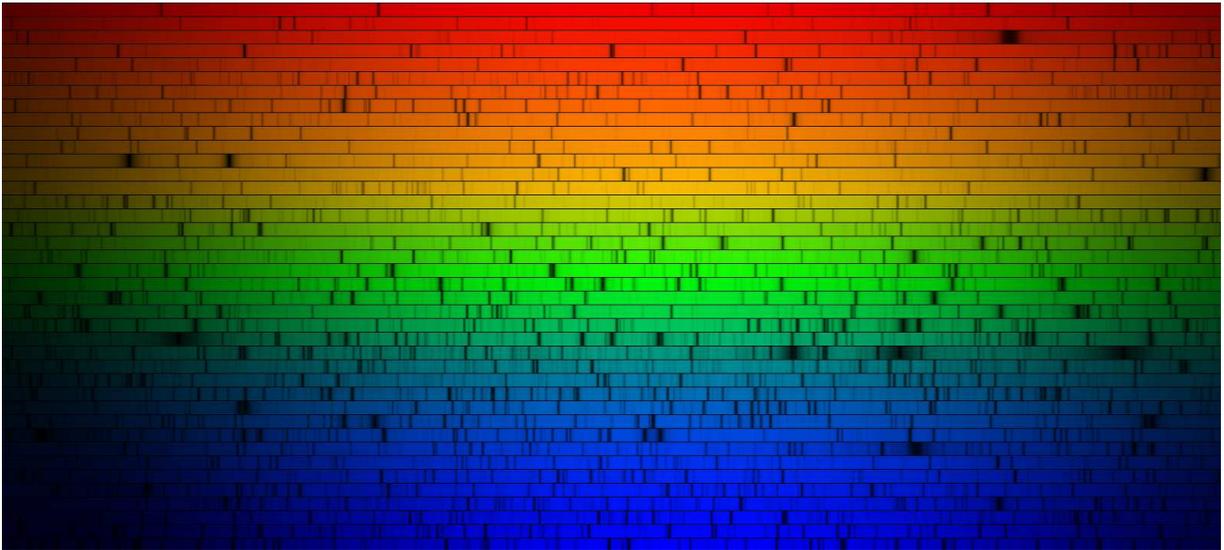


# CHIRON basic data reduction

A. Tokovinin

Version 3. October 26, 2011  
file: prj/chiron/doc/code/chireduce.tex



# 1 Overview

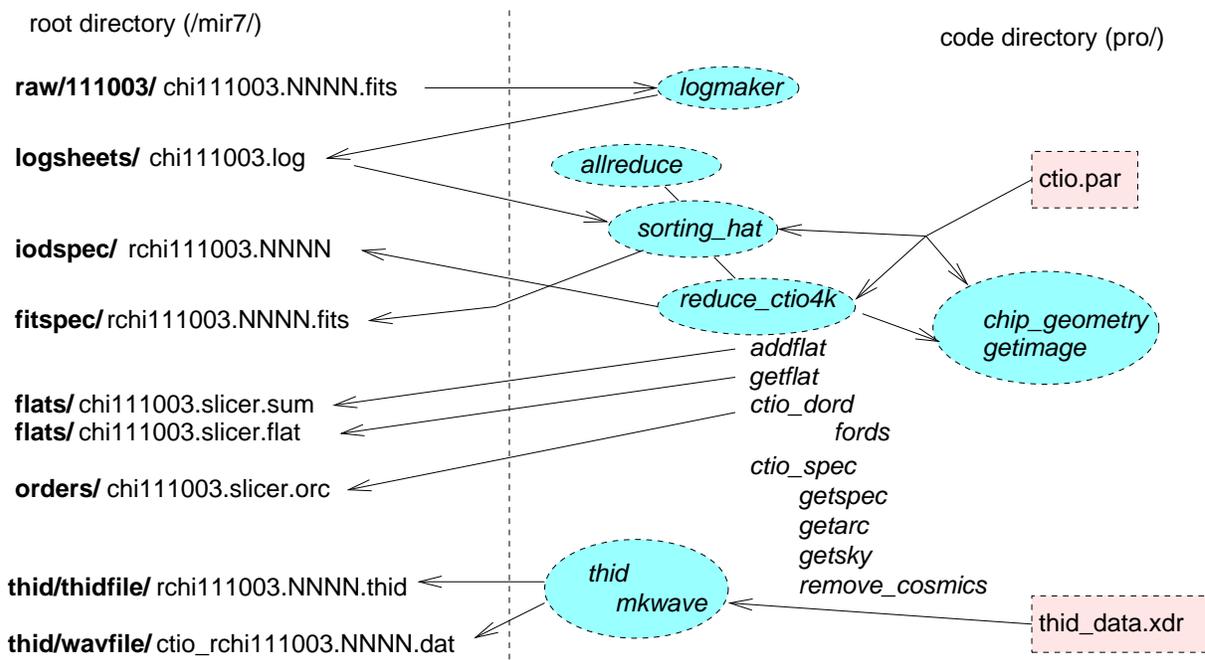


Figure 1: Directory structure of CHIRON data and reduction code.

This document describes the IDL package for reduction of echelle spectra from the CHIRON spectrograph. The package is based on the REDUCE code by Piskunov & Valenti<sup>1</sup>, but departs from their algorithm in many substantial ways. It was developed by D. Fischer and M. Giguere. Further modifications were made by A. Tokovinin in October 2011. Here we consider only basic reductions: order extraction, flat-fielding, and wavelength calibration. Measurement of precise radial velocities with iodine (the “doppler code”) is not covered.

Figure 1 gives overall view of data and programs, illustrating also the adopted naming conventions. All raw and processed files are located in the root directory `/mir7/`. Raw images (FITS files) are copied from the data-taking computer `ctioe1` at Cerro Tololo to nightly sub-directories `raw/yymmdd/` in `/mir7/raw`. In this document we consider the fictitious night of October 3, 2011 as example. The raw files are named as `chiymmdd.NNNN.fits`, where NNNN is sequential exposure number for this night.

The reduction code operates as hierarchical scripts that call other scripts or actual programs. The short top-level script is `allreduce`, it calls `sorting_hat` once per each observing mode. The scripts sort data according to the observing mode. Four modes are defined: `narrow`, `slicer`, `slit`, `fiber`. Each mode is associated with particular CCD binning, order location, and wavelength calibration. The `sorting_hat` is calling `reduce_ctio` and other routines. More details are given in Sect. 3.

<sup>1</sup>Piskunov, N.E. & Valenti, J.A. 2002, A&A, 385, 1095

## 2 Step-by-step

### 2.1 Log files

The log files, one per night, are created by `logmaker.pro` and stored in `/mir7/logsheets/`. The information is extracted from the FITS headers (change of the header keywords will require adjusting the code). Example of the logfile is given below. These files are read by the data-processing scripts using `readcol.pro`: the header is skipped, and string arrays of observation number, object, mode, etc. are produced. These arrays are used to select observations in particular mode and to match them with calibrations. The only calibrations used here are the flat fields (quartz exposures) and the thorium-argon spectra.

#### CTIO Spectrograph Observing Log

```
-----
Observer: Manuel Hernandez           Telescope: CTIO 1.5-m           Prefix: chi111003
UT Date: 2011, Oct 2/3              Chip: 201 (e2v 4k, 15micron) Foc: 10.2940 mm
Ech: CHIRON      Fixed Cross-disperser position          Foc FWHM: 2.394
-----
```

Obs number	Object Name	I2 (y/n)	Mid-Time (UT)	Exp time	Bin	Slit	PropID	Hdr Comments
0011	thar	y	20:08:20	1.00	3x1	slicer		Calib11
0012	thar	n	20:11:12	1.00	3x1	narrow		Calib
0016	iodine	y	20:13:26	2.00	3x1	slicer		Calib
0018-0032	quartz	n	20:18:32	2.00	3x1	narrow		Calib
....								
1201	177565	y	23:36:31	1200	3x1	slit		CPS
1202	177565	y	23:56:44	1200	3x1	slit		CPS
1203	177565	y	00:16:57	1200	3x1	slit		CPS

### 2.2 The parameter file

Different parts of the reduction code need to “know” the location of relevant files and other common parameters. This information is passed in the structure `redpar`. It is defined by the parameter file `ctio.par` and can be read as `>redpar = readpar('ctio.par')`. The values of the parameters can be changed by editing the `.par` file. New parameters can be added, but in this case we need to exit and re-enter the IDL because its *named* structure definition is “rigid”. On the other hand, it is passed to programs by reference and can return useful information. Fields can thus change during data processing are marked `vy (8)` in the comments, they are `imdir`, `prefix`, `binning`, `mode`, `gain`, `ron`. These changes are not saved on the disk. Comments or commented lines are allowed. However, the syntaxaxis (colons, commas after values) must be strictly observed.

```
{redpar,                ; named structure, passed by reference
rootdir: '/mir7/',      ; root directory. All other paths are relative to rootdir
logdir:  'logsheets/',  ; log sheets
```

```

iodspecdir: 'iodspec/',           ; reduced spectra in RDSK/WRDSK format
fitsdir:   'fitspec/',           ; reduced spectra in FITS format
thiddir:   'thid/wavfile/',      ; wav saved in in *.dat files, not used
thidfiledir: 'thid/thidfile/',  ; thid saved in *.thid files
rawdir:    'raw/',              ; raw files
imdir:     '111003/',           ; yymmdd/ image directory with raw night data (*)
prefix:    'chi111003.',        ; file prefix, will be set by sorting_hat  (*)
flatdir:   'flats/',           ; summed flat fields
orderdir:  'orders/',          ; order locations
barydir:   'bary/',            ; code for barycentric correction?
xtrim: [723,3150],             ; trim along line (cross-dispersion direction), UNBINNED pixels
ytrim: [611,3810],             ; vertical trim (along disp.), UNBINNED pixels 4111 - [301,3500]
readmodes: ['fast','normal'],  ; readout modes
nlc: [[5.0e-6, 4.3e-6],[4.5e-6, 4.0e-6]], ;non-linearity coefs. [left,right] in fast and normal
gains: [[2.59,2.19], [2.23, 1.92]], ; gain [ [l,r]fast, [l,r]norm], el/ADU
ron:      2.,                  ; RON estimate [ADU], to be calculated from bias (*)
gain:     1.,                  ; actual gain [el/adu] (*)
binning:  [1,1],               ; will contain actual binning [row,col] from the header (*)
mode:     0,                    ; index of the actual mode (*)
pkcoefs: [38.0,43.890,0.2310,0.002029], ; poly coefs of peak maxima @center y(iord), unbinned
nords:    40,                  ; number of orders to extract
modes:    ['narrow','slicer','slit','fiber'], ; observing modes
xwids:    [6,12,6,5],          ; extraction width, binned pixels
dpks:     [10,7,10,9],         ; peak shift for each mode, binned pixels
binnings: ['3x1','3x1','3x1','4x4'], ; binning in each mode, row x column
debug:    0}                    ; 1 in debug mode, with plots and stops

```

## 2.3 Reading the raw files

The raw images contain pre- and over-scan sections for bias subtraction. Presently the readout is made with two amplifiers, but we plan to switch to 4 amplifiers. All these instrument-specific details are isolated in two routines, `chip_geometry.pro` and `getimage.pro`. The first returns the geometry structure describing readout mode, binning, etc. It is used by `getimage` for bias subtraction. The image segments read through different amplifiers are stitched together and returned by `getimage` as a single image file. The header can be returned as well in the keyword. The `redpar` is passed as input parameter. Presently, the images are trimmed in rows and columns, and this trimming is specified in the parameter file. Independently of the binning, the same part of the CCD is always selected.

The actual binning of the image is “saved” in `redpar.binning`. The readout noise (in ADU) is calculated from the overscan and saved in `redpar.ron`. for further use. Typically it is 2.3ADU in the normal readout mode (fiber) and 3.1ADU in the fast readout mode. The right half of the image is scaled by the ratio of the amplifier gains to reduce the “step” between the two halves (the left gain then applies to the whole image and is saved in `redpar.gain`). Non-linearity of the CCD controller is corrected. The image reading routine is used in different parts of the code (flats, order definition, extraction). The images are returned without rotation, but in further reductions they are rotated to

orient dispersion along the rows by the command `im = rotate(im, 1)`.

## 2.4 Adding flat fields

First thing done by `reduce_ctio` is the co-addition of the flat fields. The list of flats is passed as array of image numbers, then `addflat` is called. Each file is read and processed by `getimage`, the sum of flats is saved in the `/mir7/flats` directory as `prefix.mode.sum`, where the prefix is `chi111003` for our example. The summed quartz frame is used in the script `reduce_ctio` directly (no need to read it back from the disk). However, when no flat-field is given on input, `reduce_ctio` tries to read the summed flat for particular night and mode from the disk.

## 2.5 Order location

The second step is to define the order centers by polynomials `orc[ncoef,nord]`. For each order the coefficients describe the order location as  $y(x)$ , where  $x$  is the horizontal pixel coordinate (along row),  $y$  is the vertical coordinate (along column). Incomplete orders that cross lower or upper image boundaries are discarded. Obviously, the order definition depends on binning and image truncation.

In the present version of the code the 40 orders are “hard-coded”. The edge orders are excluded “by construction”, so that none of the good orders crosses the image boundary. The orders are defined as positions of the order centers in the middle of the CCD, approximated by the cubic function of the order number  $y(iord)$  (quadratic is not good enough). The coefficients are given in the `redpar.pkcoefs`, in un-binned pixels. The initial peaks are shifted differently for each mode using `redpar.dpks`. The wrapper script `ctio_dord` reads the order-definition image (or uses the summed quartz exposure) and calls `fords`.

Series of cross-order image cuts (swaths) are used for the order location. Each swath is slightly smoothed by median filtering, eliminating the cosmic rays (excessive smoothing is detrimental for the order location). Given the initial (default) peak positions, the algorithm finds actual peak centers by parabolic approximation of each peak in the (smoothed) swath. It proceeds from the chip center (blaze peak) to the right and to the left, re-adjusting the peaks in each swath successively (following the orders). The 2-D array of peak positions is then approximated by 4th order polynomials (as much as 0.25 fraction of peaks may be missing). The array `orc` is saved in `/mir7/orders/prefix.mode.orc` file, but is not normally needed, being stored in the variable by the script.

Until now, the order positions were re-defined for each spectrum. This caused some wrong reductions for faint stars. In the current version the orders for each mode and each night are fixed, determined only once from the well-exposed spectrum or summed quartz. The image truncation was adjusted so that order 0 is not eliminated for the sliced frames. This order is located always at the same unbinned pixel of the central swath in all modes.

In the debug mode (`redpar.debug=1`) the program stops at several key points. First, the central swath and default peaks are plotted. Then the polynomial coefficients for the central swath are printed, and finally the grey-scale image with order centered marked with white dots is shown. After pressing `.C`, the “slit function” of the central order is plotted (Fig. 2). Order location is particularly delicate in the slicer mode, when the peaks nearly touch. Nevertheless, the rms error of peaks w.r.t. their polynomial description is about 0.06 pixels in all modes.

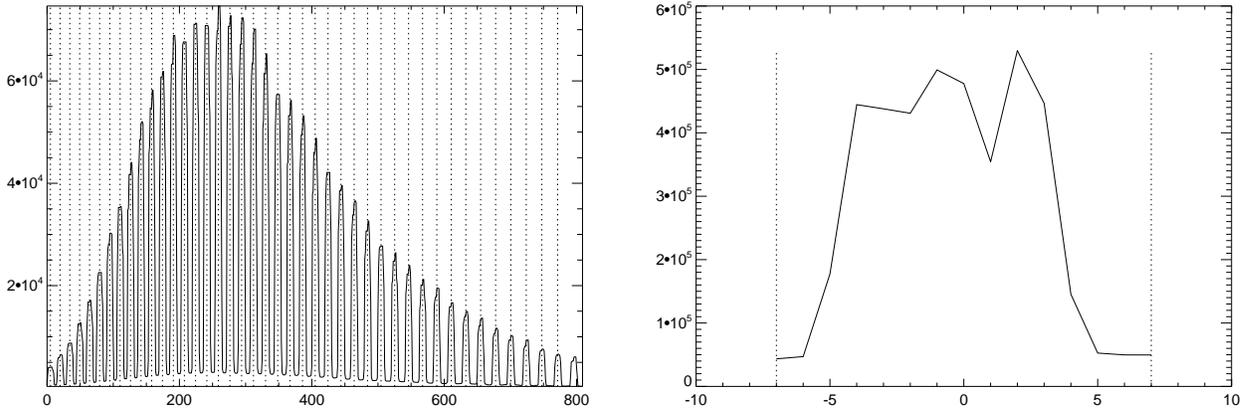


Figure 2: Diagnostic plots for order location in the slicer mode. Left: smoothed central swath with initial peak positions. Right: slit function, with vertical lines indicating the range of peak approximation by parabola.

## 2.6 Spectrum extraction

The extraction is done by the script `ctio_spec`:

```
CTIO_spec, prefix, spfname, outfname, redpar, orc, xwid=xwid, flat=flat, nosky=nosky, cosmics=cosmics
```

It reads the image file and uses the order coefficients given on input. Then the scattered light between the orders is subtracted by `getsy`. It samples the image between the orders, adjusts smooth 2D polynomial and subtracts this contribution from the image. In the current spectra the scattered light reaches 1.4% of the peak signal in all modes except slicer, where the scattered light is larger.

The spectra are extracted by boxcar average along each order on `xwid` pixels. This parameter is pre-set for each mode in `redpar.xwids` and passed to the extraction routine. The procedure `getxwd` can be called for automatic calculation of the extraction width, but it is not used at present. The extraction program `getspec` proceeds order-by-order, cleaning the cosmic rays in the process (some cosmic events remain uncorrected). It calls `getarc`, `mkslitf` and `optordv`. These routines need to “know” the gain and readout noise, which is taken from `redpar` and passed further as keyword by `ctio_dord`. The result is an array of `[npix, nord]` size containing summed counts across order for each pixel. Optimally extracted spectrum is a by-product of the cosmic-ray routine (see further discussion in Appendix A).

The extracted spectra (and FITS headers) are saved in the directory `/mir7/iodspec/` in binary format using the custom routine `wdsk`. They can be read back with `rdsk`:

```
> rdsk, sp, '/mir7/iodspec/rchi111003.NNN', 1
```

puts the spectrum in the array `sp`. To read the header, use 2 instead of 1. A better way would be to use the standard IDL `save/restore` functions, as done during wavelength calibration.

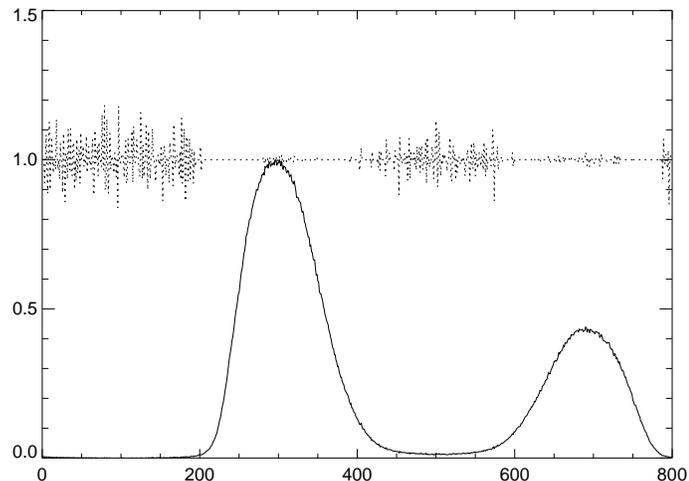


Figure 3: Trace of the quartz exposure row (full line) and the flat field correction derived for this row (dotted line). The correction is noisy between the orders and is exactly one on the high-slope parts.

## 2.7 Flat field correction

In the previous version of the code, the flat-field (FF) correction was done by two-dimensional division of the image. This procedure, used for Keck spectra reduction, works well when the flat-field spectrum is wider than the stellar spectrum (hence the term “wide flats”). This is not the case for CHIRON. The algorithm divided each line of the FF frame by its 5-pixel median to remove low-frequency components (large-scale CCD defects are thus uncorrected). The result is noisy between the orders, where the signal is low. When the line crosses the order, the signal has a large slope and its 5-pixel median equals the value in the central pixel. Therefore, the division produced 1 and there is no FF correction. This is illustrated in Fig. 3.

The FF algorithm was changed to correct the extracted spectra, rather than 2D images. The FF is extracted, normalized by the fitted polynomial (along the order, not along the line), and used to correct the extracted spectra. The resulting flat-field array of `[ncol,nord,3]` size contains the FF correction in the first plane, the extracted quartz in the second plane, and smoothed orders in the third plane. It can be used to correct spectra for the blaze function of the echelle. The array is saved on disk in `/mir7/flats/chi111003.mode.flat`, but not used normally (the script `reduce_ctio` contains it in a variable).

Figure 4 plots three consecutive orders (20, 21, 22) of the extracted and flat-fielded quartz spectrum. The smoothness of the blaze function characterizes the quality of the extraction. A sharp negative feature is seen in the order 22,  $\sim 5$  pixel wide. Such features are present in some other orders and are caused by the large contaminant particles on the CCD.

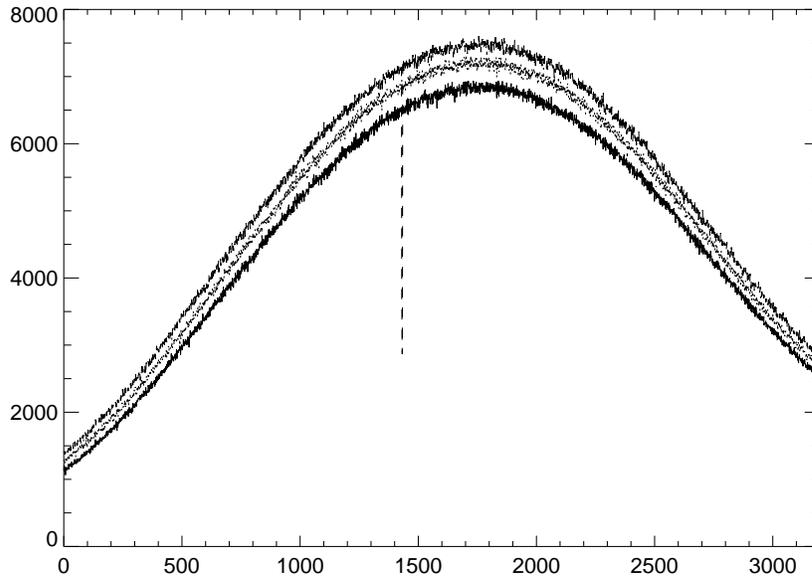


Figure 4: Three orders 20-22 of the quartz exposure after extraction and flat-fielding. File chi111003.0034 (quartz, slit).

## 2.8 Wavelength calibration

The wavelength calibration in ThAr spectra is done after extraction using the program `thid.pro` or its automated version `auto_thid`. It is called by the script `sorting_hat`. To do it manually, we issue the following commands:

```
> restore, '/mir7/thid/thidfile/rqa39.5969.thid' ; thid structre, wvc are the coefs
> rdsk, t, /mir6/iodspec/rqa39.9407.thid' ; [801,40]
> thid, t, 88., 88.*[6342.,6466], wvc, thid, init=thid.wvc, /orev
```

Here the previous line identification is restored first: the structure `thid`. The new extracted ThAr spectrum is read from the disk into array `t`, and then the interactive procedure `thid` is called. At its prompt, type `h` to see the command options. In the very least, you would type `a` to mark the lines automatically (accept three default parameters offered), `f` to fit the wavelength solution (enter polynomial degrees in `x` and `y`, typically 6), and `q` or `x` to quit. The result will be in the array `wvc` and in the structure `thid`.

Previously identified spectra used as initial guess may have different number of pixels because of the binning. This is taken care of by re-sampling the initial wavelength to match the actual ThAr spectrum (the order mismatch would be fatal, however). In the debug mode `auto_thid` displays the residual plots and the 2D spectrum with marked lines. Identification is either correct or totally wrong! The typical rms of “good” lines after outlier rejection is 0.15 pixels.

The calling script saves `thid` in the directory `/mir7/thid/thidfile/`. Further, it used `mkwave` to create the wavelength array for each order of `[npix,nord]` size. It is saved in the directory

`/mir7/thid/wavfile/ctio*.dat` after being flipped vertically (wavelength increases up, to match the spectra that are also flipped before saving). However, in the latest version of `sorting_hat` the `.dat` files are not used, being produced “on the fly” instead.

## 2.9 Reduced spectra in the FITS format

The `sorting_hat` combines extracted spectra (which were saved in binary format so far) with the wavelength solutions and writes them, along with the header, into FITS files. The FITS spectra have format `[2,npix,nord]`. The first plane contains wavelengths, the second plane – extracted spectrum. The matching algorithm selects ThAr exposure in the same mode which is closest in time to the science spectrum. Alternatively, interpolation of the wavelength in time between two ThAr spectra could be implemented.

## 3 Automated data reduction

### 3.1 Reduce\_ctio

The call is like

```
reduce_ctio, redpar,mode,flatset=flatset,thar=thar,order_ind=order_ind, star=star
```

where the keywords `flatset`, `thar`, `star` contain integer arrays with respective exposure numbers. If `flatset` is not given, the summed quartz will be read from the disk. It will also be used for the order location unless `order_ind` is given explicitly. The lists are normally produced by the calling script `sorting_hat`.

In the debug mode, there are stops to examine input data or intermediate results. Stand-alone use of `reduce_ctio` helps to debug data reduction. For example, order location in slicer mode can be checked by calling `redice_ctio, redpar, 'slicer', star=[1]`. Other essential parameters are passed through `redpar.imdir` and `redpar.prefix`, the summed flat is read from the disk (if it exists already), and we go directly to `fords`.

### 3.2 Sorting\_hat

This script is typically called as

```
sorting_hat, '111003', mode='slicer', /reduce, /getthar, /iod2fits
```

The `mode` keyword is obligatory. Optional keyword `run` specifies file prefix for the given night, like `qa39` (before September 20, 2011). However, this is not normally necessary because the prefix will be found automatically by examining data files for this night. The parameter file is read from the disk at each call of this script.

First, `sorting_hat` reads the logfile for the night. It is used to select data in the given mode: quartz flats, ThAr+iodine spectra and science spectra.

The data are first reduced (flag `/reduce`) by a call to `reduce_ctio`. Then automatic ThAr identification is performed (flag `/getthid`). The initial guess here is the most important thing. It can be given explicitly on input as `thar_soln='rchi111003.MNN'`. Otherwise, the `.thid` files for this night are looked for. Normally they are not yet present. In this case solutions for previous nights are sought by calling `findthid`. It reads the last 10 logsheets, determines calibration exposures and looks for

corresponding `.thid` files (the mode is ignored because any mode works as initial guess). If nothing is found, it returns. Note that wavelength solutions obtained at this step overwrite previous `.thid` and `.dat` files in respective directories.

The last step in standard data reduction is combination of spectra and wavelength solutions in FITS file (flag `/iod2fits`). Further operations (doppler code and data distribution) are not active in the current code.

To reduce all modes for a given night, use the short `allreduce` script which calls `sorting_hat` for all four modes.

### 3.3 Notes on the code structure

Code re-structuring was aimed primarily at increased flexibility and portability by eliminating hard-coded parameters. Most parameters are assembled in `redpar`, while the CCD-specific code is encapsulated in `chip_geometry` and `getimage`. Low-level routines that actually do the “number crunching” should remain as general as possible and should not access the disk files (with the exception of `thid` which needs the standard wavelengths from `thid_data.xdr`).

Writing and reading from the disk is reserved for the scripts, mostly (with the exception of `ctio_spec`). The scripts also “know” about naming conventions.

## 4 Summary

### 4.1 Changes to the code

This is the summary of code modifications.

- The parameter file is introduced, eliminating hard-coded parameters from the main reduction routines.
- Unified image-reading routine is used throughout. It is hardware-specific and should be changed if the CCD or controller are changed.
- All binning modes can now be reduced.
- The flat-field correction algorithm is modified, now the extracted spectrum is divided by the extracted FF.
- Summed flats, extracted flat fields and order locations are saved for each night and each mode in reserved directories, not in the main code directory.
- The script `sorting_hat` is changed, now it can be called once for all data reduction, including ThAr identification.

### 4.2 Further work

The following modifications are to be considered:

- Eliminate image truncation to use the full spectral range of CHIRON. This will involve re-definition of orders. Reduced spectra will contain zero values where there is no signal.

- Eliminate bad pixels or columns?

## A Some testing

### A.1 Spectrum extraction

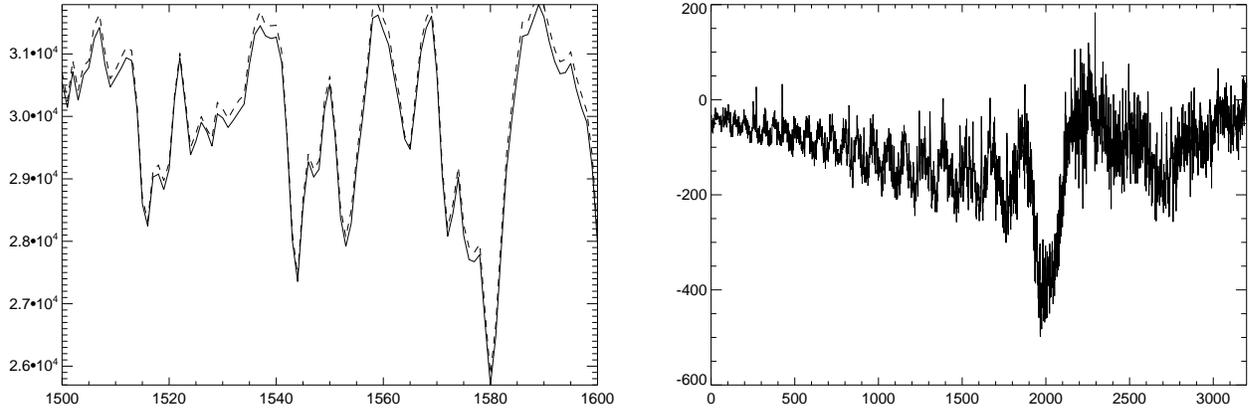


Figure 5: Comparison between boxcar (full line) and optimum extraction (dashed line) for fragment of order 10 in the iodine spectrum chi111003.1301 (slit mode). The right-hand plot is the difference between two spectra.

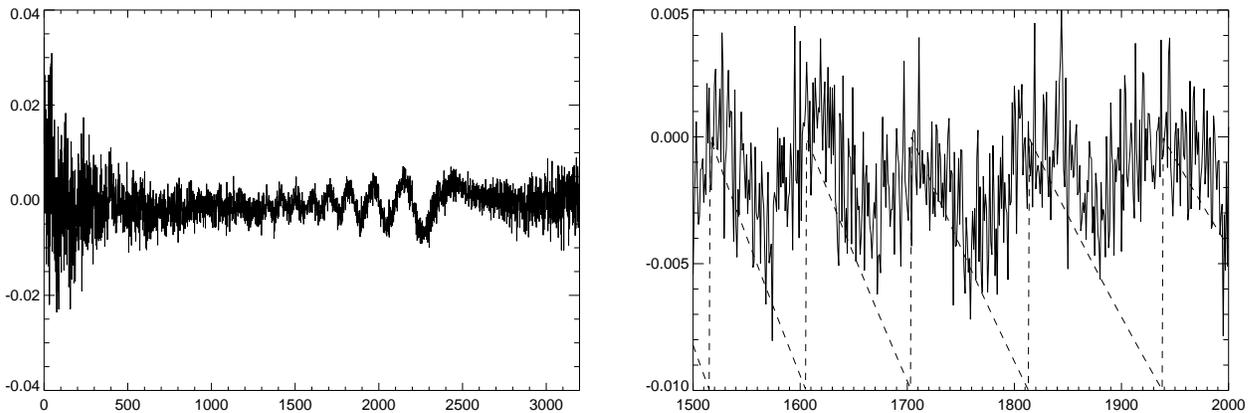


Figure 6: Normalized difference between boxcar and optimum extractions for spectrum chi111003.1301. Left: order 31. Right: order 35, with scaled fractional pixel of the order-center overlotted in dashed line.

The boxcar extraction implemented in the current code was compared to the optimal extraction. Optimally-extracted spectrum is produced by the `remove_cosmics` routine, but not used in the

getspec. Figure 5 compares a fragment of the iodine spectrum extracted by two methods. The difference is on the order of 1%, but it has a systematic “wavy” component. This is better shown in Fig. 6, where the normalized difference  $(s_1 - s_2)/s_1$  is plotted. The period of “waves” equals the interval where the order crosses vertical pixel boundaries. Further study is needed to see which extraction method gives less waves. If we select optimum extraction, it should be applied to the flat field as well, to correct same pixels as in the science spectrum (to the extent that FF and spectrum do not move vertically).

Another aspect of optimal extraction is lower noise. It was checked on the spectrum of faint star in fiber mode chi111003.2201. The noise was estimated as rms difference between extracted order and its 10-pixel median. For the average signal of 120ADU the rms was 10.7ADU for the optimally extracted spectrum and 11.7ADU for the boxcar extraction. The estimated photon noise is  $\sqrt{120/2.3} = 7.2\text{ADU}$ , so in this case the noise is dominated by the readout. We conclude that the boxcar extraction does not cause substantial increase of noise.

## A.2 Comparison with the old code

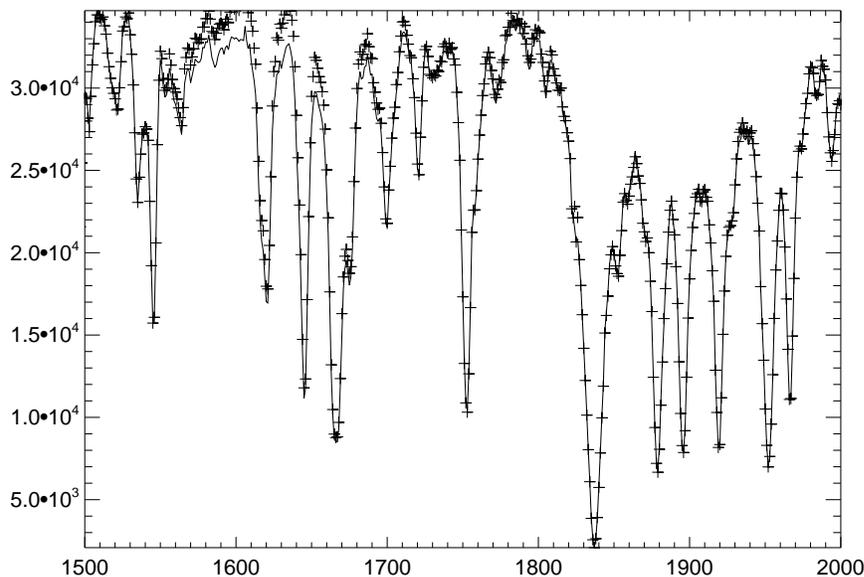


Figure 7: Portion of the extracted chi111003.1401 (iodine spectrum, narrow slit). Line – old code, crosses – new extraction multiplied by 3.0.

The extracted spectra (both boxcar and optimum) contain summed signal in ADU in each pixel. This was checked by direct summing of the image. In contrast, the old code gives signal exactly 3x larger, for unknown reason.

Figure 7 compares old and new extractions (with the factor of 3 artificially eliminated). The differences are most likely caused by the flat-field correction (the old code is not correcting anything wider than 5 pixels).

### A.3 Noise analysis

The program `noise.pro` can be run to compare the noise in “smooth” orders with its theoretical value expected from the gain and RON. Select suitable spectrum and order (say order 21), verify the absence of spectral features by plotting the order, then type `> noise, sp[*], ron=3, xwid=6, gain=2.3`. If all keywords are omitted, the RON is assumed zero, the default gain is 2.3. The program approximates the order by a polynomial, calculates relative fluctuations around this smooth curve in the central 1/3 of the order and displays the power spectrum of these residuals. It reports the average signal in ADU, relative rms and  $\chi^2$  values for the total and high-frequency fluctuations (they are same if the noise is white).

For example, noise analysis of the quartz spectra for the night of October 3, 2011 is given below for all 4 modes. The mean signal, relative rms, and  $\chi^2$  are listed for order 0 (faint) and order 30 (bright). Gain 2.3 is assumed for all modes.

Mode	File	order 0			order 30			RON	xwid
		mean	rms%	chi2	mean	rms%	chi2		
narrow	18	763	2.50	0.98	5490	1.02	1.02	3	6
slit	33	790	2.60	1.00	5760	1.00	1.14	3	6
slicer	63	1747	1.71	1.03	13000	0.70	1.15	3	12
fiber	48	4000	1.15	1.19	29500	0.52	1.34	3	5

For faint signal,  $\chi^2$  is close to one, being dominated by the readout noise. In the strong-signal regime, the actual gain can be somewhat smaller (due to under-corrected positive non-linearity), explaining slightly elevated  $\chi^2$ . We tested different flavors of extraction (with or without cosmic removal, before or after FF correction) and found little difference in the resulting white noise.

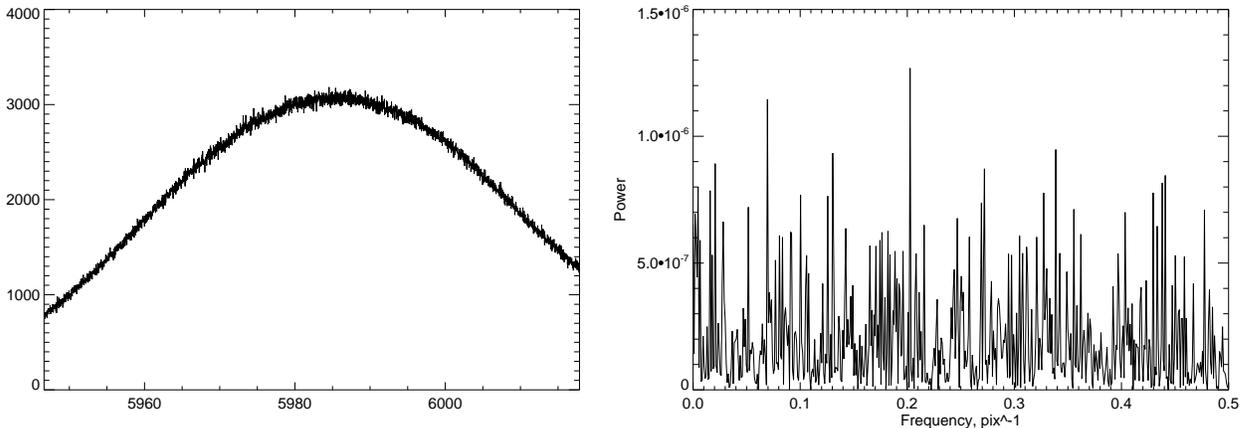


Figure 8: Example of noise analysis. Spectrum of U Oph on 11/09/04 rqa39.9398 with the slicer. Left: plot of featureless spectrum in order 22; right: power spectrum of noise in order 22 (mean signal 3276ADU, rms 1.3%,  $\chi^2 = 1.17$ ).

## B Data browser

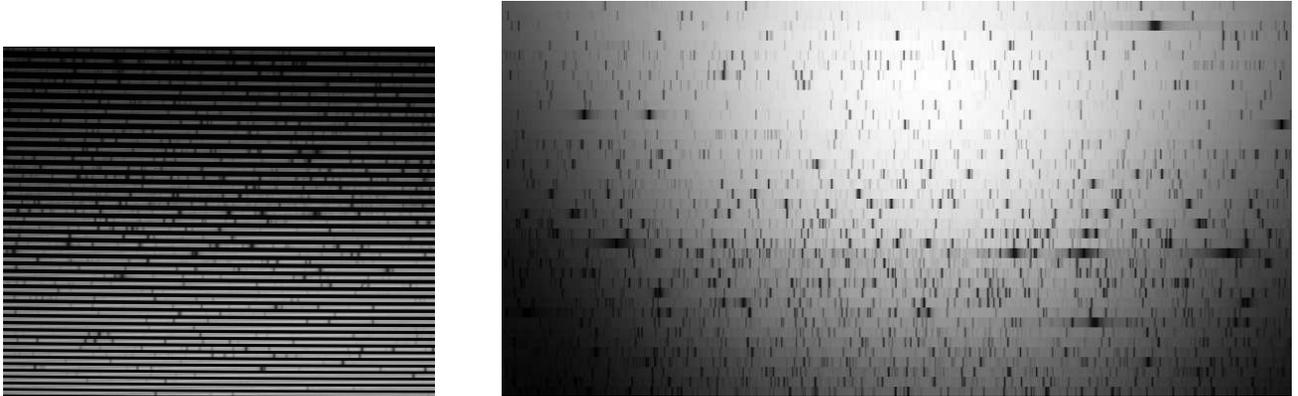


Figure 9: Spectrum of the bright star HD 20794 (slicer mode). Left: fragment of the raw image `chi1111008.1266`, right: extracted spectrum in 2D format as displayed by `xex`. The spectrum of this G8V star has a very high signal ( $7E7$  in order 21), yet no defects due to CCD blemishes or extraction are visible.

A small graphic application `xex.pro` (x-examine) for examining the reduced data is added to the code. First compile it, `>.r xex`, then execute it. The GUI can stay open while reducing the data or doing other work with IDL. The program suggests to access the common block by the command `> common xexcomm` to have the spectrum `sp` and other relevant data “at hand”. The program `xex` uses the parameter file defined above and accesses corresponding data directories. It forms the list of nights by reading the directory `/mir7/logsheets/`.

The GUI is rather obvious. First, you select the night from the drop-down menu (the last night is selected by default) and the mode. The lists of stars and ThAr for selected night and mode appear in the menus. By selecting a star, we see the corresponding line of the log-sheet and access its reduced FITS spectrum (an error message is printed if the file is not found). Press the **Read** button to read the file if the displayed name does not match your selection. Then selected orders can be plotted or browsed (buttons `<`, `>` and menu between them, button **Plot**). Use `2Dspec` to see all orders at once.