with a ! (or #), so any parameter line

may be commented out, as are several in the example above. Some of the options above may not be described below - but hopefully the comments in the setup file are clear enough.

It is important to finish the last line in this file with a line feed, otherwise it won't be read and interpreted properly. Spare blank lines at the end don't matter.

The following sections describe the effects of some of these parameters. The list here is unlikely to be complete though. If you want the full range of choices have a look at the routine which reads the setup file - vp_rdspecial.f. The one line comments against each one above should give a rough idea of what each does.

7.1 Tied parameter letter

```
lastchtied v
```

If the first letter of the one or two letter tied parameter flag is after this in the alphabet (so in this case `w - z`), then don't tie the parameters in the standard way described in Sections 11.1 and 11.2. Instead for Doppler parameters estimate temperatures (see Section 11.3) and for column densities the first in a sequence is the sum of those which follow (Section 11.4).

If the lastchtied = v then it is worth avoiding the use of that letter (or whatever other letter it is set to). There may be a bug in the Doppler parameter check which means that sometimes it may use v as a standard flag, and sometimes a special one. Finding where this might happen has so far eluded us.

The default letter is `z`, so if this parameter is not set there is no possibility of special handling of column densities or Doppler parameters.

7.2 Parameter limits

Limiting values for the fitted parameters may be specified through various keywords in the setup file. The ones which are recognized are:

- Minimum Doppler parameter:

  ```
  bvalmin nnn
  ```

  where nnn is the minimum value of $b = \sqrt{2}\sigma$ (in km s^{-1}) allowed. If the best fit solution for any ion has a lower value of b, it is automatically reset to $b =$ nnn.

  If two quantities are given, as

  ```
  bvalmin n1 n2
  ```

  then the first (n1) is taken as the minimum b for all lines apart from HI lines, and n2 is the minimum for the HI lines only.

- Maximum Doppler parameter:

```
bvalmax nnn
```

where nnn is the maximum value of $b = \sqrt{2}\sigma$ (in km s^{-1}) allowed. If the best fit solution for any ion has a higher value of b, it is automatically reset to $b =$ nnn.

If two quantities are given, as

```
bvalmax n1 n2
```

then the first (n1) is taken as the maximum b for all lines apart from HI lines, and n2 is the maximum for the HI lines only.

- Minimum column density

```
cvalmin n1
```

Here n1 is the minimum column density allowed. If the program tries a value less than this then it is set to `n1`. If `n1`< 25 then the value is taken as the log of the column density in cm^{-2}.

- Maximum column density

```
cvalmax n1
```

Here n1 is the maximum column density allowed. If the program tries a value greater than this then it is set to `n1`. If `n1`< 25 then the value is taken as the log of the column density in cm^{-2}.

- Criteria for dropping systems

```
bltdrop b1          drop system if b value less than this AND ..
clogltdrop c1        if log column density less than this.
```

Systems are dropped, and the search for the best fit continues without them, if the Doppler parameter $b <$`b1` AND the (log) column density is less than `c1`. Both parameters should be specified, but in case you don't the defaults are `b1` $=$ 0.050001 and `c1` $= 14.0$.

Another parameter

```
bgtdrop b2
```

simply ensures that systems with ridiculously large Doppler parameters are also dropped. I doubt that this is often used.

- To include a line ion in all fitting regions use

```
fcollallzn n1
```

VPFIT uses the ion list to look for lines in the specified wavelength regions, and includes them only if they fall within those regions. Lines for which the column density × oscillator strength exceed this (log) value

`n1`

are included in all wavelength regions. It slows the program down a bit, but is occasionally useful if you want to include the damping wings of a Lyα line which has a centroid which is otherwise too far from the region of interest.

- Limits to zero level adjustment (see Section 14.2.4) can be set by

  `zerolevels n1 n2`

  The effect is to constrain the zero level adjustment to be between n1 & n2 × the local continuum. This is normally not needed (and the default is no limits), but sometimes you have might have knowledge of partial coverage and wish to set limits. For normal data `n1` should be negative - otherwise the zero level will be constrained to be above zero!

- Limits to continuum adjustments (Section 14.2.3) are similar:

  `contlevels p1 p2 p3 p4`

  The continuum level factor is constrained to be between p1 & p2 (so p1 would normally be less than 1, and p2 > 1), and the slope between p3 & p4. So p3 would normally be negative, p4 positive. If the last two values are omitted then there are no slope constraints.

7.3 Miscellaneous

7.3.1 Stopping criteria

At each iteration with a given number of systems a check is made on the change in the relative χ^2, and if $\Delta\chi^2/\chi^2 < 0.001$ (default) the iterations terminate. However, if the normalized $\chi^2 > 4$, the stopping criterion is less stringent. The iterations terminate if $\Delta\chi^2/\chi^2 < 0.01$. So if the fit is not very good the program terminates earlier. This saves a bit of time if lines are being added automatically.
These thresholds can be changed in the VPFSETUP setup file, using the line

`chisqthres 1E-5 5.0 0.05`

which changes the stopping $\Delta\chi^2/\chi^2$ to 10^{-5} for a normalized $\chi^2 < 5$, and to 0.05 if for normalized $\chi^2 > 5$.
You can add another parameter to this line in the VPFSETUP file. If you have

`chisqthres 1E-5 5.0 0.05 16`

then, independent of the thresholds, the iterations will stop only if there have been more than 16 of them *and* the $\Delta\chi^2/\chi^2$ criterion is satisfied. This can be useful if you are testing the range of χ^2 for various parameters and want the number of iterations to be controlled.
The VPFSETUP file can also be used to set the maximum number of iterations allowed in attempting to find a minimum χ^2 fit for the current set of parameters (i.e. ions, redshifts, b-values, etc.). Use

```
maxitn 125
```

to set this to be 125 (which is the default). Under normal circumstances the default limit is not reached. Note that if you are adding or removing lines iteratively then the χ^2 minimization count is reset to zero when the number of ions is changed.

7.3.2 Fixed flag

Occasionally added lines within VPFIT have some parameters fixed. For example, the 'redshift' assigned to a continuum adjustment acts only as a marker to tell the program which wavelength region the continuum adjustment (Section 14.2.3) is to be applied to, so it has to be fixed (for a description of fixed parameters see Section 11.1). The line

```
setspecchar H
```

means that `H` is the letter associated with internally fixed parameters. You should avoid using this letter associated with parameter values in your input files unless you know what you are doing.

7.3.3 Progress monitor

For those who have slow machines with long interactive iterations, and like to be reassured that the program is actually doing something, you can arrange for a row of dots to appear, one for every `n1` parameters, by using

```
dots n1
```

Whenever `n1` parameters have been dealt with, a dot disappears from the screen. It is rarely used or wanted, but it is there... Default: no dots.

7.3.4 Internal variable scaling

Some variables are scaled internally so that their values are not orders of magnitude different from the others. The scalefactors are unlikely to need changing, but you can if you want to. This may be done through the setup parameters file.

- `internal lognscale p1` Scale the logN values by multiplying them by a factor `p1` for use internally. Default: `p1=1.0`.

- `internal facnscale p1` Scale the column density values N by a factor `p1` for use internally if fitting linear column densities. Default: `p1`= 10^{-14}, so column densities $\sim 10^{14}$ have values internally ~ 1.
- `internal thunit p1` Temperatures are internally measured in units of `p1` K. Default: `p1`= 1000.

7.3.5 Finite difference step sizes

The parameter derivatives are estimated by finite differences and you might (should?) want to be sure that the results are not strongly dependent on the step sizes assumed. The step sizes for the key variables (redshift, Doppler parameter, column density and, if appropriate, any fourth variable such as $\Delta\alpha/\alpha$ etc.) can be prompted for (see subsection 7.5) or preset in the setup file. If you use the latter option, then the parameter finite difference steps are set independently for each.
Doppler parameter (<value> in km s^{-1}):

```
fdbstep <value>
```

Log column density (<value> in log cm^{-2}):

```
fdclog <value>
```

Linear column density if using that mode (<value> in cm^{-2}):

```
fdclin <value>
```

Redshift (<value> is difference in z):

```
fdzstep <value>
```

.. and for any fourth variable, if used

```
fd4vst <value>
```

The descriptors can be in either upper or lower case, but not mixed case. If any value is ≤ 0 then the internal default is used.

7.4 Adding/removing ions

7.4.1 Adding continuum parameters

If the program is adding lines automatically (see Section 6.5) then it will include continuum parameters to the worst region if the normalized χ^2 value drops below a given threshold (and provided that region does not already contain a continuum adjustment, of course). Other (much rarer) occasions when a continuum adjustment might be appropriate are if there are repeated additions of weak narrow lines because. so, with `adcontf`

`p1 p2 p3 p4` in the setup file the program will put in a continuum adjustment if normalized χ^2 goes below p1 OR the last p2 entries in a row have b <p3 and log column density <p4. So, e.g.

```
adcontf 3.5 10 5.0 12.5
```

will insert a continuum adjustment when the normalized χ^2 drops below 3.5.

7.4.2 Adding zero level parameters

If the program is adding lines automatically (Section 6.5) then it will a include zero adjustment if the normalized χ^2 is below some value (p1) and the current fit for that region is below p2 for p3 channels or more.

```
adzerof 5.0 5.0d-3 5
```

will result in a zero level adjustment being included in the worst region if the normalized $\chi^2 < 5.0$ overall, and for that region the **fit** (not the data) goes below 0.005 for 5 or more pixels.

7.4.3 Guess line

The `guessline` parameter gives the line to use for initial guesses if no lines at all are specified for any wavelength region. This applies only for the mode where you are starting from an input file specifying the region(s). You may use `guessline ??  1215` if you want to, but since that is a 'special' line in some sense, you then do have to specify that any further lines to be added have to be `??` as well, if that is what you want, using

```
f ad ??  ....
```

when starting the program (see Section 6.5).

7.5 Internal variables

There are a number of further internal variables, such as step sizes used for estimating derivatives, which may be changed. If you do this, it is important that you know why you are doing it and what the consequences might be - and this tends to mean you need some familiarity with how the code works. Few people do, so these are not normally prompted for, but will be if the line

```
pcvals
```

appears in the setup file. Then VPFIT asks for some setup parameters with

```
  setup: ? print values, <CR> OK, n,z,b,x4,cs,sf,il,w,me,p,d,v to change
```

A '?' prints the current default values, any of which may be changed by typing the letter code followed by the new value(s). Note that if this prompting is turned on, the damping of the matrix calculation is turned off. To reset it, type 'il'.

8 Spectral resolution

The spectral resolution may be specified in several ways, and, if you are content to use a Gaussian to describe the instrument profile, how it is done is a matter mostly of convenience. However, it *must* be specified - there is no default. If you fail to give a value one way or another, the program will prompt for a value for each fitting region (see section 8.4). This can be a pain, and is meant to be. The options for specifying the resolution are described below.

8.1 Resolution in FITS data header

The method which involves the least effort is to specify the resolution in the header of the FITS data file, as a constant velocity FWHM (in km s^{-1}) associated with the keyword RESVEL. The instrument profile is then assumed to be a Gaussian with that full-width-half-maximum. Since profile fitting is most useful where there is a hope of resolving the lines, the main application is for echelle data where constant velocity resolution is usually a reasonably good approximation.

Where there is more information on the instrument profile this can be put in a file and the file name associated with the key RESFILE in the header to the FITS data file. The file itself must reside in database/ relative to the FITS file location This is a hangover from an old IRAF-based structure, but it has not caused any problems.

There are three options for the file contents:

- As a table of wavelengths and corresponding resolution values, for a Gaussian instrument profile. Such a file must have a descriptor on the first line ('R' or 'r' for a table of resolution as $\lambda/\Delta\lambda$ where $\Delta\lambda$ is the instrument FWHM; or 't' for a table with FWHM velocities in km s^{-1}). Such a table just has wavelength/resolution pairs, one per line, and might look like:

```
Resolution
3000 45000
5760 45000
5820 42000
6700 42000
6900 45000
8600 45000
8650 40000
11000 40000
```

  A blank line acts as a terminator, as does the end of the file. The values may be real or integer, and (multiple) spaces or (single) commas may be used as separators, as for almost all VPFIT input. Don't use tabs though - they are not recognized.

  In VPFIT the resolution value used for each data region is determined at the central wavelength for that region, linearly interpolating between the two nearest values, or adopting the nearest value if the wavelength is out of range. It is assumed that the profile is a Gaussian.

- As a polynomial fit to the FWHM as a function of wavelength. In this case it is also assumed that the instrument profile is a Gaussian, with

$$\text{FWHM} = a_1 + a_2\lambda + a_3\lambda^2 ... + a_6\lambda^6$$

where the FWHM is in A, and the coefficients a_1, a_2 are from the file, one per line. There must be six of them, so pad out with zeros if necessary. The setup allows for multispectrum files, and the spectrum referred to is specified by row number in the FITS file. So for a standard single spectrum file, the resolution file in this format has fit coefficient sets separated by lines containing 'row nn'. So such a file might look like:

```
row 1
0.25
0
0
0
0
0
row 2
0.35
0
0
.....
```

This might be useful if you have constant wavelength resolution, or echelle spectra where the echelle orders are kept separately within the file, but since otherwise the polynomial fit has to be determined somehow the alternative format for the table given above is almost invariably preferable. It is really a leftover from the earliest versions of the program.

- A profile which is specified on a pixel-by-pixel (or sub-pixel) basis.

The resolution file need not contain just the fit coefficents for the Gaussian width line-spread-function, but can be a pixel by pixel description with a weight for each pixel specified. This is assumed if there is no RESFILE specified in the header for the data and the file contains more than one line of values, and is not invoked otherwise. This allows any instrument profile you want, particularly for those cases where a Gaussian might not be appropriate – all you have to do is measure it! It is assumed that the pixel profile given applies to the whole of the spectrum with which it is associated. If there are different profiles for different parts of the spectrum then you can either

  - have mutiple copies of the spectrum, or parts of it, each with RESFILE pointing to a different file, or
  - run from file with different filenames specified on the line containing the file and region information (see section 8.2).

A LSF which has as weights the ordinates of the normal curve with sigma=1 pixel at each pixel relative to a central peak could be written to a file (in /database) as:

```
-4 0.0001
-3 0.0044
-2 0.0540
-1 0.2420
0  0.3989
1 0.2420
2 0.0540
3 0.0044
4 0.0001
<EOF>
```

where <EOF> is the end-of-file. This pixel by pixel smoothing is then applied to the Voigt profiles on the continuum before comparison with the data. Note that the file should contain an odd number of pixel values for the LSF. The normalization does not matter, since the smoothed value at each point is divided by the summed weights.

The VPFIT program needs to know that the instrument profile is specified in pixels, and this is done via the file pointed to by the environment variable VPFSETUP (see section 7). This file **must** contain

```
nsubmin 1
nsubmax 1
```

If the instrument profile is not well sampled, then you should really choose finer pixels for the data (not helpful, but true). It is possible specify an instrument profile at a sub-pixel level, and have this applied to the data. Then it is a good idea to have an odd number of sub-pixels per pixel, and the format of the instrument profile file for nine sub-pixels per data pixel is

```
 -9.444445  0.000000
 -9.333333  0.000000
 -9.222222  0.000186
 -9.111112  0.000200
 -9.000000  0.000200
 -8.888889  0.000251
 -8.777778  0.000300
 -8.666667  0.000311
 -8.555555  0.000366
 -8.444445  0.000422
 -8.333333  0.000477
 -8.222222  0.000531
 -8.111112  0.000623
 -8.000000  0.000692
 -7.888889  0.000746
 -7.777778  0.000853
 -7.666667  0.000963
.....
 -0.555556  0.331896
 -0.444444  0.391605
 -0.333333  0.452252
 -0.222222  0.505960
```

```
 -0.111111  0.544142
  0.000000  0.559950
  0.111111  0.548307
  0.222222  0.514081
  0.333333  0.463878
  0.444444  0.405957
  0.555556  0.347922
.....
  9.222222  0.000200
  9.333333  0.000181
  9.444445  0.000077
<EOF>
```

This example is for STIS on HST, and the values given are in pixels (or fractions of a pixel) relative to the centre in the first column, and relative intensities in the second. The normalization does not matter - the sum is divided out. Note that the instrument profile need not be symmetric about the reference point, but if it is not you should think carefully about what the reference wavelength really means!

If a pixel profile is used it is **vitally important** that the VPFIT program profile generation sub-stepping be the same as the instrument profile. So, in the VPFSETUP file the minimum and maximum substepping allowed must be equal, and set to the value appropriate for the instrument profile. So, for the STIS example above, the VPFSETUP file **must** contain

```
nsubmin 9
nsubmax 9
```

If you have different data files with instrument profiles specified at different sub-pixel levels, or (as is often the case) specified at intervals which are not exact integer fractions of a pixel, then (for now at least) **you have to interpolate the instrument profile on to a uniform scale which corresponds to a single integer fraction of a data pixel**. The program will not do it for you.

8.2 Specifying resolution in startup files

It is possible to specify the values for the resolution by using the key '$vsig = nn.nn$', '$vfwhm = nn.nn$', '$wfwhm = nn.nn$' or '$wsig = nn.nn$' in an input startup file which contains first guesses. If the resolution is set in this way, any resolution specified in the FITS header is not used. Here $vsig$ is the Gaussian σ in km s^{-1}, assumed constant over the region to which it applies, and $wsig$ is σ in A, again assumed constant over the region. The FWHM descriptors are similar, but with the FWHM input instead of the σ.

An example of an input file using this approach is:

```
%% HE0515m4414.fits     1   6012.49   6017.67 vsig=2.8
%% HE0515m4414.fits     1   6027.88   6033.26 vsig=2.7
MgII  1.150251  0.000003   5.24   1.13  10.962  0.075 0
MgII  1.150341  0.000005   1.91   1.20  10.887  0.176 0
```

```
MgII  1.150407  0.000002   4.96   1.43  11.494  0.083 0
MgII  1.150479  0.000003   2.84   1.55  11.197  0.152 0
MgII  1.150546  0.000001   3.44   0.21  12.164  0.014 0
MgII  1.150634  0.000004   4.57   0.40  12.275  0.093 0
MgII  1.150685  0.000001   3.20   0.53  12.478  0.184 0
MgII  1.150744  0.000019   4.78   3.31  12.557  1.412 0
```

This can be useful if you are not sure of the resolution and want to try a few cases.

A new pixel instrument profile can also be pointed to from the input startup file, using $pfin =<$ filename $>$. This can be set to point to a different file for each fitting region, but the (sub-) pixellation of the instrument profile must be the same for all regions. Then, as an example:

```
%% HE0515m4414.fits     1   6012.49   6017.67 pfinst=fileres1.dat
%% HE0515m4414.fits     1   6027.88   6033.26 pfinst=fileres2.dat
MgII  1.150251  0.000003   5.24   1.13  10.962  0.075 0
MgII  1.150341  0.000005   1.91   1.20  10.887  0.176 0
 ....
```

8.2.1 Wavelength-dependent LSFs

In this mode there is also the option of having the program interpolate between line spread function arrays provided at different wavelengths to the central wavelength of the region being fitted. To do this you obviously need a table of LSF values in a file, and this has the general form (the example is from COS g130m):

```
!   1150         1200         1250         1300         1350       1400       1450
1  7.106e-05    7.117e-05    7.099e-05    7.045e-05    7.035e-05  7.084e-05  6.879e-05
2  7.269e-05    7.280e-05    7.261e-05    7.206e-05    7.196e-05  7.246e-05  7.036e-05
3  7.437e-05    7.448e-05    7.429e-05    7.372e-05    7.362e-05  7.414e-05  7.199e-05
4  7.610e-05    7.622e-05    7.602e-05    7.545e-05    7.534e-05  7.587e-05  7.367e-05
5  7.790e-05    7.802e-05    7.782e-05    7.723e-05    7.712e-05  7.766e-05  7.541e-05
6  7.975e-05    7.988e-05    7.967e-05    7.907e-05    7.896e-05  7.951e-05  7.720e-05
 ....
```

The first line is vital. It gives up to 8 wavelengths at which the LSF is specified, and must start with "!", "#" or "%" followed by a space. If you omit the wavelength line the program will look for g130m or g160m in the filename, and, if it is there, insert 50 A spaced values starting at 1150 or 1450 for you. **Note that this application was originally designed for COS spectra, and sub-pixellation is not supported**. If you want different sub-pixel LSFs then you need to create separate files containing the different ones needed and use "pfinst=" to point to the appropriate one for the given spectral region.

Then the implementation is very like the "pfinst=" one above, with "wdLSF=" instead, so the input file now looks like:

```
%% HE0515m4414.fits     1   6012.49   6017.67 wdLSF=fileres1.dat
%% HE0515m4414.fits     1   6027.88   6033.26 wdlsf=fileres1.dat
MgII  1.150251  0.000003   5.24   1.13  10.962  0.075 0
MgII  1.150341  0.000005   1.91   1.20  10.887  0.176 0
 ....
```

The file pointer can be in upper or lower case, or as"wdLSF". The line spread function is calculated for the midpoint wavelength for each region, and applied to the whole region. You can have as many different filenames as you want, so e.g. the second "fileres1" could be "fileres2". If there are more than 8 wavelengths at which the LSF is specified you will need to split the table up to cover as many files as needed.

8.3 Mixed PSF subsampling

The input file resolution mode also allows you to mix the subsampling of the instrument PSF if you wish to. So, e.g. if you are simultaneously fitting a feature with a STIS spectrum where the LSF is subsampled over 13 steps and a COS spectrum with LSF specified per pixel then you can do it by using the startup file:

```
%% cosg130m.fits     1   1243.2026   1246.4000 wdLSF=cosg130m.dat
%% stisspec.fits     1   1243   1246 pfinst=stispsf.dat
 H I      0.0233915950    0.0    68.67645      0  12.903742    0
```

with cosg13m.dat containing the line (after the wavelength line!)

```
nsub 1
```

and stispsf.dat containing the line

```
nsub 13
```

8.4 Resolution unspecified

Knowledge of the instrument resolution is necessary for matching the Voigt profiles to the data, so it *must* be given. If you don't specify it anywhere the program prompts for a value, with the line:

```
FWHM (km/s & A) were    0.0  0.000 <CR> to accept, or enter values
```

The first number is FWHM in km/s, so if you have constant velocity resolution (e.g. from a high order echelle spectrograph) then just enter the value e.g.

```
6.7
```

If it is a constant delta wavelength, then use e.g.

```
0.0 0.12
```

If you enter two values on this line, then the FWHM at any wavelength is the sum of the two.

8.5 Rescaling the resolution

If the instrument profile is genuinely slit limited then the resolution should be determinable. However just to get adequate S/N the seeing may be comparable to, or less than, the slit width, and in these cases the true instrumental profile might not be well known. If you are unsure of the resolution, but think you have an idea of how it varies between regions, then you can rescale all the instrument FWHM's quoted in the startup file by a constant multiplicative factor. You do this by starting VPFIT with a parameter which is the multiplier set by 'vfwx=<real number>' (with no spaces). So when invoking vpfit from unix you might have a command line

```
vpfit vfwx=0.8
```

if you want to multiply all the FWHM values in the startup file by 0.8. Note that this works **ONLY** for values specified as 'vfwhm=xxx' in the startup file - not for this specified in the header, not for 'vsig=', not for wavelength resolution, and not for pixel or subpixel (pfinst) resolution. It is there so you can rerun things to test how resolution uncertainties might affect things like component decomposition and minimum χ^2 with various values for the resolution. The summary output (see section 9.2) tells you what the factor is if it is not unity, but leaves the individual 'vfwhm=' values unchanged in that file.

Of course in principle the best fit multiplier could be solved for, but we thought that could obscure what might be going on and there might be an unjustified blind belief in the results. One problem is if the lines are optically thin then there is degeneracy between the b-parameter and the width of the line-spread-function. If you are fitting a mix of optically thin and thick lines from the same ion then, since the curve of growth dominates the results, then such degeneracy should not occur, but trials suggest that you then gain very little which is believable especially if the lines have velocity structure. In such cases the best fit line line spread function can depend on how much structure you put in, which adds to the overall uncertainty. So please use this option with caution, and think about what you are doing.

9 Output

9.1 Screen output

A running commentary giving the numerical results for each iteration is normally written to the screen. The initial information depends on the mode of use (basically 'from file' or 'interactive'), then during the fitting process you can monitor the normalized χ^2 value at each iteration to see how it is approaching (or not) unity, and the fit parameters for each ion in each system chosen are also shown. The iteration output looks like

```
 iteration   :   4 ( 1 )
 chi-squared :          1.031 (       119.5556,   116 )

   C IV     12.08718     3.2558106      9.6634        0.00   0.00E+00  0 !    1
   C IV     12.63509     3.2560551      8.3725        0.00   0.00E+00  0 !    2
   C IV     12.67623     3.2563261     43.2974        0.00   0.00E+00  0 !    3

....
```

This example shows iteration 4 of set 1, where the set number will change only in the automatic add or remove ions mode. The normalized χ^2 is given, along with the actual χ^2 and number of degrees of freedom. Then for each ion in each system, log column density, redshift and Doppler parameter. The next two numbers are turbulent velocity and temperature if those two quantities are computed (see 11.3), and the '0' is a rarely used region pointer (see 9.2 below). The final numbers simply give a line count, and ID number for use with the plot of the final fit.

This output is also written to a file, `fort.18`. I can be suppressed by including the line

```
nopchan 0
```

in the VPFSETUP file. Replacing the '0' by '1' gives output to the screen only, and '2' is for screen plus file.

When the χ^2 step per iteration is small enough the process stops, with something like

```
 iteration   :   9 ( 1 )
 chi-squared :          0.848 (        98.3189,   116 )

   C IV     12.01560     3.2557412      6.5453        0.00   0.00E+00  0 !    1
   C IV     12.83274     3.2560529     11.3139        0.00   0.00E+00  0 !    2
   C IV     12.31592     3.2565894     16.7572        0.00   0.00E+00  0 !    3

 Rescaled parameter errors:
   C IV      0.12455     0.0000225      2.5080        0.00   0.00E+00    !    1
   C IV      0.02565     0.0000067      0.9479        0.00   0.00E+00    !    2
   C IV      0.06177     0.0000276      3.1832        0.00   0.00E+00    !    3

 statistics for whole fit:
  Runs test  K-S test Chi-squared  Chans ndf    APr   Xp(.68) Xp(.95) Xp(.99)
   0.16177   0.93632     98.32     125  116   0.881  122.75  141.85  153.63
```

```
 Statistics for each region :
    Start      End    Chi-squared  Chans df?
   6587.13   6591.58     48.87      62   53   0.635 < Prob < 0.887 g= 0.635   1
   6587.13   maxdev       0.7198             0.678   0.956   1.358   1.627

   6598.24   6602.75     49.45      63   54   0.650 < Prob < 0.894 g= 0.650   2
   6598.24   maxdev       0.5291             0.942   0.956   1.358   1.627

....
```

The estimated parameter errors follow the best fit parameters for the final iteration. The term `Rescaled parameter errors` reflects the fact that the diagonal terms in the Hessian matrix have been rescaled by multiplying by the normalized χ^2 value, so the error estimates are still reasonably good even if the data error array is incorrect by some factor, as is roughly the case for rebinned data where the only error array available is based on photon statistics. If the data has come from the UVES_popler package this rescaling is not done (and the word 'Rescaled' no longer appears). There is a similar indication as to whether or not the errors have been rescaled in the summary file (see section 9.2) if the change is significant.

Note that the error estimates given are *only* from the diagonal terms in the Hessian matrix, so if the variables used are not independent then they could well be underestimates. If you are doing something where the true errors are important then you should try to find out what they really are by e.g. Monte-Carlo trials with simulated data, though setting those up realistically may not be easy.

The fit statistics given in the screen (and fort.18) output are the things to look at to see if a fit is acceptable, and the main one is the χ^2 value and the number of degrees of freedom (ndf). A probability that the data matches the fit given these values is computed (APr), and χ^2 values for three probabilities is listed. You can choose whatever acceptance criterion you like - APr$>$ 0.01 is a reasonable value.

There is a region-by region statistics breakdown, so you can see the bad regions, but here the probabilities are given as possible ranges because the degrees of freedom per region is not calculable. There is a runs test and K-S test probability given as well, but these are more prone to suffer from smoothing effects in rebinned data so are usually ignored. The nature of the runs test in particular is lost in the mists of time, so I'm not even sure what it means any more.

9.2 Summary output

At the end of the fitting a summary is printed out. Precisely what you get may be determined by parameters in the VPFSETUP file (see section 7). The default is to write to (and over-write) a file called `fort.26`, but if you want a named file for each fit set then in the VPFSETUP file include the line

```
wr26s f26.w
```

or something similar. The final results are then written to a file `f26.wnnnn`, where `nnnn` is the nearest integer central wavelength for the first of the regions of those included

in the fit. The first lines of the output file start with '%%' and give the data file and fitted region limits. These are followed by a comment line with overall fit statistics - iterations, normalized χ^2, number of pixels, number of degrees of freedom, chance probability that the fit is OK, and number of systems dropped in finding solution. Then come the parameters for each fitted ion, with each line giving ion, redshift, $\pm$, Doppler parameter in km s^{-1}, $\pm$, log column density in cm^{-2}, $\pm$. The final number on that line is almost invariably zero [it is a region pointer where an ion is to be included in a single region only, a somewhat unphysical situation which has been used for testing purposes].

```
%% HE0515rc.fits     1   5562.1941   5566.0996
%% HE0515rc.fits     1   5590.8700   5595.3470
! Stats:   4     1.1842712  605  428  0.005  0
 FeII     1.150407    0.000004     3.83      1.18  11.315    0.086  0 !
 FeII     1.150484    0.000016     2.90      3.71  11.037    0.368  0 !
.....
```

This is OK normally, but occasionally the errors turn out to be zero at this precision, so more significant figures should be printed. You can set the program by including the line

```
vform
```

in the VPFSETUP file, and then the number of decimal places is increased (for the redshift to 8 or 10) depending on the redshift error. If you want to have the maximum number of significant figures every time, then `vform long` in the setup file will do it. The summary file then looks like

```
%% HE0515rc.fits     1   5562.1941   5566.0996
%% HE0515rc.fits     1   5590.8700   5595.3470
! Stats:   4     1.1842712  605  428  0.005  0
FeII     1.1504065320    0.0000038797     3.82912    1.17982  11.315082    0.086321  0 !
FeII     1.1504837563    0.0000156551     2.89745    3.70543  11.036918    0.367643  0 !
.....
```

Whatever the precision, this summary file can be used, either as it stands or with values changed, or lines added, to restart VPFIT in the 'Read from file' mode.

9.3 Looking at the fit results

After the fitting process is finished you are given the opportunity to look at the results in a PGPLOT graphics window if you want to. This is done region by region, using the same command line plotting routine as the RDGEN program. If you have more than one region involved in the fit, you are prompted for which one you want.

```
  Plot? y,n, or c=change device [y]
>
  Identify which dataset:
  # ID   Filename       data cont    lamcoeffs      nchans  KS prob
  1  1    filename1.fits   1   1    5454.23  0.0455 126911    0.533
  2  2    filename1.fits   1   1    5468.44  0.0456 126911    0.968
```

```
 Which # ? [ 1 ]
>
 plot parameter? (type he for options list)
>

PGPLOT device? [/xwindow]
>
  Plot? y,n, or c=change device [y]
>
```

The > prompt is, in this example, followed by a carriage return, since usually the defaults are sensible. It is not normally worth using the full range of plot parameters, since the program displays the fit region you have chosen, but they are all there in case you want to examine something more closely. Type ? for help if you want it. Fuller details are given in the RDGEN description.

The plots from VPFIT mark the line positions with two numbers - the top is just the system number in the output list order, and the lower one identifies the line, by number for the given ion, in the atomic data list. So, for example, CIV 1548 would be labelled 1, 1550 labelled 2, SiII 1808 2, SiII 1526 3 etc.

If you want an ASCII file of the plotted quantities, then

`as`

at the 'plot parameter' prompt will result in the program writing the results to fort.17 instead of the PGPLOT window or device, without closing the plot device. To restore plotting to that device simply enter `as` again when you want that to happen. The fort.17 format is

```
wavelength   data   error   fitted profile
```

for the region plotted (possibly with silly numbers for the fitted profile outside the fit range), followed by a list of tick mark positions with the line ID numbers as described above.

10 Subpixel profiles

Lines with FWHM $\sim$ a few pixels or less were dealt with incorrectly by versions of VPFIT earlier than VPFIT9. There is still a potential problem with very narrow lines, but the constraint is now the patience of the person running the program rather than something intrinsic to the method used.

The program resamples pixels according to the smallest b-value, so that there are a minimum number of pixels per FWHM ($= 1.68 \times b$). The values for internal substepping can be set in the setup file associated with the environment variable VPFSETUP. The defaults are:

!		— internal substepping —
nsubmin	1	minimum number of subpixels per pixel
nsubmax	11	maximum number of subpixels per data pixel
nfwhmp	7	require at least 7 pixels per line fwhm

The subpixel profile is built up for all lines, convolved with the instrument profile, and then the result is converted back to the original pixellation for comparison with the data. If you want to speed things up, with a possible loss of accuracy, set $nsubmax$ to a lower value ($\geq nsubmin$), or choose a lower value for $nfwhmp$, or both.

If you are using a pixel-based instrument profile, then set $nsubmin = nsubmax =$ profile sampling per pixel. If you are worried about intrinsically narrow lines, then you can interpolate the instrument profile on to a finer sampling per pixel.

11 Fixed & tied parameters

It is possible to fix any parameter at some specific value, and to tie parameters so that the have the same value, or specific relationships with each other.

11.1 Fixed parameters

On occasion it is useful to fix the redshift or the Doppler parameter because you know them accurately from fits to other lines.. e.g. z and b from SiII, and you wish to apply these to OI or CII. Fixed parameters are indicated by UPPER CASE letters after the parameters in question. For the input (fort.13) file the parameter lines could look like:

```
O I 14.043 4.382795E 4.25K
O I 14.043 4.382893J 4.25N
O I 13.031 4.383101S 5.75
O I 12.346 4.383487 13.04
```

.. in which case the redshifts and the Doppler parameters are fixed, but the column densities are free to vary for the first two, the redshift is fixed and the Doppler parameter and column density vary for the third, and everything varies in the fit for the fourth.

11.2 Tied parameters

Another thing you might want to do is to tie parameters – for example demand that CII and SiII always have the same redshift. This is done by the same letter approach, but now one thinks of a reference ion (with lower case letters) and others tied to it (by the corresponding upper case ones), so the ion detail lines in fort.13 become like:

```
C II 14.043 4.382893j 4.25 0.00 1.00E+00 0
C II 15.568 4.383865c 11.32 0.00 1.00E+00 0
SiII 14.043 4.382893J 4.25 0.00 1.00E+00 0
SiII 13.276 4.383865C 11.32 0.00 1.00E+00 0
```

Here SiII and CII are at the same redshift, though this common redshift is allowed to vary. The other parameters vary independently.

If you want CII and SiII to also have the same Doppler parameters (or Doppler parameters in the ratio set by the square root of their mass ratios to mimic thermal broadening), this is done by the same letter approach applied to the b-values, e.g. for the above example:

```
C II 14.043 4.382893j 4.25k 0.00 0.00E+00 0
C II 15.568 4.383865c 11.32n 0.00 0.00E+00 0
SiII 14.043 4.382893J 4.25K 0.00 0.00E+00 0
SiII 13.276 4.383865C 11.32N 0.00 0.00E+00 0
```

.. here the redshifts and the Doppler parameters are tied. More information is needed to tie them sensibly, and this is provided by the extra numbers after the Doppler parameters, here 0.00 and 0.00E+00. These are the assumed turbulent Doppler parameters and temperatures if non-zero. So, for example, if you want the temperature to be 10^4 K, then replace the 0.00E+00 by 1.00E4. The turbulent component will then be set to whatever is appropriate for the given ion. Similarly, if a turbulent component of $3.0\,\mathrm{km\,s^{-1}}$ is desired, the '0.00 0.00E+00' becomes '3.0 0.0' (as you can see, the format for the numbers does not matter). Where both are zero (as here) it is assumed that the Doppler parameters are thermally linked. So in this example $b(\mathrm{CII}) = \sqrt{28/12}b(\mathrm{SiII})$ - the program will put in the correct relative values, using the lower case flagged ones as starting guesses.

If you want CII and SiII to have the same Doppler parameter, then use e.g.

```
C II 14.043 4.382893j 4.25k 0.00 1.00E+00 0
C II 15.568 4.383865c 11.32n 0.00 1.00E+00 0
SiII 14.043 4.382893J 4.25K 0.00 1.00E+00 0
SiII 13.276 4.383865C 11.32N 0.00 1.00E+00 0
```

The mysterious 1.00E+00 is the ASSUMED temperature, which is fixed at 1K in this case so that the Doppler parameters are effectively the same, i.e. all the b is turbulent. 1K is probably a silly temperature in reality - in fact it is a good idea to make sure things which are turbulently linked don't have temperatures lower than those you would expect for the ions covered (e.g. $\sim$ 5000 - 10^4K or so for warm neutral medium gas containing e.g. CII, SiII etc), and for any thermally linked the temperature is within a sensible range. For a temperature T the thermal Doppler parameter is $b = 12.85\sqrt{\frac{T}{10^4 m}}$ km s^{-1}, where m is the atomic mass in a.m.u.

You can also tie Doppler parameters at different redshifts if you want to, but it is not at all clear what that would mean.

The output gives the best fit Doppler parameters given the constraints. It does not give a temperature estimate directly, but this can be inferred from the Doppler parameters.

There are some things to be aware of:

- Do not use the same lower case letter more than once, since then the program does not know what you are trying to link a variable to. This does not mean you are restricted to 26 linked parameters however - the "j" in the example above could be replaced by a two-letter combination (e.g. "ja"), and in such a case "J" should be changed to "JA". This means there are 676 independent tied or fixed systems possible, which should be plenty!

- Note that you can't fix both the turbulent and thermal Doppler parameters at the same time, because this fixes the total b-value (which you can do, as above). If the turbulent value input is non-zero, then the temperature value is ignored, whatever it is, and the program treats it as an input value zero. Note that the letter should be earlier in the alphabet than the 'lastchtied' one in the setup file, since if not it is liable to try fitting a turbulent and thermal component simultaneously using a different prescription.

- The example above is fine for getting fully turbulent linked Doppler parameters, but it breaks down if you want to fix the temperature to be some more realistic value. If you want the temperature to be 5000K, for example, then you must use the other input format. So the block above becomes, for that temperature,

```
C II 4.382893j 0  4.25k 0 14.043 0 0 [ 0.00 0 5.00E+03 0
C II 4.383865c 0 11.32n 0 15.568 0 0 [ 0.00 0 5.00E+03 0
SiII 4.382893J 0  4.25K 0 14.043 0
SiII 4.383865C 0 11.32N 0 13.276 0
```

  If you don't do this the values are loaded into the wrong variables, so e.g. the column density becomes the first guess at the redshift. You will notice this when the program starts to go silly, if not before!

- In this case the column densities are free parameters, but you can fix and tie these as well if you want to. Fixing is obvious, but tied column densities is not something which has been wanted often. There is one possibility which is catered for - demanding that ion ratios within a complex velocity system are the same for selected components of that complex. This is described in subsection 11.5

... and a warning.

Constraining Doppler parameters to be thermal or turbulently linked, or a mixture, is making an assumption about the region velocity structure. It is somewhat dangerous to assume a Doppler parameter for some species based on others and then infer a column density for it, unless the lines involved are on the linear part of the curve of growth when the Doppler parameter is not relevant. For narrow, saturated lines in particular you should be especially careful, since an thermal linkage e.g. between CII and SiII (based on SiII Doppler parameter from several lines, saturated and unsaturated) could give CII column densities about an order of magnitude lower than a turbulent linkage.

11.3 Temperature estimation

Temperatures and turbulent components may be estimated directly using tied ions of different mass. If the first indicator letter is later in the alphabet than 'lastchtied' in the setup file, then the Doppler parameters b are determined assuming that

$$b^2 = b_{\rm turb}^2 + \frac{2kT}{m}$$

where m is the mass of the ion and T is the temperature. Since this requires two parameters there are now two reference ions where the Doppler parameter is labelled with lower case letters, and these two ions must have different atomic masses. The rest in the sequence have Doppler parameters marked with the corresponding upper case letters.

An example startup file for doing this (in summary format) is

```
%% HE0515m4414      1    6130.8716    6131.7505
```

```
.....
%% HE0515m4414      1    5037.6309    5038.2656
 FeII      1.149084aa  0.000000      2.08wa     0.00   12.57     0.03   0
 AlII      1.149084AA  0.000000      2.55WA     0.00   11.94     0.04   0
 MgI       1.149084AA  0.000000      2.63WA     0.00   10.64     0.09   0
 MgII      1.149084AA  0.000000      2.63wa     0.00   12.50     0.10   0
 FeII      1.149115ab  0.000000      4.84wb     0.00   12.43     0.04   0
 MgII      1.149115AB  0.000000      5.90wb     0.00   13.09     0.03   0
 MgI       1.149115AB  0.000000      5.90WB     0.00   11.07     0.04   0
 AlII      1.149115AB  0.000000      5.73WB     0.00   11.91     0.04   0
```

The output summary from this then gives the fit parameter estimates in the form

[ion	z	±	b	±	log N	±		$b_{\rm turb}$	±	T	±]
FeII	1.149084aa	0.000001	2.08wa	0.00	12.571	0.029	0 [	1.534	0.301	6.697E+03	2.949E+03
AlII	1.149084AA	0.000000	2.55WA	0.00	11.942	0.037	0 !				
MgI	1.149084AA	0.000000	2.63WA	0.00	10.641	0.090	0 !				
MgII	1.149084AA	0.000000	2.63wa	0.00	12.504	0.099	0 [	1.534	0.301	6.697E+03	2.949E+03
FeII	1.149115ab	0.000003	4.84wb	0.00	12.430	0.040	0 [	3.827	0.561	2.956E+04	6.664E+03
MgII	1.149115AB	0.000000	5.90wb	0.00	13.092	0.034	0 [	3.827	0.561	2.956E+04	6.664E+03
MgI	1.149115AB	0.000000	5.90WB	0.00	11.071	0.037	0 !				
AlII	1.149115AB	0.000000	5.73WB	0.00	11.909	0.040	0 !				

Note that the error estimates for the individual ion Doppler parameters are set to zero. Zero in this context does not mean they are zero, it simply means they are not computed because they are now derived variables. The error estimates are now associated with the turbulent Doppler parameter and the temperature, in the columns beyond the '['.

If you are using upper and lower limit presets for the Doppler parameter then these are not applied for the temperature and so the Doppler parameter values may go outside the range you expect.

The temperature and the turbulent Doppler parameter are both constrained to be positive, but otherwise nothing further is assumed. If there are not two ions for which the Doppler parameter is not well constrained the error estimates may be very large, but at least the results will be physically consistent.

Error estimates

If the turbulent component or temperature is near zero, then the error estimates for both quantities may be wildly inaccurate. In software-speak this is a known problem, and someday it may be looked at. In the meantime, or in any case, it is probably worth doing an independent check by stepping through a range and looking at how χ^2 varies, or running simulations to get reasonable estimates.

Health warning.

The intrinsic line widths depend in similar ways on the turbulent component and the (square root of the) temperature, so there is some degeneracy between the two. If you are trying this approach you need to make sure that the ions used as baselines have different masses **AND** the Doppler parameters for each are very well constrained. If they are not then you may get (physically consistent) gibberish. For most cases where the lines are unresolved, this will require simultaneous fitting of lines on both the linear and logaithmic parts of the curve of growth, so at least with widely separated $\log(f\lambda)$ values. Otherwise the results could be horribly dependent on e.g. the assumed instrument profile.

11.4 Summed column densities

If you are dealing with a multicomponent complex then the column densities of the individual components may have large error estimates because of their proximity to other velocity components. The total column density may be much better constrained however, and for abundance studies you may be interested in the overall relative column densities. The column density sum, and its error, can be computed directly within the program.

The vp_setup.dat file (section 7) should contain LASTCHTIED v – or whatever letter you choose. Later letters in the alphabet are then treated in a special way for tieing column densities so that the first one in the block gives the TOTAL column density for the complex. The redshift and Doppler parameter still apply to that individual component, which now has a column density equal to the list value minus the sum of all the rest with the same identifying letter. Thus order in the `fort.13` file is important!

For example, if you want to know the total column density on the assumption that an HI system is made up of $b = 13$ clouds, then you would put in `fort.13` something like:

```
H I 13.905 3.320368 9.65
H I 16.76x 3.320970 13.00F
H I 15.91x 3.321462 13.00G
```

The 16.76 is then the sum of the column densities in the two 'x' systems. This may seem pointless, but in fact is useful for close blends in which the error estimates for the individual components are large because of difficulties in resolving the blend, while the total column density is much better constrained. In the output file the error against the first column density is now the error in this total.

As with other tied parameters, two letters may be used, so 'xa' is treated separately from e.g. 'xb'. Then only the first of the two letters is checked to see if sums are required, so for e.g. 'ax' the column density sums are NOT used. For comparing total column densities in a double system with common redshifts (and turbulent broadening of the lines) one might start with:

```
C IV 14.63xa 2.765821aa 12.69ia 0.00 1.00E+00 0
C IV 13.89xa 2.765965ab 17.73jb 0.00 1.00E+00 0
SiIV 13.96xb 2.765821AA 12.69IA 0.00 1.00E+00 0
SiIV 13.10xb 2.765965AB 17.73JB 0.00 1.00E+00 0
```

The ordering for both the `xa` and `xb` sets is important. Note that the flags against the column densities are all lower case. There is no notion of a main component which others are tied to (as here for the redshifts and Doppler parameters), since all the parameters are still free. It is only that the first in the list is computed in a different way.

You can extend this to as many CIV as you like up to the limit of the total number of parameters, but they must appear as a single block in the file of starting guesses. Same is true for the SiIV in this example. And other lines which may be blended can also appear in the list, but not within any block, so

```
C IV 14.63xa 2.765821aa 12.69ia 0.00 1.00E+00 0
C IV 13.89xa 2.765965ab 17.73jb 0.00 1.00E+00 0
```

```
H I  13.02   3.345500   25.33
SiIV 13.96xb 2.765821AA 12.69IA 0.00 1.00E+00 0
SiIV 13.10xb 2.765965AB 17.73JB 0.00 1.00E+00 0
```

is acceptable, but

```
C IV 14.63xa 2.765821aa 12.69ia 0.00 1.00E+00 0
C IV 13.89xa 2.765965ab 17.73jb 0.00 1.00E+00 0
SiIV 13.96xb 2.765821AA 12.69IA 0.00 1.00E+00 0
H I  13.02   3.345500   25.33
SiIV 13.10xb 2.765965AB 17.73JB 0.00 1.00E+00 0
```

is not.

The final thing to note is that if the first column density in a list tied in this way is smaller than the sum of the later ones in the list, then the value is taken as the column density for that component, and for the output the quantity printed is replaced by the sum. So, if one had

```
C IV 12.50xa 2.765821 12.69 0.00 1.00E+00 0
C IV 12.60xa 2.765965 17.73 0.00 1.00E+00 0
C IV 12.60xa 2.766165 10.73 0.00 1.00E+00 0
```

it would be converted to

```
C IV 13.05xa 2.765821 12.69 0.00 1.00E+00 0
C IV 12.60xa 2.765965 17.73 0.00 1.00E+00 0
C IV 12.60xa 2.766165 10.73 0.00 1.00E+00 0
```

and then a best fit solution sought.

If the first column density in the list is larger than the sum of the rest, then that column density is not modified.

11.5 Common pattern relative ion abundances

Now to tie different ions so that the relative numbers for each ion remains the same across the complex, but the total column densities may vary. Given the range of possible ionizations and dust depletions this might not be very physically reasonable unless you choose the ions carefully, but it might be worth a try sometimes. You can do this by having fort.13 containing e.g.:

```
...
C IV 12.10  2.765701  14.81
C IV 14.63x 2.765821a 12.69i
C IV 13.89x 2.765965b 17.73j
C IV 12.89x 2.765995c  8.31k
SiIV 13.96% 2.765821A 12.69I
SiIV 13.10X 2.765965B 17.73J
SiIV 12.10X 2.765995C  8.31K
MgII 12.35  0.876844   5.37
...
```

The CIV grouping is as above, and the SiIV group must have the same number of ions appearing in the same order, *immediately* after the group it is tied to (the CIV one in this case). The '%' is just a marker, to say this is a new group for which the column density for this line will be the total, and all the following ones with a captial letter corresponding to the lower case ones above are to be tied, so that the SiIV/CIV ratio for each component is constrained to have the same value. The total column density ratio SiIV/CIV is allowed to vary, and a best-fit value sought.

If you have any spare ions in either system, as for the first CIV here, do not place them between the two tied blocks. That CIV won't be regarded as part of the total, but if you want totals you can either include anothe tied SiIV and include both in the tied blocks, or add them later by hand.

This will result in the first lines for each ion containing TOTAL column densities for that ion in the complex, and, in this example, the SiIV and CIV are constrained to have the same ratio in each velocity component. Note that for your first guesses you don't have to get the ratios right – the program sorts it out by making all ratios match the first before starting to iterate.

Note forcing the ratios to be the same will affect the error estimates .. things are no longer independent. The errors flagged with '%' are those in the total column density given the ratios between components as error-free (I think).

As in the previous section, the column densities for the first lines for each different ion are adjusted if they are less than the sum of those further down the list.

Again the order is vitally important, and here the '%' block must immediately follow the base block i.e. in this example, the SiIV block immediately follows the CIV block, with the same redshift order. You cannot put other ions between the blocks, or within the blocks, since the association is done by positions within the blocks. If there are other blended lines, they can be listed either before or after the tied blocks.

Also, the number of components for both ions must match. In this example, if there are 2 CIVs, there must be 2 SiIVs. If there were 5 CIVs, then you would need 5 SiIVs. Use the stronger ones as the base ones and you should be able to avoid problems with weak lines being thrown out by the program.

You can extend this to several groups, so e.g.

```
C IV 14.63x 2.765821aa 12.69i 0.00 1.00E+00
C IV 13.89x 2.765965ab 17.73j 0.00 1.00E+00
SiIV 13.96% 2.765821AA 12.69I 0.00 1.00E+00
SiIV 13.10X 2.765965AB 17.73J 0.00 1.00E+00
H I  15.67% 2.765821AA 12.69I 0.00 1.00E+00
H I  14.34X 2.765965AB 17.73J 0.00 1.00E+00
C IV 14.92y 2.775821ac 10.39bc 0.00 1.00E+00
C IV 13.11y 2.775965ad 13.73bd 0.00 1.00E+00
SiIV 13.96% 2.775821AC 10.39BC 0.00 1.00E+00
SiIV 12.14Y 2.775965AD 13.73BD 0.00 1.00E+00
H I  15.87% 2.775821AC 10.39BC 0.00 1.00E+00
H I  14.06Y 2.775965AD 13.73BD 0.00 1.00E+00
```

would result in two groups with CIV, SiIV and HI all following the same relative column

density pattern in each of the two groups, with the first for each ion being the sum as before. Note that for multiple groups like this with the '%' you need to use a single letter marker for the column densities - hence the 'x' and 'y' in the example above. You can put a second letter in, but it may be ignored for some aspects, and not for others (so don't).

11.6 Fixing relative column densities

This is probably the last refuge of the desperate, but if you really want to you can force column densities to have constant ratios (or constant differences in log space). Thus if you firmly believe that the molecular hydrogen you are trying to measure has a temperature of e.g. 100K then you can force it to be so by having in initial guess file containing:

```
H2J0 14.63ca 2.765821aa 2.00ba
H2J1 14.83CA 2.765821AA 2.00BA
H2J2 13.11CA 2.765821AA 2.00BA
```

Note that the first letter of the tied parameter sequence must be earlier in the alphabet (or the same as) than that set as LASTCHTIED (see section 11.4).

Then there are two approaches. If you run VPFIT with such a start file, then the H2J0, H2J1 and H2J2 column densities will all vary with the constraints that $\log N(\mathrm{H2J0}) - \log N(\mathrm{H2J1})$ and $\log N(\mathrm{H2J0}) - \log N(\mathrm{H2J2})$ will always be the same as in the startup file (i.e. -0.2 and 1.52 in this example).

However, if there is a a file called vp_abund.dat in the directory you are running the program in that file will be used instead. Here you need to set out the log offsets for the column densities relative to the base - here J=0. So in this example vp_abund.dat would contain

```
H2J1 0.2 CA AA
H2J2 -1.52 CA AA
```

There is no need to specify the H2J0 value - it is zero by definition. The two letter code - here CA - tells the program which is the base, and the code and ion match tells it which number to use. The redshift tied indicators are also checked, so must be included as well - with the same case as in the startup file.

You can also fix the patterns for the same ions across a range of redshifts if you want to, but only if the redshifts are fixed or tied so that the program knows which redshift to associate with the ion. The tied redshift indicators are checked for consistency as well. So, e.g. a startup file with

```
H I      18.02628CA   3.0861225AF   15.0000BB
H I      17.88598CA   3.0863288AM   15.0000BB
H I      17.70438CA   3.0864430AN   15.0000BB
H I      17.83538CA   3.0865933AH   15.0000BB
H I      17.42798CA   3.0867457AK   15.0000BB
```

```
H I      19.26968ca   3.0881780AA   15.0000BB
H I      19.00098CA   3.0883555AB   15.0000BB
H I      18.17518CA   3.0884121DB   15.0000BB
H I      17.34698CA   3.0884494AP   15.0000BB
H I      16.46248CA   3.0893039AO   15.0000BB
```

will maintain the relative HI column densities while adjusting the base one (labelled 'ca') for the best fit.

11.7 Automatic tied component rejection

If you use a *fort.13* file to run your fit and rejection of tied systems is a possibility (e.g. because the column density is too low) then it may be that a system where the lower case tied variable is used will be rejected. If this happens, then, for the redshift and Doppler parameters, the program reassigns the lower case variable to another ion which had been upper case, so the ordering of the lower case indicators will change. If a rejected system leaves only one ion at that redshift, the tie code is set to blank since there is nothing to tie to.

If the Doppler parameters are tied in such a way that the turbulent component and the temperature are BOTH determined (see Section 11.3), and one of the base (lower case) variables is rejected the program chooses a corresponding ion with an upper case indicator and replaces it with a lower case one. There is no check that the two lower case indicators are not attached to different ionization levels of the same atom. If this happens the results are unpredictable since the determination of turbulent and thermal components requires two different atomic species.

Prior to version 11.1 the default was to attempt to remove all tied systems at once, but there was a bug which resulted in the wrong ones being removed unless the lower case ones appear first in the list.

12 Higher accuracy Voigt profile

A higher accuracy Voigt profile routine has been developed by Julian King at UNSW. It has the huge advantage over other methods that it is not significantly slower than the relatively inaccurate interpolation routine which it replaces, but the Voigt function error is now $\lesssim 10^{-5}$ (*J. A. King, Honours thesis, UNSW, 2006*).

Julian King has also implemented the Levenberg-Marquardt method to iterate towards a solution (see e.g. Burles & Tytler, ApJ, 499, 699, 1998 for an application of this method to an absorption spectrum, and Press et al. 1992, *Numerical Recipes*, Cambridge University Press for a description). This is an improvement over the Gauss-Newton method which has been used previously, in that VPFIT is less likely to wander around aimlessly in parameter space in difficult cases, and take fewer iterations to find a solution. However, when this method is used there are more function evaluations so there is a time penalty. The Julian King implementation avoids most of this by using the Gauss-Newton method when appropriate, and switching to Levenberg-Marquardt when needed. If you set the verbose level to 9 or above (see Section 7 for setup parameters such as this),

VPFIT tells you when it is using the different methods, and gives parameters associated with them.

At least that is the idea. In practice the switch to Levenberg-Marquardt seems to occur when that method also has trouble, and so further iterations result in little improvement. You may well get further by restarting VPFIT with the best fit results from that previous run and seeing if the χ^2 improves further - sometimes it does, sometimes it doesn't.

13 Reliability of parameter and error estimates

Generally the parameter and error estimates are reliable, given the model. A range of tests were done many years ago using simulated data to verify this, with the conclusion that the parameter estimates are unbiased and the estimated errors are representative of those for the parameter taken in isolation. The only exception was the Doppler parameter for damped profiles, which was not surprising since the temperature-independent Lorentzian wings dominate the profile, so the minimum of χ^2 with respect to the Doppler parameter is very wide and flat.

However, in particularly difficult cases the parameter error estimates, and maybe to a lesser extent the final parameters themselves, seem to depend on whether or not you are using optimized code compiled with the -ffast-math flag. The example below is extracted from a fit which involved 37 different systems (because of blends etc.) where the line is considerably narrower than the instrument profile of 6 or 7 km s^{-1} (depending on the spectrograph used - the fit was done using spectra from both HIRES and UVES). It is not at all clear what should be done about this, but as a warning here are some results from the same machine (a Dell Latitude D610) and the same startup file with the flags in the makefile for VPFIT set as indicated:

```
FFLAGS=  -ffast-math -Wall
 C I   1.7765227918ab  0.0000012937   0.51015bb    0.08041  13.119485    0.128159
 C I*  1.7765227918AB  0.0000000000   0.51015BB    0.00000  12.051059    0.071778

FFLAGS=  -O5 -Wall
 C I   1.7765237554ab  0.0000043223   0.52552bb    0.41552  13.162430    0.228997
 C I*  1.7765237554AB  0.0000000000   0.52552BB    0.00000  12.046898    0.085709
```

-O3 on the Dell, and the compiler on a Sun give the same results as the -O5 case. These are unlikely to be independent tests however!

80 simulations were run using the first version, and, at least for that version, the error estimates seem to be reasonably reliable. A smaller set of 14 simulations using -O5 on the Dell machine gives a mean Doppler parameter $\bar{b} = 0.515$ and a RMS of 0.078, so the error estimates from the fast-math set seem to be more reliable. The mean error from the -O5 simulations was 0.13, so where the 0.42 estimate came from in the fit to the real data is anybody's guess.

Another thing which was tried was to obtain error estimates for the same parameter estimates for the two optimizer options. They are different as well: 0.083 for -ffast-math, and 0.127 for -O5.

What should we conclude from this? Apart from reinforcing the general lesson of not

trusting anybody's computer code, including (and probably especially) one's own? Probably that much more testing is needed, but at least the error estimates provided by VPFIT range from being reasonable to fairly high overestimates for the case considered here.

14 Atomic data

14.1 File format

The atomic data file is usually called *atom.dat*, but can be anything. It is the file pointed to by the environment variable **ATOMDIR**. The format accepted by VPFIT is as shown between the horizontal lines.

Ion	λ (A)	f_{ik}	Γ	mass (amu)	comments
??	1215.6701	0.416400	6.265E8	1.00	No ID
>>	1215.6701	0.416400	6.265E8	1.00	
<>	1215.6701	0.416400	6.265E8	1.00	
--	1215.6701	0.416400	6.265E8	1.00	
H I	1215.6701	0.416400	6.265E8	1.00794	Morton(03)
H I	1025.7223	0.079120	1.897E8		
H I	972.5368	0.029000	8.127E7		
....					
PbII	1433.905	0.870	1e8	206	Kurucz ab=4
end	0000.000	0.000000	0.00000	0.000	Terminator

The atomic mass need be entered only where an element appears for the first time.

The H I could equally be HI. The assumed format for any ion is one or two characters describing the element, and up to four characters starting with a capital letter describing the degree of ionization. So, if you want some descriptor for molecular hydrogen, for example, then you might have

H2J=5 1017.836 0.02360 1.180E9

The four characters for the degree of ionization may sound limiting, but there have not been howls of protest so it has remained that way for decades.

The format could be a restriction if you are dealing with molecules, but you can use some other descriptor. The requirements are that the 2-character 'element' be a capital letter followed by anything which is NOT a capital letter, and the ionization level a capital letter followed by any three characters. The appearance of the first upper case letter in positions 2 or 3 of the character string is taken as the start of the ionization level. So e.g. `SiII` is interpreted as the element `Si` with ionization level `II`, but `SIII` is interpreted as element `S`, ionization `III`. `H2J=5` becomes 'element' `H2`, 'ion' `J=5`.

The wavelengths, oscillator strengths etc. are then strung out along the same line in the file in any convenient format to any convenient precision, with at least one space between each quantity to separate them. They are held internally as double precision variables. Comments can appear after the required information - these are ignored. If you look at the *atom.dat* file provided you will see what is allowed.

14.2 Special ions

In the example above the ion list started with some strange ones - ??, >>, <> etc. These are handled somewhat differently from the normal ions which appear further down in the list in this example, and all have been introduced to help handle either our own

ignorance (unidentified lines) or deficiencies in the data. We deal with each in turn.

14.2.1 Unidentified lines

The ion labelled '??' is simply a marker for an unidentified line. It can have any wavelength, oscillator strength, Γ-value and mass you choose, but in the default atomic data file it has the same values as for Lyα (apart from the mass, which is irrelevant). The only advantages this has over calling the ion 'H I' is that there are no associated higher order Lyman lines, and that it is easy to see where you have still to search for identifications. It can be useful to just create a file of the '??' lines and look for common line ratios in the hope of finding a further absorption system.

14.2.2 Region wavelength shifts

The >> tag is handled differently, and is used to check for, or allow for, wavelength shifts between spectral regions which might arise if the wavelength calibration has not been done properly. Obviously normally the best thing to do under these circumstances is to go back and redo the wavelength calibration more carefully, but this is not always possible.

If the line (fort.26 format)

```
>> 2.428120SA 0 1.25 0 1.00SC 0
```

appears in the startup (first guesses) file, then VPFIT applies a velocity shift to each region in which the redshifted 'line' falls (initially with a shift of 1.25 km s^{-1}) when fitting. The main use for this is when you suspect the wavelengths might be in error, and are fitting more than one region with lines from an ion, or system, spanning those regions.

For example, if you are worried about a shift between CIV 1548 and 1550, which might arise either because of poorly known rest CIV wavelengths or because of poor wavelength calibration, you might use the following:

```
%% datafile     1   4174.3265   4176.5921
%% datafile     1   4181.2306   4183.6780
 C IV     1.696697    0.000011     6.70      3.60  11.999    0.461  0 !
 C IV     1.696836    0.000039    15.44      6.34  12.486    0.183  0 !
 C IV     1.697105    0.000000     8.28      0.05  13.838    0.002  0 !
 >>       2.440478SZ  0.000000    -0.49      0.03   1.000SH  0.000  0 !
```

In this case the reference region is CIV 1548, and the region containing CIV 1550 also contains the >> 'line', so is shifted by -0.49 km s^{-1} with respect to the reference region to start. The value will change at each iteration for an overall best fit.

In this case the shift could be from poor CIV rest wavelengths. The estimate here of -0.5 km s^{-1} was from data on HE0515$-$4414 using the Morton wavelengths (ApJS, 149, 205, 2003), while Petitjean & Aracil (A&A, 422, 523, 2004) suggest a double separation of ~ 0.7 km s^{-1} less then the Morton value. Revised wavelengths have now been included

in the `atom.dat` file, so if you are using this one don't be surprised if the results are not compatible with any previous values.

You can also use the `>>` flag to allow for possible wavelength shifts between two datasets for the same line by adding a non-zero integer to the parameter line to tell the program which file you may want to apply a correction to. For example if you were just looking at the CIV 1550 line in the dataset above, but had two files 1 & 2 and were suspicious about the wavelength calibration for the second, you could use

```
%% datafile1     1   4181.2306   4183.6780
%% datafile2     1   4181.2306   4183.6780
 C IV     1.696697    0.000011     6.70      3.60  11.999    0.461  0 !
 C IV     1.696836    0.000039    15.44      6.34  12.486    0.183  0 !
 C IV     1.697105    0.000000     8.28      0.05  13.838    0.002  0 !
 >>       2.440478SZ  0.000000     0.00      0.00   1.000SH  0.000  2 !
```

The program then compute the best fitting shift in wavelength to make the CIV line in datafile2 to agree with datafile1. The '2' in the last column for the `>>` component tells the program to apply that 'line' only to the second file+region in the region list.

Note that if you are using this mode you should not apply a wavelength shift to all regions - the values are then unconstrained!

14.2.3 Continuum adjustment

Precise continuum estimates can be difficult in crowded regions e.g.the Lyα forest in QSOs, so it is quite likely that the continuum estimates there are not correct. You can adjust the continuum level using a linear multiplicative factor. It is based on false elements as tags, and the line in the fort.13 format file looks like:

```
<> clevel redshiftN cslope(M) 0.0 0.00E+00 (region number).
```

or for fort.26 format:

```
 <>  redshiftN 0.000000  cslope(M) 0.00   clevel   0.000  (region \#) !
```

The value of `clevel` is usually near unity [it is what the continuum is multiplied by at the wavelength corresponding to $\lambda_{\rm ref} = (1 + \texttt{redshift}) \times 1215.6701$, which must be within the relevant wavelength chunk], and the redshift MUST be fixed. The `cslope` value may be fixed (usually at zero, for a constant rescaling of the continuum) or may be left to be determined by the program. The input continuum through the fitting region at observed wavelength λ is multiplied by $(\texttt{clevel} + \texttt{cslope} \times (\lambda/\lambda_{\rm ref} - 1))$, so for small adjustments `level` is somewhere vaguely near unity. You can input this interactively as well. In cursor mode it asks if you want to fit the slope or not, and you fix it (at zero) by pressing the left mouse button or the central one, allow it to vary by pressing on the right mouse button. Be warned – it is all too easy to have a continuum change trade off against a broad line, so it is probably best to use this only for (small) final adjustments to the parameters. If these final adjustments are not small, worry about the reliability of your results.

As an example using the alternative input (fort.26) format

```
%% datafile     1   4174.3265   4176.5921
%% datafile     1   4181.2306   4183.6780
 C IV     1.696697    0.000011     6.70      3.60  11.999    0.461  0 !
 C IV     1.696836    0.000039    15.44      6.34  12.486    0.183  0 !
 C IV     1.697105    0.000000     8.28      0.05  13.838    0.002  0 !
 <>       2.440478SZ  0.000000     0.00      0.00   1.000    0.000  0 !
```

In this case a correction to the input continuum is applied to the region(s) containing the wavelength $(1 + z) \times 1215.6701 = 4182.486$ A, so the one containing the CIV 1550 lines.

The `region number` parameter is rarely used. If it is zero it is ignored, but for a non-zero value the continuum adjustment is applied only to the region corresponding to that number, independent of effective wavelength. In the example above if the final zero were 2 in the <> line it would have the same effect. Note that you still need a reference redshift for the linear term in the continuum adjustment. It can be used where there are overlapping regions at e.g. different resolutions used in the fit, in cases where the continuum adjustment would otherwise be applied to more than one fitting region.

14.2.4 Zero level adjustment

It is not at all uncommon for there to be some residual flux in the bottoms of what appear to be completely saturated lines, and in such cases the automatic line addition feature (see section 6.5) just stacks in lots of spurious components to try and make the overall profile.

If you use the 'line' __ (1215.67 assumed wavelength, to make life easy (?), but you can change it in the atom.dat file), then the continuum is left where it was, and the fitted spectrum stretched linearly to the new zero level. Note that the first parameter `zero-level` is now expressed as a fraction of the continuum, rather than put on a scale where it is forced to look like a column density, so the line in fort.13 now looks like:

```
__ zero-level redshiftN b-value 0.00 0.00E+00 n
```

where the zero level adjustment will normally be around zero (it may be negative), and will be less than 1! As before, N is a sample letter to fix the redshift, which should be such that 1215.67*(1+z) is within the region of interest. The number n is the dataset number for which the zero level is to be adjusted (1 for first region, 2 for second, etc.), and this is not needed if there is only one region which contains the wavelength 1215.67×(1+`redshift`) so is usually set to zero (= ignore it). It performs a linear fit of the form $\texttt{zero} - \texttt{level} + \texttt{b} - \texttt{value} * \Delta\lambda/1215.6701 \times (1 + \texttt{redshift})$, and you can fix either of the other two values (indeed, it usually is best to set the `b-value` to zero and fix it, but occasionally you might want a slope).

Note that the zero level adjustment applies to the input spectrum received at that point, so all the real ions with lines in that region should appear higher in the list than the '__' one. Continuum adjustment matters less, but it is probably best to adopt a policy of putting the '__' last just for internal consistency.

14.3 Extra parameters

For those interested in fine structure constant or electron/proton mass changes with redshift a further column has been added, as an optional extra. For these the format is

Ion	λ (A)	f_{ik}	Γ	mass	q	comments
#q						
??	1215.6701	0.416400	6.265E8	1.00		No ID
>>	1215.6701	0.416400	6.265E8	1.00		
<>	1215.6701	0.416400	6.265E8	1.00		
__	1215.6701	0.416400	6.265E8	1.00		
H I	1215.6701	0.416400	6.265E8	1.00794		Morton(03)
H I	1025.7223	0.079120	1.897E8			
H I	972.5368	0.029000	8.127E7			
....						
FeII	2600.1729	0.2394	2.70e8	55.847	1356	M03g
FeII	2586.6496	0.069125	2.72E8	55.847	1520	M03g
FeII	2382.7652	0.320	3.13E8	55.847	1498	M03g
FeII	2374.4612	0.0313	3.09E8	55.847	1640	M03g
....						
PbII	1433.905	0.870	1e8	206		Kurucz ab=4
end	0000.000	0.000000	0.00000	0.000		Terminator

Here the effective q-value is in cm^{-1} (as defined by Murphy et al., MNRAS, 345, 609, 2003). Note that the atomic mass is now needed for every line where there is a non-zero q-value, but if it is zero (or some character string which would be interpreted as zero) then it will fill in the mass from a previous value for the same atom. With this format those who are not interested in α or μ don't have to change their **atom.dat** files, or even know the capability exists. As before, you can string numbers out along the line with any convenient spacing (don't use tabs though), and add any comments [anything after column 7, non-numeric before].

A warning is given if an attempt is made to use an extended atomic data file of this form when there is not enough space allocated for the number of variables needed. It happens early on when VPFIT is reading in the atomic data, so look out for it! There are two possible causes:

1. The setup file (see section 7) does not contain

   ```
   NOVARS 4
   ```

   which tells it there are four variables per line in the list, not the usual three.

2. Not enough space was allowed for the fourth variable when the program was compiled. To remedy this in the file *vp_sizes.f* set

   ```
   * MAXimum Parameters Per System
         integer maxpps
         parameter ( maxpps = 4 )
   ```

if the value had previously been 3.

If flagged by the letter 'q', or 'Q' for tied values (using the usual two-letter sequence, with the second letter changing for different tied sets), the fourth parameter is taken as the $\Delta\alpha/\alpha$ value (internally scaled up by a factor of 10^6) which applies to that transition. Normally the same value for this quantity should apply to all ions at a given redshift. To achieve this, tie the appropriate values in the usual way. Where there is no potential line shift due to possible changes in $\Delta\alpha/\alpha$, the values should be fixed at zero.

The format of the output for $\Delta\alpha/\alpha$ can be changed. If you want values in units of 10^{-5}, for example, then in the VPFSETUP file use

```
daoaunit 1E-5
```

The default is actual $\Delta\alpha/\alpha$ values, i.e. daoaunit 1.

For molecular hydrogen the wavelength shifts are defined slightly differently, with

$$\lambda_i = \lambda_i^0 \left(1 + K_i \frac{\Delta\mu}{\mu}\right)$$

In the atomic data file for the 'element' H2 the extra parameter for each transition is just the K_i value, and the fitted $\Delta\mu/\mu$ values are subject to the same display (and internal) scalings as $\Delta\alpha/\alpha$ above. The tied flag for H_2 shifts is `m` or `M` replacing the `q` or `Q` above.

If you want the region by region flux statistics for computing a lower limit to the error in $\Delta\alpha/\alpha$, then in the VPFSETUP file put

```
WR26FS
```

and they will be written to the fort.26 output summary file.

14.4 Isotopes etc.

Isotope descriptors can be used in the atom.dat file, though the nomenclature is not standard (instead of 24Mg you enter Mg24 - this is simply so we did not have to change the element/ion separation method in the program). If you want e.g. MgI isotopes then add into the atomic data file something like:

```
Mg24I 2852.963420 1.830 5.000e8 24.0 86.0
Mg25I 2852.961407 1.830 5.000e8 25.0 86.0
Mg26I 2852.959591 1.830 5.000e8 26.0 86.0
```

The element may be up to 4 characters, and the ionization level is separated by using the first capital letter after the first character [HD and CO are recognized as special cases, but they are the only exceptions]. The use of four character element descriptors may mess up the output formatting a bit, since the program originally assumed up to two, but, apart from appearance and alignment, hopefully there should be no other problems.

15 Fitting simulated data

15.1 Single wavelength region fits

There is nothing different required for fitting simulated spectra. You can go through as for a real spectrum, and obtain fit parameters. However, you probably want to do this many times, for similar wavelength regions with similar line identifications, and don't want to repeat the same mindless set of operations for each of them. At least for simple cases there are some ways of easing the burden.
The first thing to do is set up a file containing a list of all the spectra you want to fit, giving the fit regions. Here we'll call it `input.13` and it might look like:

```
%% spectrum1 1 3535.23   3540.09

%% spectrum2 1 3535.23   3540.09

%% spectrum3 1 3535.23   3540.09

%% spectrum4 1 3535.23   3540.09

.....
```

where there is a blank line between each of them in this example. You could put in first guesses to replace the blank lines if you wish, e.g. for OVI 1032:

```
O VI     2.42812    0    11.0      0  13.0    0.0
```

in all cases, but that is not necessary. You can specify which line to use instead in the setup file (pointed to by the environment variable VPFSETUP, see 7). If the blank lines are left in, the setup file should contain

```
guessline OVI 1031    Line guesses are given feature (default is Ly-a)
```

Since there are no initial estimates for the Voigt profile parameters, VPFIT then guesses some for itself based on a crude separation into distinct components, assuming that all are identified as the line specified by the `guessline` setup parameter. The default if `guessline` is not specified is Lyα.
You should also ensure that the output control setup parameter

```
! ----  output control
! wr26s f26.w            fort.26 summary root filename
```

are commented out, as here, or removed from the setup file. This ensures that the output summary is written sequentially to a single file - fort.26 whether you like it or not - and that successive fits don't overwrite the previous ones! It will produce a giant results file fort.26, which you **absolutely must** copy somewhere before restarting the program else the results will be overwritten. I have tended to edit the input file to do things in manageable chunks (of say 100 spectra at a time), and then append this to an accumulated results file before restarting.
Then, when you run VPFIT the initial sequence will look like, where the > at the beginning of a line shows where the user provides input:

```
....

 Default line is O VI    1031.9261

....

  options:   <CR> for previous value
  I - interactive setup and fit
  F - run from an input file
  D - display profiles from input file
  ? for help
  option (key) (key)...
> f ad ec 14.5 1.0

 Add lines if prob(tot) < 0.010000   or prob(KS) < 0.00000001

 Try removing lines at end if logN <  14.50 and err(logN) >   1.00

 Column density (n), logN (l) or emission (e), scalefactor [l, 1.0]
>
 Parameter input file, # entries? [fort.13,1]
> input.13,999

 filename       : spectrum1
 spectrum number:    1
 ASCII file input
 Resolution  6.7 km/s
 260 data values
 rms set to error values
 Start & end chans:  60 203
 1 regions fitted
 1 features found
 O VI    13.0310514  2.42811519  13.4834161
  no. of ions for fitting is  1



   ion          N            z             b         bturb   temp

 iteration   :   0 ( 1 )
 chi-squared :          1.243 (       175.2894,   141 )

   O VI     13.03105     2.4281152     13.4834        0.00   0.00E+00  0 !    1

.....
```

You need to tell the program that there is more than one dataset, and the

```
> input.13,999
```

will make sure up to 999 fit sets are read (if you have even more, make the number larger!). It will ask you if you want to look at the last one in the list only if the number

given matches the number of datasets. You can use option 9 in the program to check that you like the look of the results it obtains if you are worried.
The fort.26 file will contain something like:

```
%% spectrum1        1   3535.2300   3540.0900    !
! Stats:   3     1.2011177  144  141  0.052  0
 0 VI     2.428119    0.000008    10.88      1.03  13.001    0.031  0 !
%% spectrum2        1   3535.2300   3540.0900    !
! Stats: ....
```

where the 'Stats' line gives, in order, number of iterations, noramlized χ^2, data pixels covered, degrees of freedom for fit, and the approximate probability that the fit describes the data.

There will be some regions where the fitting process fails. These are easily identified by looking through the fort.26 file to see where the final fit probability is too small. You'll need to see if you can provide better initial guesses for these ones by hand.

The input file containing the filelist and regions with blank lines can be a slight pain to set up initially, but once you have one then if you set up filenames in sequence with different names and regions then global edits can be used to provide such files for different simulations.

15.1.1 Single file multiple simulations

You'll need something to convert whatever data format you have to a FITS file, with many spectra covering the wavelength same region. The FITS file now contains a 2-D dataset, with each spectrum a single x-slice. There is a program `bintofits` included in case you have something in the right input format (it was written for Matteo Viel), so you can see what might be involved. The following instructions apply for that program, and for single zone Lyα only simulations, so some things are quite specific:

Assuming that you have blocks of simulations covering the same wavelength range, you can run a preprocessing program to convert to FITS data and error files and then feed these into a multi-run VPFIT fitting procedure. As things stand:

- You need a simulation at a given redshift with a velocity range covering whatever Matteo said it was, as a binary data file of spectra with the maxima at the ends of the range, convolved with the instrument profile at UVES/HIRES resolution, with noise added, and a signal vs error lookup table in ASCII. If an error file is provided in the same format as the data file, just run the conversion program bintofits on the error file as well to convert it to a fits file, but make sure you don't overwrite the fit guess file which ALWAYS goes to a file called fort.13 - so you might do the error file first. If you do, then if the datafile is to become <filename>.fits, then the error file should be <filename>.sig.fits if you don't want to edit the header in the data fits file.

- Run the program bintofits to convert from binary to fits form. This program generates a datafile with an extended spectrum so that the convolution with the instrument profile in vpfit does not give funny results, also an error file and a fit

guess file for vpfit. The data file has the required information on wavelengths, errors, resolution etc in the header, so no extra header editing should be needed. It asks you for the binary filename, the output filename (with a suggested default), and the ASCII signal vs error table filename(which may not be needed). The fit guess file should look like (for redshift $z = 3$)

```
%% filename 1 4862.680 4888.040

%% filename 2 4862.680 4888.040

%% filename 3 4862.680 4888.040

%% filename 4 4862.680 4888.040

.....
```

For $z = 2.75$ the wavelength limits should be 4558.76 and 4582.60, though since they are just ends of fit ranges they don't have to be too precise. If they are not what you expect, then do a global edit on the fort.13 file.

- `setenv VPFSETUP vp_setup.dat` where vp_setup.dat might contain the following, and in particular remove any reference to `wr26s` in that file, otherwise it will continually overwrite the results of one spectrum on top of the results of the previous one, and you'll lose the information. You should also set

  ```
  maxadrem 50
  ```

  or something similar, since in some cases it requires more than the default number of add/remove line cycles (25) to converge. I also commented out the add continuum parameters option so that continuum adjustment do not occur.

  ```
  !  adcontf 2.2 ...
  ```

  You might also add

  ```
  nopchan 0
  ```

  to suppress lots of the intermediate output. Then you won't get output to fort.18 (the full file giving each iteration result) and the fort.15 file (last iteration). The fort.26 summary file is still written at the end of each spectrum fit and that is the sole location for the results in this mode, and enough appears on the screen for you to see if the program is getting nowhere. If you want to see what is happening on the screen, use nopchan 1, if you want it on the screen and to fort.18 use nopchan 2, and if you want everything use nopchan 3.

- Run vpfit using as responses to the queries:

  ```
  f ad 0.0025 0.0005 ec 14.5 2.0
  <CR>
  <input file (e.g. fort.13)>  1000  -- or however many you want in
                                        this block, starting
                                        with the first.
  ```

If you run a very old version it will not recognize the wavelengths, which are set up as header variables in the form

```
GLWBASE  3.686535639476      Global log wavelength base
GLWINCR  2.267008475368D-06  Global log wavelength increment
```

- This will produce a giant results file fort.26, which you absolutely must copy somewhere before restarting the program else the results will be overwritten. I have tended to edit the input file to do things in manageable chunks (of say 100 spectra at a time), and then append this to an accumulated results file before restarting.

Assorted notes:

If you want to stop a series of fits, then

`touch stop_series`

in the directory you are operating from will terminate the program cleanly. touch stopit will stop the iterative fit (adding and subtracting lines) for the current spectrum at a convenient point - the program should then go on to the next one. touch stop stops the current parameter search as if it had converged to a chi-squared minimum.

Note that, as with data from the telescope, the vpfit program will fail to converge to an acceptable answer in about 5% of the cases (the precise number depends on the column density distribution function and the line density), and so some hand restarting is necessary for these. If you

`grep ' 0.000 ' resultsfile | wc`

or, if the default setup is used, you should find that the summary file has bad fits labelled with the word 'BAD', and so

`grep 'BAD' resultsfile | wc`

you can see how often these occur. You can refit those regions and modify the accumulated results file as appropriate. A fairly common reason for failing to converge is that there is a saturated line which it is having trouble sorting out. I half expect an excess of lines at around logN=17, since when fitting Ly-alpha only where it is on the flat part of the curve of growth noise fluctuations can drive the fits to either end of the flat part. You need more lines in the Lyman series to overcome this problem.

Convergence and run time:

Its a bit slow, and takes about 3 CPU mins per spectrum on a 2.4GHz laptop. With the suppressed output there is a gain of about 30% in speed, so we wind up with a little over two days per 1000 spectra. You can relax the convergence criteria at each iteration. In the `vp_setup.dat` file the line

`chisqthres 0.001 4.0 0.01`

results in the iterations terminating if the relative χ^2 $[(\chi^2(new) - \chi^2(old))/\chi^2(new)]$ changes by less than 0.01 if the normalized χ^2 is above 4.0, and less than 0.001 otherwise. Those numbers are the defaults. I would not increase the last much (I tried, and the program gets a bit lost), but you might try 0.001 2.0 0.01 to speed things up a bit.

15.2 Multiple wavelength regions

15.2.1 Single ions

The spectral file list with fitting regions method described above in Section 15.1 can be extended to cover more than one region if a single ion is involved. If the list looks like:

```
%% spectrum1 1 3554.73   3559.61
%% spectrum1 1 3535.23   3540.09

%% spectrum2 1 3554.73   3559.61
%% spectrum2 1 3535.23   3540.09

.....
```

then the behavior is as above, but both regions, and so both lines of the OVI double in this example, are fitted. The initail guesses are based on a search of the last region in the list before the blank line ONLY, but the fits include all regions.

15.2.2 Two ions

If you have reason to believe that some ions are linked, e.g. HeII and HI in the example here, then you can take the analysis further, for either simulations or real data. This involves more setup file parameters:

```
xlink HI  0.0 HeII 2.0
turbulent
nofix
```

The interpretation of these lines is:

1. Crosslink HI and HeII, such that if the line adding routine within VPFIT adds HI, for example, then HeII will be added at the same redshift with a column density such that $\log N(\mathrm{HeII}) = \log N(\mathrm{HI}) + 2.0 - 0.0$.

2. The HeII and HI Doppler parameters are linked through turbulence i.e. there is no atomic mass dependence. If this is absent then thermal linkage is assumed.

3. If a base line is rejected in subsequent iterations, then the associated line (denoted by upper case letters aginst the redshift) is also removed. So no systems are left as having fixed redshifts.

Then for the default line ID (Lyα) the input.13 file can then look like:

```
%% flux_He2    1    1032.850    1037.920
%% flux_H1   1   4133.2800   4153.5200

%% ...
```

In principle this could be extended to more than two ions, but this has not been tried...

16 RDGEN with VPFIT: their use together

The interactive mode of VPFIT has some low level use, once you know where the lines are and have at least suggested identifications, but it does not help much with that essential first step. If you are presented with a spectrum and know little about it, then you have to try to identify lines, and then, if you want to fit them, give VPFIT some start-up guesses. There are lots of ways of doing this, with varying degrees of tedium and trial-and-error. In this section we illustrate how RDGEN and VPFIT can be used together to set up the fits, fit the profiles, and examine the result.

We have used the Keck HIRES spectrum of Q2206 − 19, which is one of those made available by Jason X Prochaska & Arthur M Wolfe. It is available from

http://www.ucolick.org/ xavier/DLA/Spectra/HighRes/index.html. There are two files, the data in Q2206_f.fits & Q2206_e.fits. Amost inevitably, the default naming convention for VPFIT files is different from theirs, so to make life easier later, you should change the error file name to Q2206_f.sig.fits (e.g. by `mv Q2206_e.fits Q2206_f.sig.fits`). The .sig extension to the data file name (minus the .fits) is automatically picked up as the error file by VPFIT.

If you are using the standard setup files, and have not reset the environment variables from the standard ones, then

```
$ rdgen
------ RDGEN 10.0 ------
```

and a fair amount of startup information .....

```
>>pf hiion.pg
>>pf loion.pg
....
>>pf CIV.pg
....
>>fq
Data files in this directory are:
1 Q2206_f.fits
2 Q2206_f.sig.fits
>>
```

Now the program has finished loading whatever was set up for it (including not finding any .fits files in this case), you first need to read the data, so at the `>>` prompt enter

```
rd Q2206_f
```

(you don't need the .fits, but if you do put in in it wll still read it). You should get the question `full continuum name?`, to which a `<CR>` will do - it then assumes unit continuum. There is also a cryptic response `Using rescaled error array for rms` which you can ignore.

Then you might as well look at it, plotting with the cursor:

```
>>pg
plot parameter?  (type he for options list)
>
PGPLOT device?  [/xw]
>
```

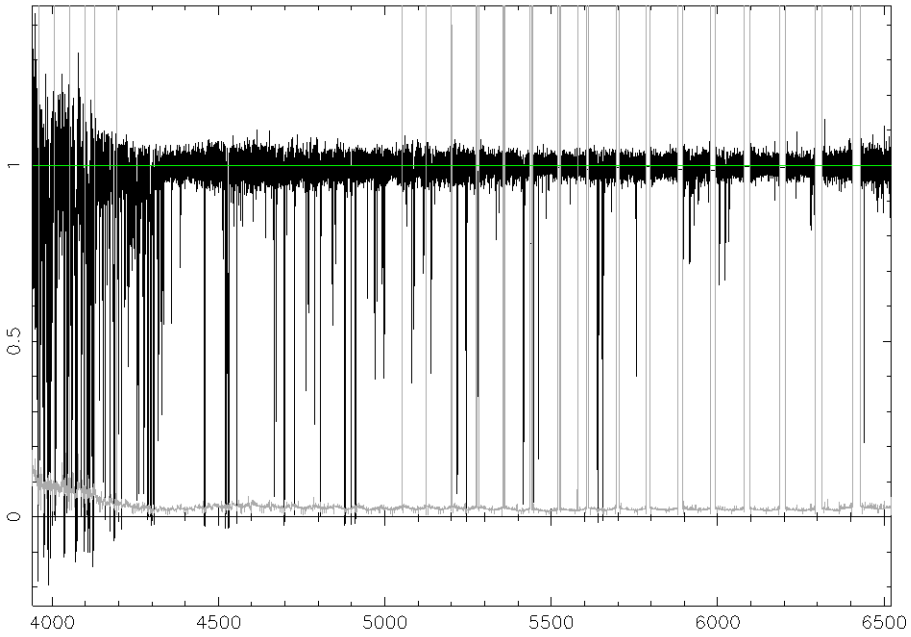

Figure 4: The Keck HIRES spectrum of Q2206 − 19. The grey curve is the 1-σ error estimate - bad pixels are shown by the error going off-scale.

```
Expand plot if needed:
Cursor ("e" to mark edges, "q" when OK)
```

will do that, and with `<CR>` at each `>` you should have a plot on the screen. If you have the defaults set, it should look like Fig 4. If it does not, then type `a` (for 'all the spectrum') **in the graphics window**, which is where control has been transferred to. Here the green line is an estimate of the (unit) continuum, and the grey one is the 1σ error estimate.

You learn nothing from the whole spectrum, so to select a part which may be of interest. For example, the strong FeII 2600 absorption at $\sim$ 5245A by putting the cursor on 5200A (the y-position does not matter), clicking the left mouse button, moving it to 5300 and clicking again. You can expand this even further by doing the same then at e.g. 5240 & 5250A. You should then see something which looks like Fig 5.

A single line like that is not desperately informative. It would be nice, perhaps, to see how it relates to heavy element lines. It would also be vaguely useful to get rid of the uninformative error array from the plot at this stage, so to do this type `C` in the plot window. Now to see other lines in the same redshift system as this lyα, set the cursor in the middle somewhere (like 5245A), and **still in the graphics window** type `v` (for velocity plot). A list[1] then appears in what had been the command window,

[1]The VPFIT source comes with a subdirectory 'pgfiles', which contains lists of lines one might want to see put on a common velocity scale. The filenames are mostly indicative of what is in them e.g. `hiion.pg` includes a few Lyman series, plus CIV, NV, OVI, SiIV; `loion.pg` Lyman lines plus CII, OI, SiII etc; `lowz` MgII, FeII etc.

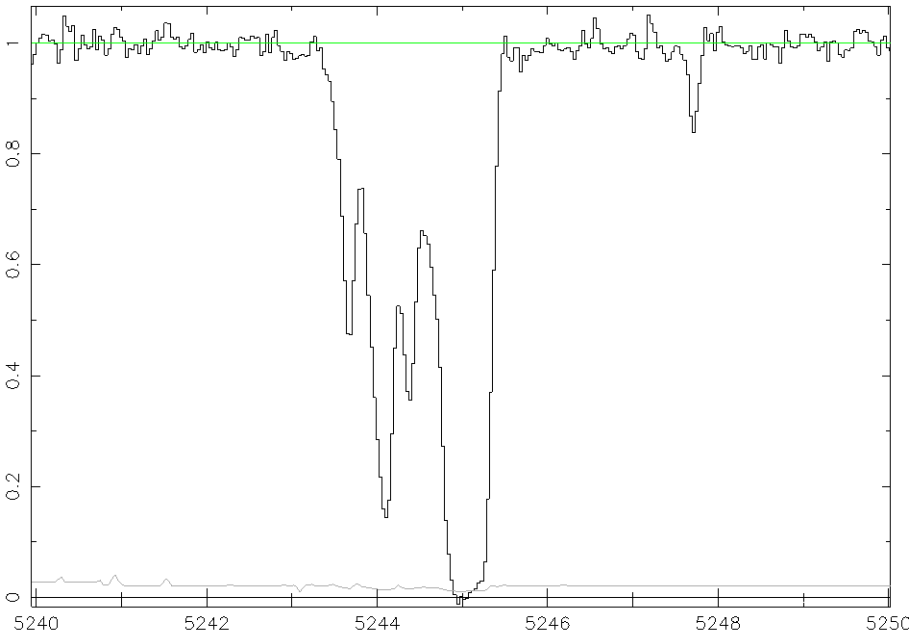

Figure 5: A section of the Keck HIRES spectrum of Q2206-19.

```
File ID?
1   hiion.pg
2   loion.pg
3   lohi.pg
4   lyabg.pg
5   lyman.pg
6   lowz.pg
7   CIV.pg
.....
n   FeII.pg
o   lointhi.pg
p   sdlaion.pg
..  in plot window
```

Try - still in the plot window - typing `6`. Then in the plot window you'll get a list of lines, starting from the bottom with `NaI 5891` (which is outside the data range), and including lines from MgI, MgII, ALII, AlIII, SiII, CaII and FeII. Put the cursor on `FeII 2600.17` (the y-position within or just above that string, the x-position does not matter). You then see a velocity scale plot similar to Fig 6 (with different colours to stand out better against a black background).

So far we've not done much, but we have now reached the stage where we can set some regions and initial parameters for us by VPFIT. Suppose you want to fit FeII at this redshift. From the FeII 2586 line it looks as if there are something like seven components, and 2367 is so weak as to be unmeasurable. First we might as well write out the fit

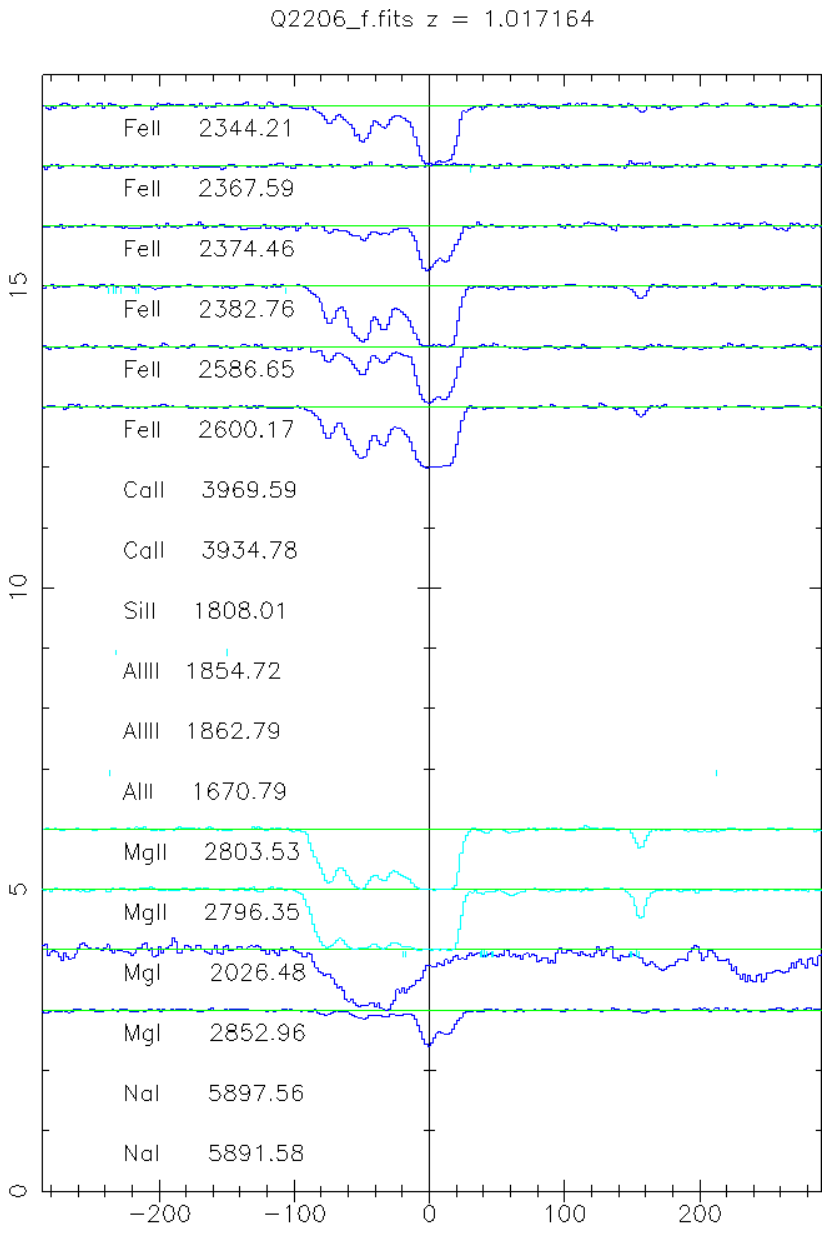

Figure 6: A velocity plot showing singly ionized species (mostly) relative to a reference redshift of $z = 1.01717$.

regions for the lines. It does not matter in which order, so with the y-cursor between the 0 and 1 levels for FeII 2344, set the x-position slightly beyond where the continuum to the left of the line is reached - say at about -120 km s^{-1} relative to the strongest component - and (still in the plot window) type `b` (for boundary). Move the x-cursor to a about 60 to the right if the 1304 line and do it again. Something like

```
%% Q2206_f.fits    1   4726.8008   4729.6006
```

will appear in the text window. Do the same for the other FeII lines, including 2367 if you feel like it. The line is not there, but it does provide an upper limit to the FeII column density. You will then have something like:

```
%% Q2206_f.fits    1   4726.8008   4729.6006
%% Q2206_f.fits    1   4773.8838   4776.7646
%% Q2206_f.fits    1   4787.7378   4790.6665
%% Q2206_f.fits    1   4804.4941   4807.3672
%% Q2206_f.fits    1   5215.5986   5218.7466
%% Q2206_f.fits    1   5242.8657   5246.0151
```

Now we need some initial line guesses. In the FeII 2586 section put the x-cursor where you think a velocity component is, and type `l` (for line). Do this as many times as you think there are components - I guessed four, so wound up with (ion, redshift, 0, Doppler b, 0,log column, 0) sets:

```
FeII    1.016660  0.000000     5.74    0.00 12.995  0.000
FeII    1.016774  0.000000     7.55    0.00 13.253  0.000
FeII    1.016823  0.000000    11.36    0.00 13.684  0.000
FeII    1.016921  0.000000     4.00    0.00 12.963  0.000
FeII    1.017008  0.000000     3.00    0.00 12.473  0.000
FeII    1.017155  0.000000    10.69    0.00 14.351  0.000
FeII    1.017248  0.000000     9.78    0.00 14.220  0.000
```

So you should now have something you can copy and paste directly into a VPFIT input file (sometimes simply called fort.13), which, after adding something to tell the program the instrument resolution for each region, looks like

```
%% Q2206_f.fits    1   4726.8008   4729.6006 vfwhm=6.5
%% Q2206_f.fits    1   4773.8838   4776.7646 vfwhm=6.5
%% Q2206_f.fits    1   4787.7378   4790.6665 vfwhm=6.5
%% Q2206_f.fits    1   4804.4941   4807.3672 vfwhm=6.5
%% Q2206_f.fits    1   5215.5986   5218.7466 vfwhm=6.5
%% Q2206_f.fits    1   5242.8657   5246.0151 vfwhm=6.5
FeII    1.016660  0.000000     5.74    0.00 12.995  0.000
FeII    1.016774  0.000000     7.55    0.00 13.253  0.000
FeII    1.016823  0.000000    11.36    0.00 13.684  0.000
FeII    1.016921  0.000000     4.00    0.00 12.963  0.000
FeII    1.017008  0.000000     3.00    0.00 12.473  0.000
FeII    1.017155  0.000000    10.69    0.00 14.351  0.000
FeII    1.017248  0.000000     9.78    0.00 14.220  0.000
```

Now try running that file as a VPFIT starter.

```
$ vpfit
```

```
....
options:  <CR> for previous value
I - interactive setup and fit
F - run from an input file
D - display profiles from input file
?  for help
option (key) (key)...
> f ad
```

Where the `f ad` is what you type. `f` to say read from a file, and the `ad` bit to get the program to add lines if the fit is not good enough. Then

```
Column density (n), logN (l) or emission (e), scalefactor [l, 1.0]
>
Parameter input file, # entries?  [fort.13,1]
```

> or if not fort.13, your input filename from above

VPFIT then gives a commentary on how its getting on, and adds a small continuum adjustment, to end up with

```
iteration   :    14 ( 2 )
chi-squared :                    1.121 (       581.7852,   519 )
FeII     12.76844     1.0166652      6.2749        0.00   0.00E+00  0 !        1
FeII     13.10892     1.0168380      4.1470        0.00   0.00E+00  0 !        2
FeII     12.80161     1.0167800      4.9478        0.00   0.00E+00  0 !        3
FeII     12.88136     1.0169360      5.1196        0.00   0.00E+00  0 !        4
FeII     12.52347     1.0170344      5.4631        0.00   0.00E+00  0 !        5
FeII     14.00015     1.0171633      7.3806        0.00   0.00E+00  0 !        6
FeII     13.65137     1.0172617      4.6205        0.00   0.00E+00  0 !        7
<>        0.99153     3.2916027SZ   57.6233        0.00   0.00E+00  0 !        8
Rescaled parameter errors:
FeII      0.01117     0.0000011      0.2321        0.00   0.00E+00    !        1
FeII      0.03147     0.0000022      0.3774        0.00   0.00E+00    !        2
FeII      0.08432     0.0000051      0.7273        0.00   0.00E+00    !        3
FeII      0.01650     0.0000010      0.3742        0.00   0.00E+00    !        4
FeII      0.05816     0.0000027      0.9796        0.00   0.00E+00    !        5
FeII      0.01128     0.0000011      0.2373        0.00   0.00E+00    !        6
FeII      0.01561     0.0000013      0.1486        0.00   0.00E+00    !        7
<>        0.00345     0.0000000SZ   17.1141        0.00   0.00E+00    !        8
statistics for whole fit:
Runs test  K-S test Chi-squared  Chans ndf   APr   Xp(.68) Xp(.95) Xp(.99)
0.00000   0.00474    581.79     542  519   0.029  533.75  572.83  596.25
Statistics for each region :
Start      End    Chi-squared  Chans df?
4726.51   4729.92    104.98      89   68   0.003 < Prob < 0.119 g= 0.003   1
4726.51   maxdev       1.0337             0.236   0.956   1.358   1.627
4773.57   4777.07     52.08      91   70   0.946 < Prob < 1.000 g= 0.946   2
4773.57   maxdev       0.8445             0.474   0.956   1.358   1.627
4787.44   4790.99     86.21      92   71   0.106 < Prob < 0.651 g= 0.106   3
4787.44   maxdev       0.6922             0.724   0.956   1.358   1.627
4804.20   4807.67    103.59      89   68   0.004 < Prob < 0.138 g= 0.004   4
```

```
4804.20   maxdev       1.3182              0.062   0.956   1.358   1.627
5215.26   5219.08    109.81      91   68   0.001 < Prob < 0.087 g= 0.001   5
5215.26   maxdev       1.0405              0.229   0.956   1.358   1.627
5242.53   5246.35    125.11      90   69   0.000 < Prob < 0.009 g= 0.000   6
5242.53   maxdev       1.6796              0.007   0.956   1.358   1.627
Plot?  y,n, c=change device, t=change ticks [y]
>
```

You can plot each region idividually to see how good the fit looks, and then exit by

```
Plot?  y,n, c=change device, t=change ticks [y]
> n
Summary output was to f26.9p5w4728
Fit more lines?  [n]
> n
```

Have a look at the summary output, f26.9p5w4728, which is an abbreviated form of the fit results with everything needed for a restart if you want to. It should contain something like:

```
%% Q2206_f.fits     1   4726.8008   4729.6006 vfwhm=6.5  !    0.119    89 2010/10/19
%% Q2206_f.fits     1   4773.8838   4776.7646 vfwhm=6.5  !    1.000    91   2
%% Q2206_f.fits     1   4787.7378   4790.6665 vfwhm=6.5  !    0.651    92   3
%% Q2206_f.fits     1   4804.4941   4807.3672 vfwhm=6.5  !    0.138    89   4
%% Q2206_f.fits     1   5215.5986   5218.7466 vfwhm=6.5  !    0.087    91   5
%% Q2206_f.fits     1   5242.8657   5246.0151 vfwhm=6.5  !    0.009    90   6
!  Stats:    14     1.1209734  542  519  0.029  0
FeII     1.0166651672    0.0000010938     6.27487      0.23210  12.768436    0.011167  0 !
FeII     1.0168379718    0.0000022476     4.14700      0.37738  13.108916    0.031467  0 !
FeII     1.0167800205    0.0000050995     4.94784      0.72731  12.801610    0.084319  0 !
FeII     1.0169360041    0.0000009907     5.11960      0.37417  12.881360    0.016497  0 !
FeII     1.0170344150    0.0000027054     5.46309      0.97965  12.523467    0.058155  0 !
FeII     1.0171632821    0.0000010815     7.38065      0.23725  14.000153    0.011283  0 !
FeII     1.0172616675    0.0000012609     4.62046      0.14859  13.651367    0.015614  0 !
<>       3.2916026553SZ  0.0000000000    57.62327     17.11410   0.991529    0.003448  0 !
```

If the FeII column density, velocity structure etc. is what you want to know, that is it. However, the velocity plot (Fig 6) also showed MgII with similar structure. If you go through the same procedure for that you might wind up with a solution (here in f26.9p5w5640) along the lines of:

```
%% Q2206_f.fits     1   5637.7109   5642.3652 vfwhm=6.5  !    0.454   124 2010/10/19
%% Q2206_f.fits     1   5652.1602   5656.9971 vfwhm=6.5  !    0.647   128   2
!  Stats:      7     1.1208850  252  220  0.105  0
MgII     1.0169320212    0.0000262899    30.25758      2.08731  13.438610    0.023898  0 !
MgII     1.0171761089    0.0000113353     7.47677      2.83208  14.685478    1.336895  0 !
MgII     1.0174467271    0.0000044869     0.20000      0.25662  11.395206    0.368958  0 !
MgII     1.0175749588    0.0000046127     5.32584      1.27258  11.416082    0.061674  0 !
MgII     1.0166634806    0.0000016698     4.95192      0.36214  13.038178    0.019946  0 !
MgII     1.0165951259    0.0000051255     3.55825      0.58716  12.192366    0.108438  0 !
MgII     1.0168093195    0.0000013777     7.05837      0.59932  13.303292    0.040241  0 !
<>       3.6514092229SZ  0.0000000000     8.19986      8.12270   0.999496    0.002665  0 !
MgII     1.0172744510    0.0000057217     2.65701      0.79527  14.545276    0.766496  0 !
MgII     1.0169380097    0.0000019075     1.46473      1.06155  12.918981    0.842270  0 !
MgII     1.0170333214    0.0000069261     1.18010      0.95608  12.575938    0.344313  0 !
```

(Note that sometimes after adding a continuum adjustment the program terminates because the normalized χ^2 increases - normally because the continuum adjustment is not needed. In such cases just restart from where it left off by using the output summary file as input.)

You might like the MgII and FeII to share the same velocity structure. If you do then you'll need to force the program to do that, and fit yet again with a new input file which you can construct from the two you have by merging the two files sensibly. To do this use Unix commands:

```
$ cp f26.9p5w5640 temp
$ cat f26.9p5w4728 >> temp
$ grep '%%' temp > fort.13
$ grep -v '%%' temp | sort -n -k 2,2 >> fort.13
```

This should give you a merged startup file which looks like:

```
%% Q2206_f.fits     1   5637.7109   5642.3652 vfwhm=6.5  !    0.454   124 2010/10/19
%% Q2206_f.fits     1   5652.1602   5656.9971 vfwhm=6.5  !    0.647   128   2
%% Q2206_f.fits     1   4726.8008   4729.6006 vfwhm=6.5  !    0.119    89 2010/10/19
%% Q2206_f.fits     1   4773.8838   4776.7646 vfwhm=6.5  !    1.000    91   2
%% Q2206_f.fits     1   4787.7378   4790.6665 vfwhm=6.5  !    0.651    92   3
%% Q2206_f.fits     1   4804.4941   4807.3672 vfwhm=6.5  !    0.138    89   4
%% Q2206_f.fits     1   5215.5986   5218.7466 vfwhm=6.5  !    0.087    91   5
%% Q2206_f.fits     1   5242.8657   5246.0151 vfwhm=6.5  !    0.009    90   6
!  Stats:      7     1.1208850  252  220  0.105  0
!  Stats:    14     1.1209734  542  519  0.029  0
MgII     1.0165951259    0.0000051255     3.55825      0.58716  12.192366    0.108438  0 !
MgII     1.0166634806    0.0000016698     4.95192      0.36214  13.038178    0.019946  0 !
FeII     1.0166651672    0.0000010938     6.27487      0.23210  12.768436    0.011167  0 !
FeII     1.0167800205    0.0000050995     4.94784      0.72731  12.801610    0.084319  0 !
MgII     1.0168093195    0.0000013777     7.05837      0.59932  13.303292    0.040241  0 !
FeII     1.0168379718    0.0000022476     4.14700      0.37738  13.108916    0.031467  0 !
MgII     1.0169320212    0.0000262899    30.25758      2.08731  13.438610    0.023898  0 !
FeII     1.0169360041    0.0000009907     5.11960      0.37417  12.881360    0.016497  0 !
MgII     1.0169380097    0.0000019075     1.46473      1.06155  12.918981    0.842270  0 !
MgII     1.0170333214    0.0000069261     1.18010      0.95608  12.575938    0.344313  0 !
FeII     1.0170344150    0.0000027054     5.46309      0.97965  12.523467    0.058155  0 !
FeII     1.0171632821    0.0000010815     7.38065      0.23725  14.000153    0.011283  0 !
MgII     1.0171761089    0.0000113353     7.47677      2.83208  14.685478    1.336895  0 !
FeII     1.0172616675    0.0000012609     4.62046      0.14859  13.651367    0.015614  0 !
MgII     1.0172744510    0.0000057217     2.65701      0.79527  14.545276    0.766496  0 !
MgII     1.0174467271    0.0000044869     0.20000      0.25662  11.395206    0.368958  0 !
MgII     1.0175749588    0.0000046127     5.32584      1.27258  11.416082    0.061674  0 !
<>       3.2916026553SZ  0.0000000000    57.62327     17.11410   0.991529    0.003448  0 !
<>       3.65140922295Z  0.0000000000     8.19986      8.12270   0.999496    0.002665  0 !
```

You need to edit this to tell the program which FeII are associated with which MgII. This is done by putting lower case letters after the redshift of one (the reference redshift), and corredponding upper case letters after the associated ion (or ions if there are several), and perhaps editing a Doppler parameter or two to make them look compatible. You can normally get away with some fairly drastic looking associations in the hope that the program will sort them out. So a restart file might look like:

```
%% Q2206_f.fits     1   5637.7109   5642.3652 vfwhm=6.5  !    0.454   124 2010/10/19
%% Q2206_f.fits     1   5652.1602   5656.9971 vfwhm=6.5  !    0.647   128   2
%% Q2206_f.fits     1   4726.8008   4729.6006 vfwhm=6.5  !    0.119    89 2010/10/19
```

```
%% Q2206_f.fits     1   4773.8838   4776.7646 vfwhm=6.5  !    1.000    91   2
%% Q2206_f.fits     1   4787.7378   4790.6665 vfwhm=6.5  !    0.651    92   3
%% Q2206_f.fits     1   4804.4941   4807.3672 vfwhm=6.5  !    0.138    89   4
%% Q2206_f.fits     1   5215.5986   5218.7466 vfwhm=6.5  !    0.087    91   5
%% Q2206_f.fits     1   5242.8657   5246.0151 vfwhm=6.5  !    0.009    90   6
!  Stats:      7     1.1208850  252  220  0.105  0
!  Stats:     14     1.1209734  542  519  0.029  0
MgII     1.0165951259    0.0000051255     3.55825      0.58716  12.192366    0.108438  0 !
MgII     1.0166634806AA    0.0000016698     4.95192      0.36214  13.038178    0.019946  0 !
FeII     1.0166651672aa    0.0000010938     6.27487      0.23210  12.768436    0.011167  0 !
FeII     1.0167800205ab    0.0000050995     4.94784      0.72731  12.801610    0.084319  0 !
MgII     1.0168093195AB  0.0000013777     7.05837      0.59932  13.303292    0.040241  0 !
FeII     1.0168379718ac    0.0000022476     4.14700      0.37738  13.108916    0.031467  0 !
MgII     1.0169320212AC    0.0000262899    10.25758      2.08731  13.438610    0.023898  0 !
FeII     1.0169360041ad    0.0000009907     5.11960      0.37417  12.881360    0.016497  0 !
MgII     1.0169380097AD    0.0000019075     1.46473      1.06155  12.918981    0.842270  0 !
MgII     1.0170333214AE    0.0000069261     1.18010      0.95608  12.575938    0.344313  0 !
FeII     1.0170344150ae    0.0000027054     5.46309      0.97965  12.523467    0.058155  0 !
FeII     1.0171632821af    0.0000010815     7.38065      0.23725  14.000153    0.011283  0 !
MgII     1.0171761089AF    0.0000113353     7.47677      2.83208  14.685478    1.336895  0 !
FeII     1.0172616675ag    0.0000012609     4.62046      0.14859  13.651367    0.015614  0 !
MgII     1.0172744510AG    0.0000057217     2.65701      0.79527  14.545276    0.766496  0 !
MgII     1.0174467271    0.0000044869     0.20000      0.25662  11.395206    0.368958  0 !
MgII     1.0175749588    0.0000046127     5.32584      1.27258  11.416082    0.061674  0 !
<>       3.2916026553SZ  0.0000000000    57.62327     17.11410   0.991529    0.003448  0 !
<>       3.6514092229SZ  0.0000000000     8.19986      8.12270   0.999496    0.002665  0 !
```

The resultant fit might be (with all the file region information removed):

```
MgII     1.0166004836    0.0000062796     4.00829      0.48429  12.276427    0.072433  0 !
MgII     1.0166636921AA  0.0000000000     4.24054      0.47065  13.016418    0.062213  0 !
FeII     1.0166636921aa  0.0000008504     5.90601      0.33128  12.749250    0.015210  0 !
FeII     1.0168133638ab  0.0000012695     8.66508      0.30080  13.183416    0.017713  0 !
MgII     1.0168133638AB  0.0000000000     7.18169      1.01723  13.300394    0.070999  0 !
FeII     1.0168423802ac  0.0000025199     2.20957      0.46396  12.695545    0.064481  0 !
MgII     1.0168423802AC  0.0000000000    24.63564      2.29012  13.239352    0.130358  0 !
FeII     1.0169384714ad  0.0000010284     4.51217      0.38700  12.839721    0.018557  0 !
MgII     1.0169384714AD  0.0000000000     3.42578      0.81127  12.794066    0.072467  0 !
MgII     1.0170351884AE  0.0000000000     6.90747      0.84268  12.876522    0.063838  0 !
FeII     1.0170351884ae  0.0000031022     6.06827      1.03482  12.562298    0.060039  0 !
FeII     1.0171625465af  0.0000011475     7.20960      0.23488  13.996297    0.011183  0 !
MgII     1.0171625465AF  0.0000000000     6.12331      1.12426  14.660359    0.487784  0 !
FeII     1.0172606943ag  0.0000012114     4.70418      0.14126  13.663937    0.022953  0 !
MgII     1.0172606943AG  0.0000000000     3.34677      0.15343  14.974409    0.179412  0 !
MgII     1.0174447068    0.0000045346     0.20000      0.25437  11.384483    0.317722  0 !
MgII     1.0175750447    0.0000046385     5.21090      1.53644  11.399992    0.074632  0 !
<>       3.2916026553SZ  0.0000000000    58.30159     15.46498   0.991512    0.004118  0 !
<>       3.6514092229SZ  0.0000000000     6.91587      7.82659   0.998618    0.002805  0 !
```

This is still not quite right, since the FeII Doppler parameter should be $\leq$ that for MgII, and no less than that for MgII $\times\sqrt{(\text{MgII/FeII mass ratio})}$. You can ensure this by tagging the associated Doppler parameters (see section 11.3). You might wish to try this, but constraining the parameters further by linking them in this way does mean

you are likely to need further velocity components.

17 Fitting the Lyα forest

Automatic fitting of single regions with a known line is handled by setting up startup files with just those regions in. In the default mode the file

```
 %% q2000_norm_sp.fits  1  5502.57185  5513.57833

 %% q2000_norm_sp.fits  1  4641.05459  4651.89758

 %% q2000_norm_sp.fits  1  4399.58159  4410.39294

...
```

could be used with VPFIT, and then with the usual startup

```
....
  options:   <CR> for previous value
  I - interactive setup and fit
  F - run from an input file
  D - display profiles from input file
  ? for help
  option (key) (key)...
> f ad

 Add lines if prob(tot) < 0.010000   or prob(KS) < 0.00000001

 Column density (n), logN (l) or emission (e), scalefactor [l, 1.0]
>
 Parameter input file, # entries? [fort.13,1]
> test.13 3
...
```

the program generates its own list of Lyα lines for each region and adds more until it obtains a satisfactory fit (or gives up because it can't). Each of these regions is fitted independently, and the results go to the summary file.

However, these regions were chosen to correspond to Lyα, Lyβ and Lyγ centered on the same redshift (~ 3.53 in this case), and what one might really like to do is fit them automatically together. If all have strong lines just putting the longest wavelength region (Lyα) last in the list and omitting the blank lines between them so they are all fitted together does not work very well. Lyα lines are put in for the last region only, and the other regions are then fitted so badly that the program can (i.e. does) lose its way completely.

An alternative procedure is to use the input file given above and fit the regions independently, but ensuring that the lines found in the higher wavelength regions are put into the continuum for the one which is being fitted. This can be done by

```
....
  options:   <CR> for previous value
  I - interactive setup and fit
```

```
  F - run from an input file
  D - display profiles from input file
  ? for help
  option (key) (key)...
> f ad cum test.cum

 Add lines if prob(tot) < 0.010000   or prob(KS) < 0.00000001

 Column density (n), logN (l) or emission (e), scalefactor [l, 1.0]
>
 Parameter input file, # entries? [fort.13,1]
> test.13 3
...
```

The effect of the `cum test.cum` is to write the results cumulatively to the file `test.cum`, and every time a new fit is started within VPFIT the parameters are read from this same file and applied to the continuum before the fit is undertaken. So, when fitting the Lyβ region in the example above, the results of the Lyα fits are put in as Lyβs in that region, and then Lyα lines added to that region as needed. You can continue this process up the Lyman series if you want to. The test case shown here stops at Lyγ.

The result of this procedure is not likely to be wonderful either, since the first Lyαs were fitted without the higher order line constraints, and you need those particularly for the higher column density systems. You can fix this by going around again, with a new input file of first guesses from `test.cum` which you can make by

```
$ grep '%%' test.cum > restart.13
$ grep -v '%%' test.cum >> restart.13
```

You now have a file which looks like

```
%% q2000_norm_sp.fits     1   5502.5719   5513.5783    !
%% q2000_norm_sp.fits     1   4641.0546   4651.8976    !
%% q2000_norm_sp.fits     1   4399.5816   4410.3929    !
! Stats:   6     0.9356639  220  185  0.725  0
 H I   3.5280373082 0.0000040424  24.00206  0.42508  14.153667  0.006331  0 !
 H I   3.5302387347 0.0000047564  23.60935  0.31797  13.773950  0.006908  0 !
 H I   3.5319355734 0.0001543133  22.66707  6.33160  13.567904  0.331476  0 !
....
```

which is now a first guess file for restarting VPFIT, now doing a fit to all the lines simultaneously over the regions.

```
....
  option (key) (key)...
> f ad

 Add lines if prob(tot) < 0.010000   or prob(KS) < 0.00000001

 Column density (n), logN (l) or emission (e), scalefactor [l, 1.0]
>
```

```
 Parameter input file, # entries? [fort.13,1]
> restart.13
....
```

There is still room for confusion - metal lines and bad zero levels are potential problems. You can avoid metal lines by having their parameters in the `test.cum` file before you start. The trouble then is that you have to strip out that information from the `restart.13` file when you are creating it. Zero level adjustments can be edited into the `restart.13` file if you need to, but that has to be done by hand. It is probably not the greatest idea to put zero levels in straight way, but to wait until VPFIT has dealt with the `restart.13` file as far as it can (i.e. until adding more lines does not help), and then putting zero level corrections into that best fit, and then restarting yet again.

This process can be slow with the default startup parameters for VPFIT. To speed things up for echelle forest spectra in the VPFSETUP file use

```
! ----  internal substepping ----
 nsubmin 1          minimum number of subpixels per pixel
 nsubmax 1          maximum number of subpixels per data pixel
```

otherwise spurious narrow lines force internal substepping. If you want to you can reset `nsubmax` to a larger number for a final go-through.

There is still a hand-held component to all this, and that is setting the fitting regions and generating the intial startup files. One way of doing this is to use RDGEN in the cursor plot mode ('`pg`'), setting region boundaries by cursor position (using '`b`' in the plot window) for Lyman lines plotted in stack velocity mode. See the RDGEN writeup for details of this.

Also, the 'cum' option appends the results to the file, whether or not the fit is bad. You may wish to just include previous fits using 'inc' instead, and then append the results if acceptable. You can check this in the summary file by looking either at the fit probability, or the flag 'BAD' which is printed on the same line if the probability is below threshold.

While this is OK for single sets of regions, and so be used for the Lyα-only region of the forest, it is still difficult to fit the whole forest self-consistently in this way. With individual region sets lines identified as Lyα near e.g. a Lyγ of interest may well be Lyβs. If you want to do it properly there are too many linked regions by the time you get a reasonable way down the Lyman series. If anybody wishes to suggest a solution to this problem I'd be glad to hear about it.

A Ancillary programs

A.1 FITS header editing

There is a simple program HEDIT for adding (or removing) items from FITS headers for those who don't have the capability on their machine e.g. through IRAF. On linux machines 'make hedlx' will compile and link it, after the makefile has been edited to reflect where the CFITSIO library is held.
Then

```
./hedit
```

will run it. There is one request input line

```
Filename, variable, value, action?
```

Entering a '?' then gives

```
Parameters are:
1: full filename (including the .fits)
2: keyword to be added/removed/updated
3: value associated with the keyword
    this is stored as a character string or
    a double precision variable
4: update (u), delete(d) or full list (l)
    must be lower case
 If 4th parameter omitted, update is assumed
 unless 2nd and 3rd omitted as well, in which
 which case an abbreviated keyword list is given
 (HISTORY, HIERARCH & ESO omitted)
```

... which about says it all. The 'u' option will update a keyword if it is already in the FITS header, and add it if it is not.

For example, to add 'RESVEL 6.6' to a fits header in a file example.fits,

```
./hedit
Filename, variable, value, action?
example.fits RESVEL 6.6 u
```

B Rebinning and combining spectra

B.1 Error estimates

We assume that in each raw data channel i we have a signal f_i with a Gaussian distributed error e_i. We wish to rebin the data on to a new channel set, and assign weights accordingly. We suppose that the initial data bins are contiguous, and for the final ones this is also true - the example we have in mind is rebinning spectrscopic data on to a uniform wavelength or velocity scale.

If the original chosen bin i is such that some fraction of it w_{ij} falls in to the jth bin in the rebinned data, then it adds $w_{ij}f_i$ to bin j[2]. Hence $\sum_j w_{ij} = 1$, and we are conserving the total signal in the entire spectrum with this scheme.

The total signal in bin j is

$$F_j = \sum_i w_{ij} f_i \tag{1}$$

We know that *If the independent variates x_i $(i = 1, ..., n)$ are normally distributed with means a_i and variances σ_i^2, the variate $\sum c_i x_i$ is normally distributed with mean $\sum c_i a_i$ and variance $\sum c_i^2 \sigma_i^2$.* (e.g. Weatherburn, C. E., *A First Course in Mathematical Statistics* CUP, 1968, p58.). Hence the variance in

$$F_j = \sum_i w_{ij} f_i \;\text{ is }\; \sum_i w_{ij}^2 e_i^2 \tag{2}$$

We can of course normalize this to anything we want - flux per wavelength bin, unit velocity ... by setting

$$G_j = \frac{F_j}{B_j} \pm \frac{\sqrt{\sum_i w_{ij}^2 e_i^2}}{B_j} \tag{3}$$

where B_j is the width of each j-bin. The B_j can also be divided by the exposure time or any other constant obviously.

The error estimate $\sum_i w_{ij}^2 e_i^2$ provides an estimate of the expected deviation per pixel, but if we take n pixels and sum exactly into N new pixels then the **total** signal-to-noise ratio in the original pixellation is

$$\frac{\sum_{i=1}^{n} f_i}{\sqrt{\sum_{i=1}^{n} e_i^2}} \tag{4}$$

[2]Note that this weighting assumes that the flux within a pixel is distributed uniformly across the pixel. This is unlikely to be true in cases of interest i.e. where there is a spectral feature. However it is not obvious how to get around it - you'd need to model the spectrum which you have not formed yet, and so presumably know little about! A consequence of this incorrect sub-pixel weighting is likely to be small wavelength shifts relative to the true values in the rebinned data.

and in the new pixellation it is

$$\frac{\sum_{j=1}^{N} F_j}{\sqrt{\sum_{j=1}^{N}\sum_{i=1}^{n} w_{ij}^2 e_i^2}} = \frac{\sum_{i=1}^{n} f_i}{\sqrt{\sum_{j=1}^{N}\sum_{i=1}^{n} w_{ij}^2 e_i^2}}. \tag{5}$$

We can interchange the order of summation so the new pixel total S/N is then

$$\frac{\sum_{i=1}^{n} f_i}{\sum_{i=1}^{n} e_i^2 \sum_{j=1}^{N} w_{ij}^2} \tag{6}$$

Now each $w_{ij} \leq 1$, so $\sum_{j=1}^{N} w_{ij}^2 \leq \sum_{j=1}^{N} w_{ij}$, and so in general the total S/N estimate for the repixellated spectrum is higher than in the original data. This is not what we want if we want to maintain overall S/N in the data.

The requirement that the total S/N is the same over this range of pixels in either pixel scheme then becomes a requirement that

$$\sum_{i=1}^{n} e_i^2 = \sum_{j=1}^{N}\sum_{i=1}^{n} b_{ij} e_i^2 \tag{7}$$

where the b_{ij} are weights on the variances which we have to determine. Switching the summation order this becomes

$$\sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} e_i^2 \sum_{j=1}^{N} b_{ij} \tag{8}$$

which can be satisfied for any n only if $\sum_{j=1}^{N} b_{ij} = 1$. But this, for any N (or n) is exactly the condition we had on $\sum_j w_{ij}$, so this means that to maintain S/N over numbers of pixels we should have $F_j = \sum_i w_{ij} f_i$ with an error estimate S_j given by

$$S_j^2 = \sum_i w_{ij} e_i^2. \tag{9}$$

So, in summary, for maintaining overall S/N, use

$$F_j \pm S_j = \sum_i w_{ij} f_i \pm \sqrt{\sum_i w_{ij} e_i^2}, \tag{10}$$

and for pixel-to-pixel fluctuations (e.g. which you would use in minimizing $\chi^2 = \sum_j (F_j - \text{model}_j)^2/\text{error}_j^2$) use

$$F_j \pm E_j = \sum_i w_{ij} f_i \pm \sqrt{\sum_i w_{ij}^2 e_i^2}. \tag{11}$$

Note that for summing different spectra using inverse variance weighting equation (10) should be used. Equation (11) if used overweights regions where the old and new pixels mesh badly. Then, for data normalized to the same scale, use inverse variance weighting when the variance= S_j^2 for each set, i.e. for k as a label for each different spectrum

$$\mathcal{F}_j = \frac{\sum\limits_k \frac{F_j^k}{(S_j^k)^2}}{\sum\limits_k \frac{1}{(S_j^k)^2}} \tag{12}$$

and variance $\mathcal{S}_j^2$ is given by

$$\frac{1}{\mathcal{S}_j^2} = \sum_k \frac{1}{(S_j^k)^2} \tag{13}$$

where $(S_j^k)^2$ is the **population** variance for the jth channel in the kth spectrum. This is slightly different from the estimate of the variance based on the data. The next section explores this a bit further.

The fluctuations follow a normal distribution, so we can use the combination result to get final fluctuations

$$\mathcal{E}_j^2 = \frac{\sum\limits_k \left(\frac{E_j^k}{(S_j^k)^2}\right)^2}{\left(\sum\limits_k \frac{1}{(S_j^k)^2}\right)^2} \tag{14}$$

While these fluctuation estimates are correlated in rebinned spectrum, they do represent the expected values of the fluctuations in the data, so these are the appropriate quantities to use for the errors e_i when forming $\chi^2 = \Sigma(d_i - f_i)^2/e_i^2$, where d_i is the data and f_i the fit at pixel i.

(As a check, in the unrebinned case $E_j^k = S_j^k$, and then the last equation becomes $\mathcal{E}_j^2 = \frac{1}{\sum\limits_k \frac{1}{(S_j^k)^2}}$, which is correct.)

B.2 Biases in combined datasets

If we have a sample of k observations all drawn from the same population, with mean m and (Gaussian) error for the ith sampling σ_i, then for each observation we will have a value $m + \epsilon_i$, where ϵ_i is distributed as

$$\frac{1}{\sqrt{2\pi}\sigma_i} \exp\left(-\frac{\epsilon_i^2}{2\sigma_i^2}\right) \tag{15}$$

Since neither m or σ_i are known, these are estimated from the data, with the data value d_i taken as an estimator for m, and the variance

$$s_i^2 \approx A_i \times d_i + B_i \tag{16}$$

where A_i is a scalefactor and B_i allows for additional noise contributions e.g. from sky or detector readout.

In combining the data to estimate m we take

$$D = \frac{\sum_{i=1}^{k} w_i d_i}{\sum_{i=1}^{k} w_i} \tag{17}$$

where w_i are the weights for each estimate.

To maximize the resultant signal-to-noise, then an appropriate weight is the inverse variance $\frac{1}{\sigma_i^2}$, so

$$D = \frac{\sum_{i=1}^{k} \frac{d_i}{\sigma_i^2}}{\sum_{i=1}^{k} \frac{1}{\sigma_i^2}} \tag{18}$$

That is fine, but we don't know σ_i, and an obvious way to proceed is to estimate it using the s_i values from the relation above. Unfortunately this provides a biased estimate for both D and the variance in D. Rather than attempt to show this in full generality, we take a simple example which is related at least to the way we sum data for an individual channel from several exposures.

We suppose that we expect n photons in a single observation, and take k of them. then, using inverse variance estimate weighting scheme, we maximize the signal-to-noise ratio by forming

$$D = \frac{\sum_{i=1}^{k} \frac{d_i}{s_i^2}}{\sum_{i=1}^{k} \frac{1}{s_i^2}} \tag{19}$$

where (if there is no background noise component) we can take

$$s_i = \sqrt{d_i} \tag{20}$$

Then D becomes

$$D = \frac{k}{\sum_{i=1}^{k} \frac{1}{d_i}} \tag{21}$$

To see what is going on, set

$$d_i = n + \epsilon_i \tag{22}$$

where ϵ_i is normally distributed about zero with variance n. Then

$$D = \frac{k}{\sum_{i=1}^{k} \frac{1}{n+\epsilon_i}} \tag{23}$$

i.e.

$$D = \frac{k}{\sum_{i=1}^{k} \frac{1}{n(1+\frac{\epsilon_i}{n})}} \tag{24}$$

or

$$D = \frac{nk}{\sum_{i=1}^{k} \frac{1}{1+\frac{\epsilon_i}{n}}}. \tag{25}$$

Now

$$\frac{1}{1+\frac{\epsilon_i}{n}} = 1 - \frac{\epsilon_i}{n} + \left(\frac{\epsilon_i}{n}\right)^2 - ... \tag{26}$$

so

$$D = \frac{n}{1 - \frac{1}{kn}\sum_{i=1}^{k}\epsilon_i + \frac{1}{kn^2}\sum_{i=1}^{k}\epsilon_i^2 - ...} \tag{27}$$

$$= n\left(1 + \frac{1}{kn}\sum_{i=1}^{k}\epsilon_i - \frac{1}{kn^2}\sum_{i=1}^{k}\epsilon_i^2 + \left(\frac{1}{kn}\sum_{i=1}^{k}\epsilon_i\right)^2 ...\right) \tag{28}$$

We take each of these summation terms in turn, and ignore higher order terms. Generally those with odd powers will average to zero, and even powers contributeto higher orders of $\frac{1}{n}$.

- $\sum_{i=1}^{k}\epsilon_i$ has an expectation value zero and variance $k\sigma^2$, so $\frac{1}{kn}\sum_{i=1}^{k}\epsilon_i$ has mean zero and 1σ deviation $\frac{1}{\sqrt{kn}}$.

- The next term looks like the sum of variances, so an estimator for $\sum_{i=1}^{k}\epsilon_i^2$ is $k\sigma^2 = kn$.

- The quantity $\left(\sum_{i=1}^{k}\epsilon_i\right)^2$ can be estimated using the fact that the distribution of $\sum_{i=1}^{k}\epsilon_i$ is a Gaussian with mean zero and variance kn, so for a variable x and $\nu = \sqrt{kn}$ the distribution is

  $$p(x)dx = \frac{1}{\sqrt{2\pi}\nu}\exp\left(-\frac{x^2}{2\nu^2}\right)dx \tag{29}$$

  Setting $y = x^2$ this becomes, for $y \in [0, \infty]$

  $$p(y)dy = \frac{2}{\sqrt{2\pi}\nu}\exp\left(-\frac{y}{2\nu^2}\right)\frac{1}{2\sqrt{y}}dy \tag{30}$$

  where the extra 2 appears because x can be negative or positive for a given y.

  Therefore

  $$\langle y\rangle = \frac{1}{\sqrt{2\pi}\nu}\int_0^\infty \exp\left(-\frac{y}{2\nu^2}\right)\sqrt{y}dy \tag{31}$$

or setting $2\nu^2 z = y$ this is

$$\langle y \rangle = \frac{2\nu^2}{\sqrt{\pi}} \int_0^\infty e^{-z} \sqrt{z} dz \tag{32}$$

$$= \frac{2\nu^2}{\sqrt{\pi}} \Gamma\left(\frac{3}{2}\right) \tag{33}$$

Now $\Gamma(\frac{3}{2}) = \frac{1}{2}\Gamma(\frac{1}{2})$, and $\Gamma(\frac{1}{2}) = \sqrt{\pi}$, so

$$\langle y \rangle = \nu^2 = kn \tag{34}$$

So the the corresponding term will on average be $\frac{1}{kn}$.

Bringing all these together the result is

$$\langle D \rangle = n\left(1 + 0 - \frac{1}{n} + \frac{1}{kn}\right) \tag{35}$$

i.e.

$$\langle D \rangle = n - \left(1 - \frac{1}{k}\right) \tag{36}$$

So $\langle D \rangle$ is a biased estimator for the true counts n. And, unfortunately, the relative bias depends on the signal-to-noise ratio $\sqrt{n}$ (though in this example the absolute bias does not).

This general result is known by optimal extraction people (e.g. Horne K., PASP, 98, 609, 1986; Marsh T. R., PASP, 101, 1032, 1989) and is the reason they effectively smooth the weights when adding into the overall profile.

For combining spectra (or anything) we need a better estimate of the relative local sigmas to provide the appropriate weights.

B.3 Suitable weights for combining spectra

The error estimates S_j used in equations (12), (13) and (14) need to be based on estimates for the population values, and we have seen that single pixel estimates bias the data. The error ratios between spectra are likely to be smooth functions, depending primarily on blaze functions, camera functions, detector response and atmospheric extinction, most of which can be taken as varying slowly with wavelength over the scales of interest. Atmospheric absorption (and emission) can present problems, but if the airmasses are similar we might, to first order, expect that their contributions will mimic absorption lines (except where the heliocentric correction moves them in one spectrum relative to another).

With this in mind, can construct a median or mean or fitted variance for each spectrum, and try using those as weights. It may not matter much which is used, since all we need is something which

- scales with the variance in the same way between spectra, and
- is not biased by single pixel small fluctuations.

Then if we set weight $u_j^k = \frac{1}{\left\langle (S_j^k)^2 \right\rangle}$, where $\langle\rangle$ is some average/median/fit over several j pixels evaluated at j, we can form a combined spectrum by using

$$\mathcal{F}'_j = \frac{\sum\limits_k u_j^k F_j}{\sum\limits_k u_j^k}. \tag{37}$$

Then, using equation (2) with $w_{jk} = u_j^k / \sum\limits_k u_j^k$, the variance is

$$\left(\mathcal{S}'_j\right)^2 = \frac{\sum\limits_k (u_j^k S_j^k)^2}{\left(\sum\limits_k u_j^k\right)^2} \tag{38}$$

and fluctuations

$$\left(\mathcal{E}'_j\right)^2 = \frac{\sum\limits_k (u_j^k E_j^k)^2}{\left(\sum\limits_k u_j^k\right)^2} \tag{39}$$

Again as a sanity check setting $u_j^k = \frac{1}{(S_j^k)^2}$, the individual pixel values, gives the answers we had in equations (12), (13) and (14).

The only thing left to worry about is which form one should use for u_j^k. Effectively we want a (relatively) unbiased quantity which scales as the local variance in the same way between spectra, and almost any scheme which results in some estimate based on the local properties over a range of pixels will do. You may wind up with something which is a little less than optimum in a S/N sense, but it will be less biased.

B.4 Wavelength bias

In a footnote in section B.1 we noted a possible bias in wavelength estimates caused by rebinning. We return to this question using an illustrative example (we hope) based on an unrealistic absorption profile which is simple enough to see what is happening. We choose one which is just a triangle, so the flux f is unity for $x \leq 0$ and $x \geq 1$, and in the range [0,1) $f = 1 - x$. Then the mean wavelength

$$\lambda = \frac{\int_0^1 x(1-f)dx}{\int_0^1 (1-f)dx} = \frac{\int_0^1 x^2 dx}{\int_0^1 x dx} = \frac{2}{3}.$$

If we choose, for example, to record this profile in 3 data pixels $x_i = [0, 1/3), [1/3, 2/3), [2/3, 1)$, then these pixels are centered on $x_i = 1/6$, 3/6 & 5/6, with $d_i = 1 - f_i$ values of 1/6, 3/6 & 5/6. Then if we determine the wavelength by

$$\lambda' = \frac{\sum x_i d_i}{\sum d_i} = \frac{35}{36} \Big/ \frac{9}{6} = \frac{35}{36} \times \frac{2}{3} = 0.6481.$$

What is happening is clear - the centre of the pixel is being taken as the mean position for the line depth, while it should be further along because of the line shape. Things

obviously improve if you take a finer pixellation. Using five, for example, with $x_i(= d_i) =$ 0.1, 0.3, 0.5, 0.7 & 0.9 the result is $\frac{99}{100} \times \frac{2}{3} = 0.660$

So for this profile just recording the data gives a wavelenth bias if we use the central pixel wavelength as the appropriate one for the data in that pixel.

Now what happens if we take the data which was in three pixels, and rebin that to a five pixel set using the prescription in section B.1? The new values of the line depths d_i^p at each x_i are (expressed as fractions)

x_i	d_i^p
1/10	3/18
3/10	5/18
5/10	9/18
7/10	13/18
9/10	15/18

The mean wavelength is now

$$\lambda^p = \left(\frac{3+15+45+91+135}{180}\right) / \left(\frac{1+3+5+7+9}{10}\right) = \frac{289}{450} = 0.6422,$$

so we have added a further bias to the wavelength estimate.

This is a pathological example, but it does illustrate what may happen if e.g we have an asymmetric profile, which could arise, for example, by a blend of symmetric ones where there is a redshift - column density correlation. For the intrinsic shift problem it is a matter of modelling the profile correctly, and VPFIT (version 9.0 onwards) allows you to do this by computing the profiles at a finer level than the pixel size (see section 10). You can subsample a profile as far as you like - the cost is only in computing time and array space.

This does, of course, assume that the wavelength scale has been correctly determined from the comparison arc lines or whatever. In particular the comparison line positions are determined by suitably integrating the model profile which falls within each pixel in determining the centroid position. You should look at the documentation for whatever wavelength calibration package you use - if you are lucky it may tell you what is done, and if you are even luckier it may do it accurately.

For rebinning the data the solution is less clear. If the profile is symmetric, and centered either on the middle of a new pixel or the edge of one, then symmetry arguments show that there will be no wavelength shift. However if the new pixellation centers fall somewhere else there may be a shift. As an example the profile

```
 996.0  1.0
 997.0  1.0
 998.0  1.0
 999.0  1.0
1000.0  1.0
1001.0  0.875
1002.0  0.625
1003.0  0.375
1004.0  0.125
```

```
1005.0  0.125
1006.0  0.375
1007.0  0.625
1008.0  0.875
1009.0  1.0
1010.0  1.0
1011.0  1.0
1012.0  1.0
1013.0  1.0
```

which clearly has a centroid at 1004.50, was mapped onto pixels of width 0.85 in these wavelength units, with starting wavelength 997.75. The result is

```
  1000.    1.0
  1000.85  0.886029412
  1001.7   0.691176471
  1002.55  0.485294118
  1003.4   0.279411765
  1004.25  0.125
  1005.1   0.132352941
  1005.95  0.375
  1006.8   0.588235294
  1007.65  0.794117647
  1008.5   0.9375
  1009.35  1.0
```

which gives a mean wavelength of 1004.495039, i.e. a shift of 0.005 of the original pixel size.

The only lesson I can derive from this is - don't rebin the spectrum. This may mean you have to deal with each exposure individually (which brings some problems with continuum level matching) if you want wavelength accuracy.

Index