ROBOSPECT: Automated Equivalent Width Measurement
A User's Guide

Christopher Z Waters
Julie K Hollek

August 13, 2013 v2.10

Contents

1 Introduction **2**

2 Installation **3**
2.1 Compilation 3

3 Quick Examples **4**

4 Data Handling **5**
4.1 Input Files 5
4.1.1 Input Spectra 5
4.1.2 Linelist 5
4.2 Output files 6
4.2.1 Line catalog 6
4.2.2 Output spectrum model 7
4.2.3 Line fit plots 8
4.2.4 Log file 8

5 Command Line Options **9**
5.1 Fitting Parameters 9
5.2 Model Selection 10
5.3 Line Identification 13
5.4 Program Control 14

6 Calculations **16**
6.0.1 Spectrum Models 16
6.0.2 Line fitting 18
6.0.3 Line profile functions 20

7 Adding Modules **21**

Chapter 1

Introduction

ROBOSPECT is a program designed to automate the measurement of spectral line features in astronomical data. The goal is to determine the best fitting equivalent widths for each feature, while at the same time constructing a model of the input spectrum continuum and line profiles.

Although originally designed to perform measurements of a known list of well-isolated lines for metal-poor stars, additional features have been added to ROBOSPECT that allow it to be used on a variety of astrophysical sources. A short list of these features include

- Identify and measure lines above a user-specified signal-to-noise threshold.
- Support for FITS spectra.
- Radial velocity measurement.
- Modular design to allow for additional fitting functions to be added.

A detailed explanation of the algorithms used in ROBOSPECT, as well as a comparison of the automated measurements to those performed by hand can be found in the Waters & Hollek (2013), available at `http://arxiv.org/abs/1308.0757`.

ROBOSPECT is free software, **Julie:** you can put whatever you want to say about that here .

Chapter 2

Installation

Source code for ROBOSPECT is available at `http://www.ifa.hawaii.edu/users/watersc1/robospect/`. The current released version is v2.10. It is written in C, and therefore needs a compiler to build. ROBOSPECT depends on various C libraries, and these libraries need to be installed prior to compiling.

The GNU Scientific Library (GSL) is a numerical library for C that is used to handle various mathematical operations. It may be obtained from `http://www.gnu.org/software/gsl/`, and installed following the instructions found there.

FITS spectra can be used with ROBOSPECT, but this requires installing the CFITSIO library (`http://heasarc.gsfc.nasa.g` This library is not essential for compilation. If CFITSIO is not installed, ROBOSPECT can still be used on ASCII spectra.

Finally, ROBOSPECT has the ability to directly plot the best fitting line models. The PLplot library (`http://plplot.sourceforge.net`) is currently the only supported plotting library to handle this. This is also an optional library, as ROBOSPECT can be compiled and run without generating plots.

2.1 Compilation

Once all dependencies have been installed, ROBOSPECT can be built simply by entering the directory created from unpacking the tar.gz file, and issuing the commands:

```
./configure && make
```

This produces a binary file src/robospect which can then be installed into the user's path. After compiling, `./src/robospect -h` can be run to check that no errors were generated. If the optional libraries were found installed on the system, this command will list this fact:

```
Compiled with libplplotd support for plotting.
Compiled with libcfitsio support for fits io.
```

Chapter 3

Quick Examples

The following examples show a small set of “cookbook” examples that can be used to quickly perform fits without reading through the description of all the program options. See below for further description of the command line options and file formats.

Print help page with all command line parameters. The `-h` help option can be appended to any robospect command, and will print the value of all user-configurable parameters that are set in that command line. By itself, it shows the default values used by ROBOSPECT.

```
robospect -h
```

Run a normal fit on input stellar absorption spectra example.dat, saving results to files with the path and prefix “/tmp/example1”. The `-P` option sets the “path base” that is used to construct the output filenames. ROBOSPECT will attempt to find all lines (enabled by the `-F` option) with peak S/N ¿ 5.0 (set by `-f`) to fit.

```
robospect -F -f 5.0 -P /tmp/example1 example.dat
```

This performs the same kind of fit to example.dat, but reads in a user-supplied line list via the `-L` option.

```
robospect -L lines.dat -F -f 5.0 -P /tmp/example2 example.dat
```

As above, with verbosity increased to show line fitting activity:

```
robospect -L lines.dat -F -f 5.0 -P /tmp/example3 example.dat \
          -v line -v screen
```

Fit a flux calibrated spectrum, changing the continuum size to match an increase in resolution (`-V` sets the width of the continuum box in Angstroms).

```
robospect -L lines.dat -F -f 5.0 -P /tmp/example4 fc_example.dat \
          --loosen --flux_calibrated -V 10
```

Fit the flux calibrated spectrum again, but change the continuum estimation module (`-C`) to account for a large gradient in continuum values.

```
robospect -C logboxcar -L lines.dat -F -f 5.0 -P /tmp/example4 \
          fc_example.dat --loosen --flux_calibrated -V 10
```

Chapter 4

Data Handling

4.1 Input Files

4.1.1 Input Spectra

Both ASCII and FITS spectra can be specified for use with ROBOSPECT. ASCII data must have the following format:

```
WAVELENGTH FLUX OPTIONAL_ERROR OPTIONAL_ORDER
```

The `WAVELENGTH` and `FLUX` columns must exist to define the spectrum. The `OPTIONAL_ERROR` column can be used to supply an external estimate of the spectrum error to ROBOSPECT. In general, this is not necessary as ROBOSPECT will construct its own estimate of the error. However, if an `OPTIONAL_ORDER` is to be supplied, a column of zeroes in the `OPTIONAL_ERROR` column is needed for formatting. The `OPTIONAL_ORDER` column may contain a series of integers that denote that a single input spectrum consists of multiple spectral orders. Each order is fit independently, and produces its own output files. No comments or text should appear in the input spectrum file.

FITS spectra are supported if CFITSIO library support has been compiled into ROBOSPECT. The spectrum should exist in the primary image extension, and may contain either a 1-dimensional spectrum (which is read into the `WAVELENGTH` and `FLUX` components according to the FITS WCS information) or a 2-dimesional set of spectra (which is read as a multi-order spectrum based on the FITS WCS information).

If a valid FITS spectrum is not correctly read, this is likely due to the spectrum not matching examples of spectra that were available when writing the code. This kind of bug can likely be fixed quickly.

4.1.2 Linelist

The input linelist is an optional but highly recommended way to indicate which lines ROBOSPECT will fit. The format for this linelist is an ASCII file with columns:

```
LINE_CENTER_WAVELENGTH OPTIONAL_COMMENT
```

All text after the first column (containing the expected line center) are converted into a comment string for that line, which is retained and returned in the output line catalog.

4.2 Output files

4.2.1 Line catalog

Upon completion of fitting, all lines in the internal line catalog are saved to an output file of the form `PATH_BASE[_orderXXX].robolines`.

This file contains a comment block at the beginning of the file listing the command line used to generate the file, a summary of common flag hex values, and the column headers with the units for each. The columns returned are:

Column	Quantity	Description
1	x0	Linelist supplied center/auto-found line peak
2	mean	Best fitting line center
3	sigma	Best fitting line width (Gaussian sigma)
4	flux	Best fitting line flux
5	eta	Best fitting line eta parameter (for Voigt line profiles)
6	dmean	Error in line center (based on fit covariance matrix)
7	dsigma	Error in line width
8	dflux	Error in line flux
9	deta	Error in line eta
10	FWHM_mean	"pre" module estimate of line center
11	FWHM_s	"pre" module estimate of line width
12	FWHM_F	"pre" module estimate of line flux
13	EQW -or- FLUX	Line equivalent width or the line total flux, see below
14	dEQW -or- dFLUX	Error in line equivalent width or line total flux, see below
15	chi	χ^2 value of line fit
16	flags	Fit quality flags (described below)
17	group	Deblend group this line was fit with.
18	comment	Comment field from linelist or from line finding.

For continuum normalized spectra (where the input continuum level is approximately 1.0), columns 13 and 14 return the equivalent width and error in the equivalent width. However, for flux calibrated spectra, the equivalent width is poorly defined. Instead, these columns report the total integrated flux of the line and the associated error. Note that this flux is not the same as that reported in column 4, which has a factor related to the local continuum value included.

The "flags" column contains the bitwise-AND of a set of quality flags that are determined for each line. The program `utilities/robo_decode_flags.pl` can be used to convert this hexidecimal string into a list of flag parameters. The current list of defined flags are:

Flag name	Flag value	Description
ROBO_MAX_ITER	0x000001	Solver hit maximum number of iterations without returning a fit within tolerance.
ROBO_FIT_FAIL	0x000002	Solver returned impossible value and aborted further computation.
ROBO_FIT_IGNORED	0x000004	Line rejected and ignored from consideration due to concerns with fit parameters.
ROBO_FIT_REFUSED	0x000008	Line rejected and refused from consideration due to lack of χ^2 improvement with inclusion.
ROBO_FIT_LARGE_SHIFT	0x000010	Fit mean shifted significantly from expected initial value.
ROBO_FIT_RECENTER	0x000020	Fit mean reset to initial value and refit with fixed mean.
ROBO_BAD_WAVELENGTH	0x000040	Suspected bad wavelength solution around this line.
ROBO_FIX_WAVELENGTH	0x000080	Automatically corrected supplied line peak to fit bad wavelength solution.
ROBO_FIT_BLEND	0x000100	Line believed to be part of a blend.
ROBO_FIT_DEBLEND	0x000200	Line solution based on deblending modeling.
ROBO_ALT_REFUSED	0x001000	Alternate line model rejected and refused due to lack of χ^2 improvement with inclusion.
ROBO_BAD_ERROR_VAL	0x002000	Error value out of range and ignored.
ROBO_RESET	0x0000a0	Flags to reset on each solve iteration.
ROBO_BAD_FIT	0x00000f	Generic concerns with line. Not used in final model.
ROBO_INIT	0x010000	This signifies we haven't done anything to the line yet.
ROBO_FUNC_GAUSSIAN	0x100000	Line fit using a Gaussian model.
ROBO_FUNC_LORENTZIAN	0x200000	Line fit using a Lorentzian model.
ROBO_FUNC_HUMLICEK	0x400000	Line fit using a Humlicek (Voigt) model.
ROBO_FUNC_HJERTING	0x800000	Line fit using a Hjerting (Voigt) model.

In general, the hexadecimal digit in which a bit is raised can be used to quickly guess what is being reported in the flag value. Currently, the least significant digit notes fit failures, while the other digits signify other properties of the fit generated

0x00000F	Fit failed for some reason.
0x0000F0	Line had a correctable problem.
0x000F00	Line was deblended.
0x00F000	Fit has an issue with the "pre" fit model.
0x0F0000	Internal flags for state control.
0xF00000	Line function notification flag.

4.2.2 Output spectrum model

Upon completion of fitting, all lines that have flag values that do not exclude them as having an invalid fit are used to construct a final model spectrum. This model is written to a file with the form `PATH_BASE[_orderXXX].robospect`.

This file contains a comment block at the beginning of the file listing the command line used to generate the file and the column headers. The columns returned are:

Column	Value	Description
1	wavelength	Input spectrum sample wavelength
2	flux	Input spectrum sample flux
3	err_est	Calculated noise value at this sample
4	continuum	Calculated continuum value at this sample
5	lineflux	Calculated line flux (without continuum model) at this sample
6	model_spect	Complete model (continuum + lines) spectrum
7	fitflux	Internal "fit flux" value used while fitting lines

4.2.3 Line fit plots

This postscript file contains plots of either linelist specified or all lines (with the `-A` option), if plplot plotting support was compiled into ROBOSPECT. These plots are written to a file with the form `PATH_BASE[_orderXXX].robo.ps`.

4.2.4 Log file

This file contains all output from the verbose logging. Please include this file in any bug reports or code queries. This log is written to a file with the form `PATH_BASE.robo.log`.

Chapter 5

Command Line Options

The command line options for ROBOSPECT can be examined using the `-h` option. In addition to listing the available options, this option also displays the current values for each parameter (whether derived from the default settings, or specified by the user on the command line).

The following list reviews the help block (as of v2.10), along with the default options. A variety of parameters are listed as either deprecated or experimental. Use of these options is not recommended, and will either be removed or improved in the future.

```
Usage: ./src/robospect <options> <input.spectrum>
```

An input spectrum must be specified on the command line. The format of this spectrum is detailed above.

```
 -h, --help                    This help                          (found)
```

5.1 Fitting Parameters

```
                        FITTING PARAMETERS
```

These options control the behavior of the fitting algorithms, and should generally be fixed for spectra of similar resolution and quality.

```
-V, --continuum_box <N_AA>     Width of box used for continuum
                                and noise estimates.            (40.000000)
```

For the default boxcar continuum model, a box size is necessary to filter the spectrum. This value specifies the width of that box in Angstroms (see Input Files above for notes on units). This value may be automatically changed by the continuum modeling code if this value is either too small (the number of samples in the box drops below 5) or too large (the number of samples includes a significant fraction of the spectrum size). Ideally, this number should be larger than any spectral feature, but smaller than any known continuum error features (such as poorly connected echelle orders).

```
-r, --deblend_radius <N_sig>   Number of a line's stdev to look
                                for possibly blended lines.     (4.000000)
```

Deprecated option.

The standard deblending code assigns lines into blend groups if their peaks are close relative to the line widths (using the Gaussian sigma value for the lines). This value defines how "close" two lines must be before they are assigned to the same group. This radius is applied to the sum of the line widths, such that two lines are grouped if abs(x1 - x2) ¡ radius * (sigma1 - sigma2).

```
-R, --deblend_ratio <F>        Fraction of a line's peak to bother
                                with possibly blended lines.       (0.100000)
```

Deprecated option.

```
-d, --deblend_iterations <N>   Number of times to attempt to find
                                neighboring lines.                 (3)
```

Deprecated option.

```
-T, --tolerance <eps>          Tolerance value for fitting.        (0.001)
```

Set the fitting tolerance used to determine when a line solution has converged. After a given line fitting iteration, each line parameter is defined to have converged if:

$$\mathrm{abs}\left((P_{i-1} - P_i)/P_{i-1}\right) < tolerance \tag{5.1}$$

Similarly, the χ^2 solution is also checked, and if the χ^2 improvement is small:

$$\mathrm{abs}\left((chi_{i-1} - chi_i)/chi_{i-1}\right) < tolerance \tag{5.2}$$

then the fit has converged and no further iterations are performed.

5.2 Model Selection

```
MODEL SELECTION
```

These options control which modules to use to fit the lines, continuum, and noise models. Many of these modules are either deprecated or still in development, and therefore may not supply a useful fit. Additional models can be added with minimal coding and effort, as discussed below (ADDING MODULES).

```
-C, --continuum_model <NAME>   Select continuum model to use:       (boxcar)
```

Select the continuum model to use. Note that the continuum is calculated with the line solution from previous iterations subtracted, which should result in low contamination of the continuum model by strong lines.

```
boxcar (default)            Simple boxcar model
logboxcar                   log10 boxcar model
```

Use a simple median boxcar filtering method to calculate the continuum level. The logboxcar module first takes the logarithm of the spectrum flux, which improves the model in the case of large gradients in the continuum. These modules are the most robust for the spectra that ROBOSPECT has been tested with.

```
peak                        Fit boxcar across maxima
```

Deprecated. Attempts to place a fit along the "peaks" in the spectrum, in an attempt to closer match hand measurements in low S/N spectra. This method often over-estimates the continuum level, resulting in larger-than-expected equivalent width values.

```
blackbody                   Fit blackbody curve to continuum
powerlaw                    Fit powerlaw to continuum
```

Fit either a blackbody or a powerlaw function to the spectrum.

```
histogram                    Fit Gaussian to data histogram
```

Deprecated. Use a boxcar as above, but fit a Gaussian to a histogram of the data to construct a more robust estimate of the median. Due to the subtraction of lines before the continuum is modeled, this module and the boxcar module yield very similar results, with the boxcar module executing far faster.

```
null                         Do no fit, assume continuum = 1.0
```

Assume that no continuum fitting is necessary, and set the continuum to a fixed value of 1.0. Unuseful for scientific results, but very useful in debugging ROBOSPECT.

```
fft                          Use FFT frequency-space filtering
devel                        Development continuum model
bspline                      Devel: better with discontinuities
```

Development modules that are not likely to work.

```
-N, --noise_model <NAME>       Select noise model to use:          (boxcar)
```

Select the module to use in estimating the spectrum noise. Some continuum modules above will set the noise while calculating the continuum, but this option can be used to select a different calculation method.

```
boxcar (default)             Simple boxcar model
```

Calculate the noise based on the median absolute deviation value in a box.

```
Poisson                      Poissonian noise model
```

Development module. Calculate the noise in a spectrum sample as the sqrt(flux). This has not been robustly tested in fitting, and may not provide acceptable results.

```
null                         Do no fit, inherit from continuum fit.
```

If the continuum module supplies a noise model, this can be used to skip extra execution of the noise code. In general, ROBOSPECT correctly determines when the noise code can be safely skipped.

```
-M, --line_model <NAME>        Select line model to use:           (best)
```

Select the line fitting module to use. The majority of these modules are deprecated, and so only the three currently being developed are noted here.

```
null                         Do no fit, assume lines = 0.0
```

Skip the line fitting stage entirely. This is useful for debugging the continuum and noise modules.

```
pre                          Estimate lines from FWHM
```

Do a quick preliminary fit by assigning the line center to the centroid of the data around the line peak. This center is then used to estimate the FWHM value of the line, and that width is then used to construct a line flux (and equivalent width) based on the assumption that the line is a simple Gaussian. This is a very fast way to fit the lines, but it does not handle blended lines well, nor does it provide the most accurate fits.

```
best (default)               Pre + automatic deblending (best)
```

Do the preliminary fit as described above, assign lines to blend groups, and then perform a non-linear least squares minimization to each blend group to find the best fitting values for all line parameters. This is much slower than the simple "pre" fit, but provides the accurate results that are discussed in Waters & Hollek (2013).

```
gauss                    Simple NLLS Gaussian fitter
fixed                    Simple NLLS Gaussian, fixed mean
nofit                    Do not fit lines
twostage                 Pre + Gauss
twosticky                Pre + Gauss + fixed
nonparametric            Estimate lines from non-parametric model
```

Deprecated and experimental modules that should not be used for scientific results. See below for more information on the non-parametric model.

```
-Q, --function_model <NAME>    Select line function to use:        (Gaussian)
```

Select the functional form for the lines to be fit.

```
Gaussian                 Gaussian model.
```

All lines are fit as simple Gaussians.

```
Lorentzian               Lorentzian model.
```

All lines are fit as Lorentzians.

```
Hjerting                 Voigt approximation.
Humlicek                 Voigt approximation.
```

All lines are fit using an approximation to the Voigt profile. This uses the definition of eta such that $\eta = \gamma_{Lorentz}/\sigma_{Gauss}$. The Hjerting approximation is only valid for $\eta < \sqrt{0.5}$, and should in general be replaced with the Humlicek algorithm.

```
Nonparametric            Non-parametric model (experimental).
```

All lines are fit using a non-parametric estimate of the line shape. This estimate starts from the peak value of the line, and constructs a profile under the two assumptions that the line is symmetric and that it monotonically approaches the continuum level. This can be used to quickly estimate the total flux of wide non-Gaussian shapes, but can fail to properly fit wide wings that are obscured by noise.

```
SkewGauss                Skew-Gaussian model (experimental).
```

All lines are fit using a skew-normal function. The eta parameter is used to hold the degree of skewness. This value can be converted to the standard third moment definition using the formula:

$$\gamma = \frac{4-\pi}{2} \frac{\eta^3}{\left(\frac{\pi}{2} + \eta^2\left(\frac{\pi}{2} - 1\right)\right)^{1.5}} \tag{5.3}$$

```
-D, --deblend_model <NAME>     Select deblending model to use:      (null)
      null (default)              Do not deblend lines.
      nlls                        NLLS Gaussian method.
```

Deprecated. Independent deblending provides worse results than the simultaneous fitting provided by the `-L best` module described above.

5.3 Line Identification

```
                    LINE IDENTIFICATION
```

Control the way ROBOSPECT handles line catalogs. One of `-L` or `-F` must be specified to provide ROBOSPECT with a set of lines to fit (using both `-L` and `-F` is also allowed and generally provides the best results).

```
-L, --line_list <file>          Fit the lines specified in this file ()
```

Supply a list of line centers from a file (see INPUT FILES above for formats). These lines are considered a priority, and have plots constructed for the line fits if plplot support has been compiled into the code.

```
-F, --naive_find_lines          Attempt to find additional lines to
                                 fit. Uses sigma threshold.          ()
```

Identify new lines from the data that are above a given noise threshold. The S/N value for these lines are listed in the output line catalog in the comment field (see OUTPUT FILES below). This identification is "naive" in that it does not do a matched filter search using a PSF to maximize the identification of lines.

```
-f, --find_sigma <s>            Set the threshold for line finding.  (3.000000)
```

Set the signal-to-noise threshold for the naive line finding algorithm.

```
    --wavelength_min_error <dw>Set the limits for acceptable error
                                in the peak finding code.           (0.100000)
    --wavelength_max_error <dw>Set the limits for acceptable error
                                in the peak finding code.           (0.500000)
    --wavelength_limit <N>      Set the limit for the number of lines
                                with bad wavelengths before the
                                code attempts to correct.           (2000)
```

Attempt to correct the input line list to match the spectrum in the case where the spectrum wavelength solution is faulty. This crossmatches all lines in the input line list with the peaks identified in the spectrum. If more than `wavelength_limit` lines have a peak that is nearby ($\delta\lambda <$ `wavelength_max_error`), but an offset greater than a threshold ($\delta\lambda >$ `wavelength_min_error`), then a wavelength solution correction is calculated as a linear trend between the line list and the spectrum. This trend is then used to modify the expected line centers in the line list, to avoid issues interpolating the spectrum onto a new grid.

In most cases, if you didn't follow this discussion, don't worry, you probably don't need it. ROBOSPECT is not designed to do a full calibration of the input spectrum, so it's best to have the wavelength solution correct, and then not bother using these options.

```
    --radial_velocity <v>       Apply a known radial velocity
                                 correction to wavelengths in units
                                 of (wavelength_unit * 1e13)/sec.   (0.000000)
```

Modify the line list to match a spectrum with a known radial velocity offset. Assuming the input spectrum has samples measured in Angstroms, this option needs to be supplied the radial velocities in kilometers/second.

```
    --measure_radial_velocity  Attempt to match lines to linelist to
                                measure the radial velocity.        (0)
```

A radial velocity correction can be calculated from the input spectrum and the supplied line list. Briefly, the algorithm identifies a set of strong lines (10-σ), and determines an average width and flux. These values are used to construct two fake spectra: the first with Gaussians at the line centers from the set of strong lines with the average widths and fluxes; the second using the same line shapes but with the line centers from the input line list. For a set of radial velocity steps from the minimum to maximum values, the linelist lines are transformed, and the cross-correlation of these two spectra is calculated. The peak of this correlation spectra is then identified and the process repeated until the error in the radial velocity is smaller than the specified value. This best radial velocity is then used to transform the input line list centers to match the observed line centers in the data.

As in the user specified radial velocity correction, the velocity units are in km/s if the input spectrum has wavelength units of Angstroms.

```
    --radial_velocity_range    Search for radial velocities +/- this
                                value.                          (300.000000)
```

Set the radial velocity range around 0 km/s to search.

```
    --radial_velocity_error    Set the maximum rv error allowed.    (0.010000)
```

Repeat cross-correlation search until the uncertainty in the radial velocity is smaller than this value.

```
    --radial_velocity_step     Set the number of correlation steps. (100)
```

Number of steps to use to search the radial velocity range. With the default settings, this sets the steps at 6 km/s in width, producing a convergent radial velocity in two iterations.

5.4 Program Control

```
                        PROGRAM CONTROL
```

These options control the output file generation and the program decision making.

```
  -i, --iterations <i>           Number of continuum/line iterations  (1)
```

Define the number of iterations of full continuum/line fitting to perform. A single iteration will provide a decent estimate, but the line solutions greatly improve with increased iterations. The test spectra used in Waters & Hollek 2013 used 5 iterations, with no evidence if dramatic improvement beyond that number.

```
    --loosen                   Loosen line quality checks. Useful   ()
                                with flux calibrated data.
```

This option allows the line flagging checks to be slightly less aggressive at removing questionable lines based on the ratio of the FWHM to the line flux. This generally isn't required for absorption spectra, as the FWHM/flux ratios are more constrained than those found in emission spectra. This option is recommended for use with flux-calibrated data.

```
    --flux_calibrated          Input is in flux units(erg/s/cm^2/A) (0)
```

This option should be used if the input spectra are flux calibrated instead of continuum normalized.

```
    --fits_row <N>             Fit row N of 2-d fits spectrum.      (0)
```

For FITS input spectra with multiple orders assigned to different rows of a single 2-d image, specify only a single row/order to fit.

```
-I, --save_temp                Save intermediate results           ()
```

Save the line catalog and spectra models for all iterations instead of saving only the final result.

```
-A, --plot_all                 Default plot only contains lines
                                from the line list. This option
                                allows found lines in the plot.    ()
```

If plplot plotting support is compiled in, construct plots for all lines in the catalog, not just the lines specified in the input line list.

```
-P, --path_base <output_root>  Output file rootname.               ()
```

Define an path for output products. This is highly recommended, as it should prevent any accidental overwriting of already existing data products. See OUTPUT FILES for more details.

```
-v, --verbose <verbose_level>  Change verbosity level:             (0x0001)
                                A logfile is used for most levels.
      silent                     No messages              ()
      default                    Normal messages          (found)
      io                         File IO                  ()
      id                         Line identification      ()
      line                       Line fitting             ()
      continuum                  Continuum fitting        ()
      noise                      Noise fitting            ()
      math                       Math operations          ()
      screen                     Print messages to screen ()
      debug                      Code debug messages      ()
      all                        Enable all messages      ()
```

Enable logging of various components. Enabling line and screen facilities (“-v line -v screen”) provides a useful peek at code operation (i.e., running ROBOSPECT produces lots of output to the screen, making that terminal look busy).

Chapter 6

Calculations

Taken from Waters & Hollek (2013):

6.0.1 Spectrum Models

Continuum Normalized Spectra

ROBOSPECT models spectra based on the assumption that all spectra are comprised of three components: the continuum level, $C(\lambda)$, the line solution relative to that continuum $L(\lambda)$, and an error component $E(\lambda)$ that contains the deviation between the true spectrum and the current model. By iterating the fitting of these components, we can ensure that the line and continuum solutions are not biased by the other. These components are combined in different ways to measure line strengths, depending on the type of spectrum to be fit. For a continuum normalized spectrum (where the mean input continuum is nearly unity), we assume that any low spatial frequency deviations from unity are due to a poor normalization correction, and high spatial frequency deviations are a combination of unknown and unfit lines and the $E(\lambda)$ component. In this case, we construct a model of the final spectrum,$S(\lambda)$, as:

$$S_{continuum\ normalized}(\lambda) = C(\lambda) * (L(\lambda) + 1.0 + E(\lambda)) \tag{6.1}$$

This relation ensures that in a spectrum with the continuum normalization perfectly performed, the line model and noise add directly to modulate the spectrum around unity. The measured continuum is assumed to represent imperfect normalization, and so is multiplied to the expected final spectrum. Figure 6.1 demonstrates an example of how these components combine to form the observed spectrum.

Using this spectrum model, we can solve Equation 1 for the continuum component, by letting the input spectrum be the expected final spectrum solution. We can then remove the lines from the data, based on previous iterations of the fitting routine. For the initial iteration, we have no knowledge of the correct line solution, and so expect the continuum level to be slightly biased by strong lines. However, as further iterations are performed, this bias decreases, improving the continuum solution. We ignore the error component when measuring the continuum. As the errors should cause deviations in both directions from the true continuum, they should therefore not bias the result. By subtracting the current line model from the input spectrum, we assume that we are simply left with the continuum structure (modulated by the unknown error component). For a continuum normalized spectrum, we must first rescale the individual line measurements to match the possibly imperfect input spectrum continuum normalization:

$$C_{continuum\ normalized}(\lambda) \quad = \quad f_C\left(S_{input}(\lambda) - C(\lambda) * L(\lambda)\right) \tag{6.2}$$

The function f_C denotes some smoothing operation that returns the continuum model, given the input of a spectrum with all lines removed. ROBOSPECT defaults to a simple median boxcar filter for f_C, which can

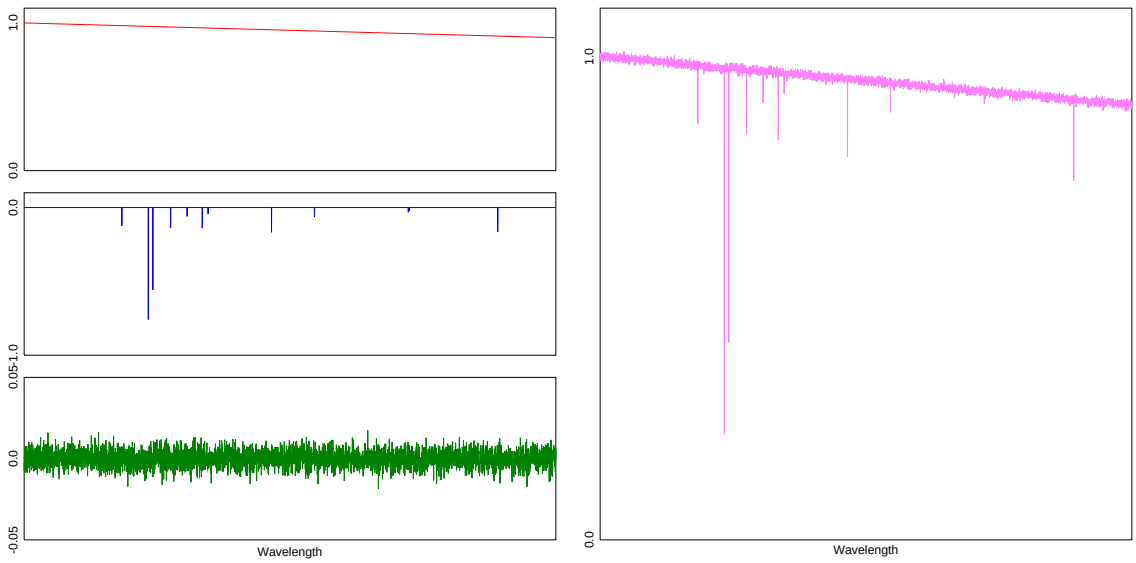

Figure 6.1: Simple diagram for the individual components that are used to make up the observed spectrum. The left panels show the continuum (red), line (blue), and noise (green) models. The right panel contains the expected observed spectrum, which is the product of the continuum model and the sum of the line and noise components.

generally fit continuum models that are smooth and do not have discontinuities. Other continuum smoothing functions are supported, including a spline-based method that can be used when the spectrum does have known discontinuities, such as those due to the combination of echelle orders into a single spectrum. Functional models, such as a blackbody or power-law model, can also be chosen for f_C, in which a known analytic function is fit to the line-subtracted spectrum.

With the continuum modeled, we can start to investigate the effect that the unknown $E(\lambda)$ has on the data. This error is not, in general, drawn from the per-pixel noise value. As ROBOSPECT only attempts fits for lines in the linelist, any unknown line will create a deviation between the best model spectrum and the input data. Because of this, ROBOSPECT estimates a noise for each pixel that represents the observed scatter in the spectrum, which includes the contribution from small, unfit lines. This measured noise is generally higher than the expected Poissonian noise, due to this extra contribution. This noise is used both in the line fitting stage to determine how well the model matches as well as in the internal line finding stage. Similar to the continuum solution above, we find a noise solution, $E(\lambda)$:

$$E_{continuum\ normalized}(\lambda) \quad = \quad f_E\left((S_{input}(\lambda)/C(\lambda)) - L(\lambda) - 1.0\right) \tag{6.3}$$

This solution removes the current best fitting continuum and line models from the data, leaving only what should be considered the error component. As above in Equation 2, the function f_E attempts to model the underlying noise that this error component is drawn from. This is also generally a filtering process, with the default filter using the median absolute deviation statistic ($MAD = \mathrm{median}(\mathrm{abs}(x_i - \mathrm{median}(x_i)))$). This value is then converted to an equivalent Gaussian σ by noting that for a Gaussian distribution, the MAD is half of the interquartile distance, and therefore this equivalent σ is equal to $1.4826MAD$. Using this statistic instead of a directly measured standard deviation allows this estimate of the noise to be robust against the influence of outliers, such as those that may be left behind by large unmodeled spectral features.

At this point in the fitting process, ROBOSPECT can attempt to identify potential lines that are not yet in its linelist. This is done by looking for local peaks that are more than a fixed user-specified number of σ

deviant from the continuum subtracted level:

$$Z_{continuum\ normalized}(\lambda) \quad = \quad \left(\left(S_{input}(\lambda)/C(\lambda)\right) - 1.0\right)/E(\lambda) \tag{6.4}$$

We expect that a large fraction of lines specified by the linelist will be significant in this measurement. Therefore, after finding all the local peaks that are above the threshold, we merge this list of peaks with those already in the linelist and remove the duplicates. The current line model is not subtracted from the data before looking for new lines, as the model residual can introduce significant local peaks that are not necessarily real.

ROBOSPECT constructs a line solution model using the same methods as when it finds lines, except without normalizing by the spectrum noise:

$$L_{continuum\ normalized}(\lambda) \quad = \quad f_L\left(\left(S_{input}(\lambda)/C(\lambda)\right) - 1.0\right) \tag{6.5}$$

Again, f_L is a function that represents the fitting process performed for each line in the current list (whether supplied by the user or found automatically). This fitting process can use various methods, and can use different models for the line profile shape, as described in Section 6.0.2.

Flux Calibrated Spectra

For flux calibrated spectra, the same three components are used to describe the input spectrum. However, these components are combined in a different manner than is used for the continuum normalized spectra. For flux calibrated spectra, we assume that the continuum solution combines with the line solution additively and not multiplicatively as in the case of the continuum normalized spectra. Therefore, the spectrum model for flux calibrated data is:

$$S_{flux\ calibrated}(\lambda) = C(\lambda) + L(\lambda) + E(\lambda) \tag{6.6}$$

From this model, we can derive relations for the various components as was done above for the flux normalized spectra.

$$C_{flux\ calibrated}(\lambda) \quad = \quad f_C\left(S_{input}(\lambda) - L(\lambda)\right) \tag{6.7}$$
$$E_{flux\ calibrated}(\lambda) \quad = \quad f_E\left(S_{input}(\lambda) - C(\lambda) - L(\lambda)\right) \tag{6.8}$$
$$Z_{flux\ calibrated}(\lambda) \quad = \quad \left(S_{input}(\lambda) - C(\lambda)\right)/E(\lambda) \tag{6.9}$$
$$L_{flux\ calibrated}(\lambda) \quad = \quad f_L\left(S_{input}(\lambda) - C(\lambda)\right) \tag{6.10}$$

The same fitting and estimation algorithms can be used on both kinds of spectra, as we isolate the component being measured in both cases. Due to the large dynamic range possible for flux calibrated spectra, it is useful to calculate the continuum and noise models using the logarithm of the flux levels. This prevents a large continuum gradient from being dominated by the very largest values.

6.0.2 Line fitting

Figure 6.2 shows a flow chart of the steps used in fitting individual lines. Before doing more complicated fitting, an initial guess for the fit parameters is constructed for each line, under the assumption that the line profile is a Gaussian. The line center is estimated from the centroid around the line peak. The FWHM is calculated by interpolating the line profile at the half-peak value. As heavily blended lines may have excessive contribution from other lines at the half-peak value, we check this value with the width calculated from the 75%-max peak value. In the case of large differences in the expected Gaussian σ between these two values, we take the smaller value to minimize the possible effect of blending. Finally, total line flux, F, is estimated for a simple Gaussian model from the peak flux and the FWHM ($F = P\sigma\sqrt{2\pi}$). After this initial fit is performed for all lines, a set of deblending groups is determined. If the line centers are closer than a

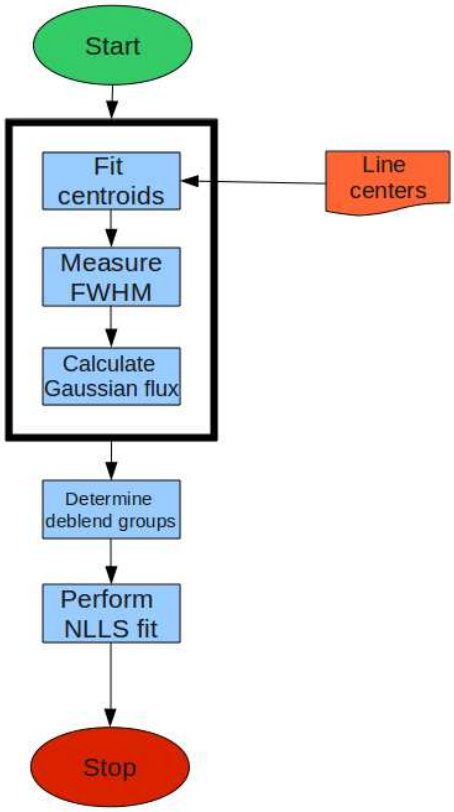

Figure 6.2: Diagram of the steps used by ROBOSPECT to fit individual lines.

given number of line widths (taken by default as six times the Gaussian model σ), then they are assigned to the same deblend group. This deblend group is then simultaneously fit as a single unit, to allow for the contributions of all the lines at a given wavelength. The full line fitting to find the best fitting set of model parameters is done via non-linear least-squares minimization of $\chi^2 = \sum_\lambda \frac{L(\lambda)-\sum_i Q_i(\lambda)}{E(\lambda)}$ for each deblending group, where $Q_i(\lambda)$ is the component of the ith line at wavelength λ. Different line model choices can made for Q, as described in Section 6.0.3. Once convergence is reached for all the lines in one deblend group, those values are recorded and the fitting moves to the next deblend group, until all lines over the entire spectrum have been fit.

After fitting all lines, a set of quality checks are performed to ensure that only valid fits are retained in the final model. First, any line that did not converge within a fixed number of fitting iterations is flagged and rejected from further consideration. Similarly, lines that have final fit values that are non-finite are flagged and rejected. Finally, the total line solution is calculated by summing only the valid lines, further rejecting any lines that do not reduce the total χ^2 value.

Once the line fitting has been finished and the line solution accumulated, further iterations can be performed starting with these models as an initial set of parameters. As the continuum model is based on the spectrum with lines removed, better line models result in a smaller line residual, allowing for a better continuum model. In turn, with a more accurate continuum model, the line fits are more reliable, with less noise as there are fewer residuals remaining. In

6.0.3 Line profile functions

The default assumption for all lines is that they have a profile shape consistent with a simple Gaussian. This is appropriate for most lines used in a typical equivalent width-based stellar abundance analysis. Lines that fall on the linear portion of the curve of growth have a 1:1 ratio between the line width and the absorbers, thus a small change in the number of absorbers is easily reflected in the line profile. However, as the line leaves the linear portion of the curve of growth, this ratio becomes smaller and the assumption of a Gaussian profile begins to break down, resulting in wings that are broader and contain a larger fraction of the total flux. To account for these strong lines, we allow the user to select other profile functions that can better model these lines.

In the extreme case of the line shape generated solely by pressure broadening, we support a Lorentzian model. This is in general not appropriate for most lines included in the typical analysis of a stellar spectrum; however the Gaussian line shape is not appropriate for certain lines from which abundances are derived (e.g., Mg b) and for lines that are important in calculating the continuum (such as the hydrogen lines). Between the extreme Lorentzian and the Gaussian is the Voigt profile, the result of the convolution of a Gaussian and Lorentzian models. This profile is somewhat difficult to directly calculate, and so we utilize the Hjerting (or Humlicek) approximation that describes the Voigt profile as a polynomial approximation in η, the ratio of the Lorentzian width to the Gaussian (Doppler) width. This function reduces to the simple Gaussian form when $\eta = 0$, allowing this function to be used on all lines, with the degree of non-Gaussianity allowed to best match the data. This function is significantly more complicated than a simple Gaussian, however, increasing the computation time required for fitting.

Chapter 7

Adding Modules

This will be more detailed in the future, but the steps for adding a new module are:

1. Create a source file in src/models/ to contain the module. Write the module code.
2. Add the module prototype to robo.h, and augment enum values to support the new module.
3. Add the module to the appropriate place in src/model.c to ensure it is called correctly.
4. Quickly document the new module in src/config.c so it appears in the program help.
5. Add new source files to Makefile.am, and rerun aclocal/automake/autoconf as needed.
6. Compile/test/submit changes for inclusion in future versions.