Alternative Scheme for CHIRON Data Reduction Manual

By: Murray McBain

Date: May 3, 2019

Introduction:

CHIRON:

CHIRON is a cross dispersed fiber-fed echelle spectrometer with multiple allowed input modes. CHIRON was developed by the Fischer Exoplanet Group to find Earth mass planets using radial velocity (RV) methods. CHIRON is installed on the 1.5m telescope at the Cerro Tololo Inter-American Observatory (CTIO) in Chile and has been recently accompanied by the RC-Spec and SIMON instruments. CHIRON was commissioned in 2011 and is currently operated by the Small and Moderate Aperture Research Telescope System (SMARTS).

Before CHIRON:

Originally, CTIO's 1.5m telescope used the bench mounted echelle in the 1980's, which CTIO discontinued in 2001. This left CTIO without a high resolution spectrometer for the 1.5m telescope for 7 years. In 2008 a new fiber feed was constructed and adapted for CTIO's old Blanco echelle (which was retired in 2004 from CTIO's 4-m telescope). This combination was offered to SMARTS and National Optical Astronomy Observatory (NOAO) users from 2008 - 2010 until replaced by CHIRON in 2011.

The entire instrument was upgraded in January of 2012. This upgrade introduced a new optical design, which would ultimately provide better results in the infrared region but limit the ultraviolet region. This limitation would result from the newly installed camera lens and from the fiber transmission breaking down in the blue. Despite losing the UV region, CTIO is still able to achieve their main goal because of the iodine line concentration between 500nm and 600nm.

CHIRON Data Reduction:

All CHIRON data reduction is processed by Yale and distributed as raw data and extracted data. It is through Yale's reduction pipeline that we lose any observations in the blue region. Yale avoids processing these regions, knowing that these regions will be useless to their research and potentially produce poor results.

Our objective however is to process these truncated regions, regardless of observation quality. This brings us to a new data reduction pipeline developed by Frederick M. Walter of Stony Brook University.

Yale's Reduction:

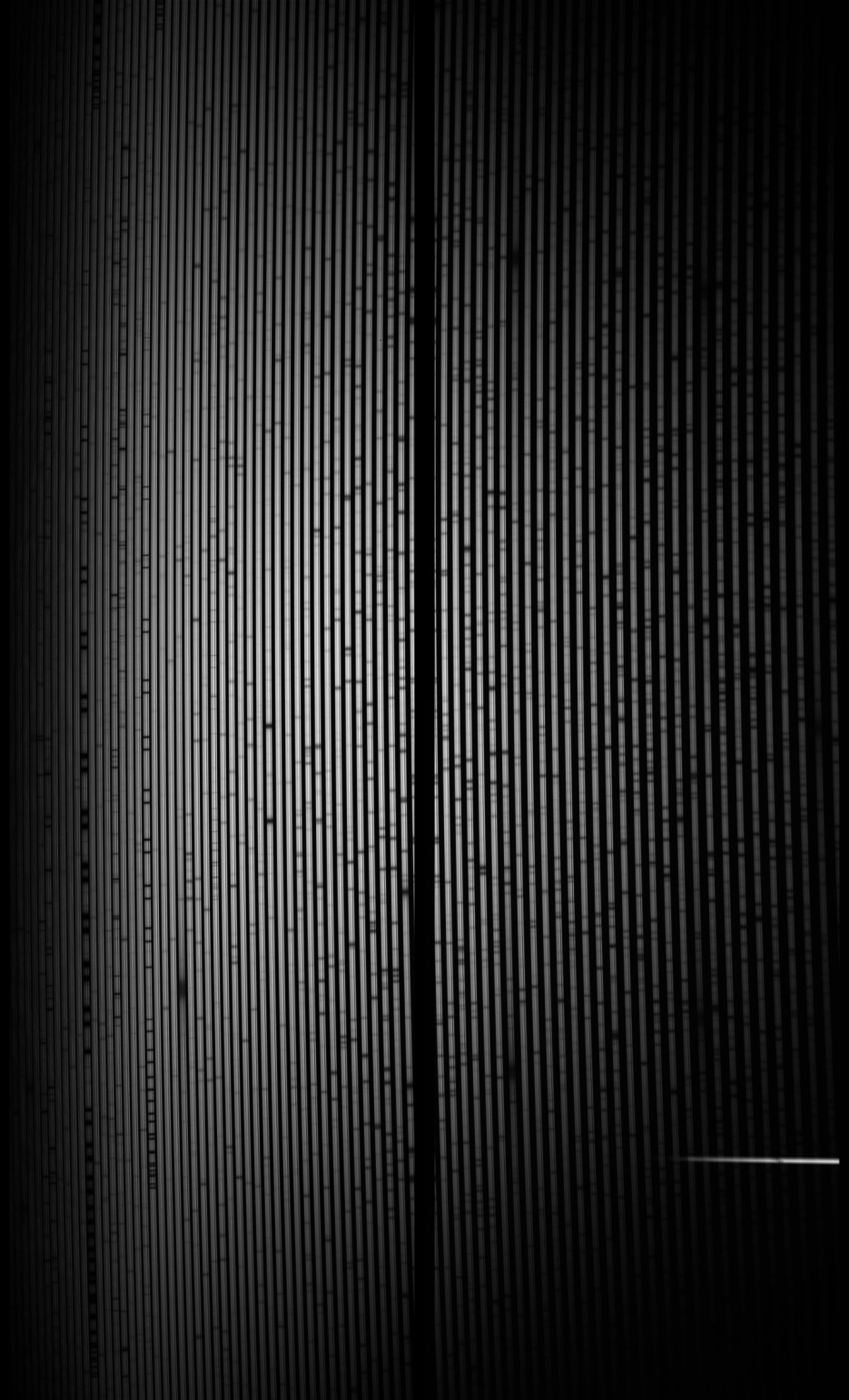
Yale's reduction package is based on the REDUCE code by Piskunov and Valenti, created in 2002. It was originally developed by D. Fischer and M. Giguerre with further modifications made in 2011 by A. Tokovinin. An overview of Tokovinin's "CHIRON Basic Data Reduction" can be found in the reference section of this write-up. The appearance of the CHIRON

Walter's Reduction:

This new reduction pipeline was developed by Frederick M. Walter of Stony Brook University in 2017. Walter used CHIRON for many years and realized that Yale's reduction pipeline treated some emission lines like cosmic rays that were being deleted. He also found that Yale was throwing away orders *m* =126 - 138 (*∼*4100 - 4600 Å). For Walter's research in galactic novae procession these missing orders would be advantageous.

Walter then created his own reduction pipeline called ch\_reduce. This pipeline could reduce data in fiber and slicer modes. As well as extracting 13 additional orders. ch\_reduce provides for different treatments of the local background, cosmic ray rejection, and allows for Gaussian and box car extraction. For an in-depth description of the calibration and data reduction techniques look for "An Alternative Scheme for CHIRON Fiber and Slicer mode data reductions" in the reference section.

The appearance of the CHIRON spectrum of HR2493 Procyon as processed by Walter’s reduction pipeline is shown in the following picture produced by SAOImage DS9 software.



Data Organization:

In order to use ch\_reduce you must organize your data properly in order for Walter's reduction pipeline to find the necessary data. Start by creating a folder called "chirondata" in any desired location. For example:

*D* : \*chirondata*

This means I've created my "chirondata" folder in my D drive. After this step is completed, 3 new folders need to be made in the chiron data folder. The names must be in the "chirondata" folder. The names must be thid, cal and YYMMDD. YYMMDD stands for the date of observation. For example:

*D* : \*chirondata*\*thid*

*D* : \*chirondata*\*cal*

*D* : \*chirondata*\181103

The folder 181103 is my YYMMDD folder, meaning my observation date is November 3, 2018. A new YYMMDD folder must be made for each night of observation. There is one last folder to be created; this folder must be made within the cal folder. This folder has to be called YYMMDD\_cals. Like the YYMMDD folder, a new folder needs to be made for every night of observation.

*D* : \*chirondata*\*cal*\181103\_*cals*

Finally we need to obtain and organize our actual data. We can get our actual data from the Yale archive that is used as the distribution hub for astronomers acquiring data from CHIRON. This will be downloaded as a tar file and has to be uncompressed. To unzip this tar file we need to use a correct unpacker like WinRar (DO NOT USE WinZip as this will corrupt the data). Once this tar file is unpacked, open the file named YYMMDD\_planid\_#. This file can be opened using Notepad and contains the general information for each exposure (e.g. exposure time, ThAr lamp, etc.). We can use this to distinguish each .fits file (unpacked from the tar file) as a calibration file or an observation file. All observation files go to the folder that we created called YYMMDD and all calibration files go to YYMMDD\_cals. All files listed as achi\*.fits are already reduced files from Yale. Each file’s observation number (chiYYMMDD.x.fits, where x is the observation number) matches with an observation number from YYMMDD\_planid\_# file; this is how we can determine which file is which. After we've organized all our data, we can then move on to the set-up of IDL.

Summary:

* *D* : \*chirondata* - Holds all data
* *D* : \*chirondata*\*thid* - Holds wavelength solutions from thid.pro
* *D* : \*chirondata*\*cal* - Holds all calibration data
* *D* : \*chirondata*\*YYMMDD* - Holds observation data for night of YYMMDD
* *D* : \*chirondata*\*cal*\*YYMMDD*\_*cals* - Holds calibration data from night of YYMMDD

Interactive Data Language:

Interactive data language (IDL) is an array based programming language provided by Harris Geospatial. This language is quite similar to Fortran. The main advantage IDL has over other programming languages is its multi-platform capabilities and its integrated graphics output which resembles Mathematica. A disadvantage to IDL is the license required to use it like Mathematica or CAD.

To obtain IDL, we will enter the URL ["www.harrisgeospatial.com"](http://www.harrisgeospatial.com/) into our web browser. This will bring us to the Harris Geospatial Solutions homepage. To the top right of our screen we'll see a profile icon. We can use this icon to register an account with Harris Geospatial. Once we've done this we can begin shopping for various services provided by Harris Geospatial.

After installing IDL and its license administrator, we only need to enter our obtained license into the license administrator and run IDL. Now that we can use IDL, we will set up the subroutines for Walter's software.

Importing Routines:

When IDL is first opened it will ask us to select a workspace. This workspace is where all our projects will be saved. If IDL does not ask us to select a workspace then we can use the following:

*file → switchworkspace → other*

Once we've completed this, we will then create a new project. The naming of the project doesn't matter, but let's use "chiron" for simplicity. We can create files for organization and import into any file destination (including chiron) by right-clicking on the destination and using import. Below are all the routines necessary for Walter's software (including Walter's software itself) and their respected URL locations

|  |  |  |
| --- | --- | --- |
| Library/Routine | Location | File Name |
| Astron | Idlastro.gsfc.nasa.gov/ftp/ | Astron.dir.tar.gz |
| CConroy | Github.com/cconroy20/fsps/releases | V.3.0 |
| Coyote | Idlastro.gsfc.nasa.gov/ftp/ | Coyote\_astron.tar.gz |
| Giguerre IDL Utilities | Github.com/mattgiguerre/idlutils | Download All |
| Markwardt IDL Library | Cow.physics.wisc.edu/~craigm/idl/idl.hmtl | Cmtotal.tar.gz |
| Thid | Github.com | Thid.pro |
| Fmtnum | Github.com | Fmtnum.pro |
| Lint | Github.com | Lint.pro |
| Thid\_function | Github.com | Thid\_function.pro |
| Slicer\_run\_thid | Email: murraymcbain@outlook.com | Slicer\_run\_thid.pro |
| Walter’s Software | Astro.sunysb.edu/fwalter/SMARTS/NovaAtlas/ch\_reduce/ch\_reduce.html | IDL Procedures |
| Wavfit | Email: frederick.walter@stonybrook.edu | Wavfit.pro |

Table 1: List of Routines

Various routines need to be modified in order to run more effectively. This is either because we are using Walter's software in Windows with its original platform being Linux or from our preferred formats.

Modifications:

Below is a list of modifications. This list contains the original line code and the altered line code. The list also includes where to find these modifications to make it easy for us to find.

|  |  |  |
| --- | --- | --- |
| Program | Line | Change To |
| Ch\_cals | 56 | if not ffile(caldata+'/chi'+day+'.1001.fits') then begin |
| Ch\_cals | 70 | if (mode eq 'fiber') and ffile(zmode+day+'\_cfits.sav') and ffile(flatsave) and ffile(qname+'.fits') and ffile(tharsave) and ffile(trf1) then calx=1 else calx=0 |
| Ch\_cals | 71 | if (mode eq 'slicer') and ffile('sl'+day+'\_cfits.sav') and ffile(flatsave) and ffile(qname) and ffile(tharsave) and ffile(trf1) then calx=1 else calx = 0 |
| Ch\_setwcal | 35 | root=getenv('chirondata')+'/cal/'+date+'\_cals' |
| Ch\_setwcal | 37 | cd,root,current=cdir |
| Ch\_setwcal | 38 | restore,'slthar'+date+'.sav' |
| Ch\_setwcal | 42 | thid0=get\_thid(date, mode = 'slicer') |
| Ch\_setwcal | 53 | thidfile=getenv('chirondata')+'/thid/slthid'+date+'.sav' |

Table 2: List of Modifications

Using Walter's Software:

The use of Walter's software is very straight forward after the modifications. All options for calibration and reduction can be chosen by inserting the appropriate "keyword". The input that goes along with ch\_cals (calibration program) and ch\_reduce (reduction program) is just simply the date of observation, YYMMDD. For example: ch\_cals, 'YYMMDD'. This input is the same for ch\_reduce. Below is a list of keywords for ch\_cals and ch\_reduce. To use these commands input needs to look like: ch\_cals, 'YYMMDD', /Keyword1,/Keyword2 ...

|  |  |
| --- | --- |
| Keyword | Purpose |
| DOGAUSS | Set for gaussian extraction of flat |
| EXAMINE | Set to examine each extracted order (sets plt) |
| FIXWID | sets fixed extraction width |
| FORCE | set to remake the quartz.fits file |
| MKTRACE | set to make new trace (2 to lookup w/o shifting) |
| MODE | default=fiber |
| NOWAVCAL | if set, skip wavelength solution migration |
| REFTHAR | index of reference ThAr spectrum, def=first |
| USEFLAT | set to use extracted flat for this day |
| WCALONLY | set to redo ch\_migratewcals |

Table 3: Ch\_cals Keywords

|  |  |
| --- | --- |
| Keyword | Purpose |
| CALDATE | date of calibration files, if not data date |
| DCR | set to use dcr CR filtering |
| DEFCAL | set to use default cal files |
| DO2, DO3 | set to reduce pair or triad starting with FILE |
| DOGAUSS | 1 or 2 for Gaussian extractions |
| EXAMINE | set to examine each extracted order (sets porders) |
| FILE | name of file; def = all |
| IGNORECAL | set to process cal file as object |
| LACOSMIC | set to run LaCosmic CR reduction (def) |
| MODE | default=fiber |
| NOCLEAN | set to skip extrapolation over bad points |
| NOCR | set to skip CR filtering; set if mode=slicer |
| NOHAMASK | set if H-alpha is in emission |
| NOPROC | set to skip post processing |
| NOSAVE | set to run code but to not save the data; sets STP |
| NSIG | sigma cut for replacing bad points in gauss fit |
| OPTIMUM | set for optimum extraction. Not yet implemented |
| PORDERS | set to plot individual orders |
| PLT | wavelengths to plot, all if set to 1 |
| PREFILTER | 1xn filter, def=5 |
| RDIREC | set to copy final .sav files to the ch\_red directory |
| SAVFILE | name of output .sav file prefix, if not the default |
| SPFITORD | background order for ch\_cleansp; def=2 |
| STARTORD | starting extraction order, def=0 |
| UPDATE | set to update single order |

Table 4: Ch\_reduce Keywords

A special characteristic to note about ch\_cals is that it will only ever need to be run once and it even detects if the files it produces have already been created. If the files have already been created then it will simply return and not alter anything unless specified by the keyword FORCE.

After running these routines you will have various IDL binary files (.sav) and FITS files (.fits) in the cal folder and YYMMDD folder.

Viewing reduced data:

To view a fits file, we may use any fits viewer like fv for example. The .sav file has to be read through IDL. These .sav are used to restore variables into IDL. The variables could be of any type but there are 3 that we are particularly interested in. Two of them are structures named THS and Zand the last one is a 2-D array named TH1. THS contains information on Walter’s “homemade” wavelength solution, Z contains information on a reduced observation and TH1 is the Thorium Argon echelle spectrum.

If Z contains the reduced observation data, then each .sav file in the YYMMDD folder has its own Z structure. To observe the structure, simply load in the desired .sav file through file explorer or use the command line. If you wish to use the command line, input the path along with the file name as a string (‘string’). For example:

* IDL> ‘D:\chirondata\181103\xchi181103.1163.sav

There are many objects within the Z structure. Use tag\_names(Z) in the command line to display the name of every single structure. To call these objects, use the object name and the structure name. If we wanted to call the wavelengths held in the Z structure we’d say:

* IDL> wavelengths=Z.W

Knowing this we can create an intensity versus wavelength spectrum. If Z.F represents flux (intensity) and Z.W represents wavelengths then we can plot this spectrum by calling both objects and plotting them like so:

* IDL> Plot(Z.W,Z.F)

Wavelength solution:

The wavelength solution is produced by ch\_cals and is saved in YYMMDD\_cals folder as YYMMDD\_cfits.sav. This .sav final contains the THS structure. One of the most important objects in THS is fits. THS.cfits are the coefficients for the n-th order (typically n=6) polynomial that approximates the wavelength versus pixel solution. There are other objects that become important later such as THS.m (order number) and THS.mlam (order times central wavelength).

ThAr Echelle Spectrum:

The Thorium Argon echelle spectrum is also contained in a .sav file called sltharYYMMDD.sav. This file does not load in a structure but rather a 2-D array called TH1. In order to observe this spectrum we simply plot the 2-D array by entering the below command into the command line.

* IDL> plot(TH1)

Results:

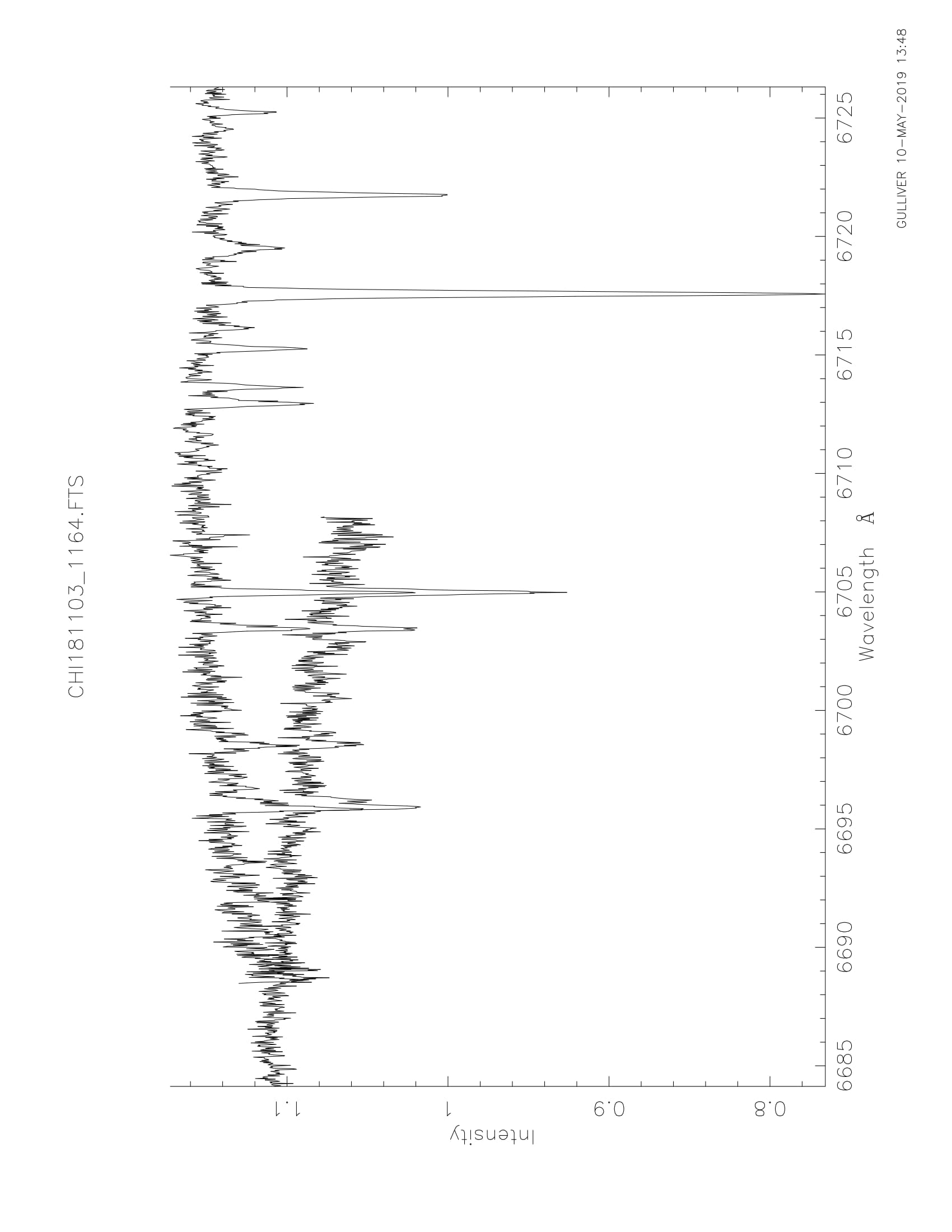
Viewing the observation spectrum using IDL can prove frustrating at times. It is easy to plot the spectrum using the previously discussed Z structures, but the problem is that we can distinguish between orders because we can’t manipulate the plot of the z structures. This led us to using an OpenVMS program, created by Dr. Austin Gulliver, in order to view the individual orders of the spectrum. We can determine the quality of a spectrum by examining the overlap of orders. If the overlap of an order with its adjacent orders do not agree then the spectrum’s quality is poor. The reason for this poor quality is most likely from an inadequate wavelength calibration. Our spectrum using Walter’s software of HR2943 Procyon has orders that don’t overlap properly with adjacent orders. Below are 3 photos, the first being a spectrum with agreement between orders in the UV region, the second being our reduced spectrum HR2943 Procyon using Walter’s software in the UV region and the last being from the same spectrum as the second but in the infrared region.

A close up of text on a white surface

Description automatically generated

A picture containing nature

Description automatically generated



This tells us that there is something wrong with Walter’s wavelength solution. As you can see we have good agreement in the infrared region, however we have no agreement in the UV region. There are 3 potential reasons for the type of behavior we see. It may be because of a poor wavelength solution algorithm in which case we would need to provide ch\_reduce with our own wavelength solution, we are using too much of each order (extends too far out in the edges), or the least likely is that the CHIRON spectrograph’s optical design is to blame. We will be examining the first 2 potential reason as there is obviously nothing we can do about the last reason, that of faulty optical design.

New Wavelength Solution:

Our first attempt to solve this inconsistent wavelength solution is by using another program that Walter has created called ch\_setwcal. This routine takes an old wavelength solution from Yale typically named achiYYMMDD.thid and uses it as an initial guess for another wavelength solution. Within this routine it will use thid to form a new wavelength solution and save it so that ch\_reduce uses this wavelength solution instead of the original THS structure that contains Walter’s wavelength solution. There is a problem however with this; we do not have an old wavelength calibration solution from Yale. So we need to create a wavelength solution of our own using a widely used program called thid.pro.

Thid:

To use thid we need to provide it with the Thorium Argon echelle spectrum, TH1 and specific objects from Walter’s wavelength solution THS. Thid is set-up for input and output as the following:

* IDL> thid, spec, obase, mlam, wvc, thid, inpord=inpord, inpcoef=inpcoef, fitdeg=fitdeg

|  |  |
| --- | --- |
| Variable | Definition |
| SPEC | Thorium Argon Echelle Spectrum (TH1) |
| OBASE | Minimum Order Number (min(THS.m)) |
| MLAM | Order Times Central Wavelength (THS.mlam) |
| WVC | Output Wavelength Coefficients |
| INPORD | Orders Used For Initial Guess (THS.m) |
| INPCOEF | Initial Guess For Wavelengths Coefficients (THS.cfits) |
| FITDEG | Degree Of Polynomial Fit (Typically 6) |

Thid can be challenging to use in the command line. We would need to load in two IDL binary files (TH1 and THS). Luckily we have developed a program that will automatically open these files and run thid. This program is called slicer\_run\_thid and it works the same way as ch\_reduce. You only need to call the program and specify the date.

* IDL> slicer\_run\_thid, ‘YYMMDD’

After obtaining the necessary thid structure and wavelength coefficients, we should be able to run ch\_setwcal.

Problems:

As said previously after running thid, ch\_setwcal should run smoothly. This however is simply not the case. Ch\_setwcal can’t seem to find a decent number of pixels that satisfy conditions, along with being unable to provide a decent number of wavelengths. This lack of good data points (good lines) causes the procedure in ch\_setwcal called wavfit.pro to not have the ability to form any sort of wavelength fit.

Further problems may include that this reduction pipeline isn’t made to reduce in slicer. The evidence lies in the strange behavior when telling ch\_reduce to reduce in slicer or ch\_cals to calibration in slicer. Certain commands that have been commented which appear to be useful toward slicer reduction.

Last evident problem is that thid may not be working as well as initially presumed. Thid runs which is good but the amount of “good lines” produced is much too low to do anything with. This may be the source of the problem with ch\_setwcal. Factors for the thid problem may include that our initial guess (Walter’s solution) needs to be improved to result in a better thid structure.

Next Steps:

Thid needs to be perfected. It is difficult to understand what methods it using to determine the good lines. The methods seem obvious conceptually but mathematically are challenging. So the initial thing to work on is the beginning guess of thid, or potentially the spectrum range of thid (TH1).

After thid is corrected the next step would be to run ch\_setwcal. This correction to thid inputs may cause ch\_setwcal to run smoothly, but that is often wishful thinking with this program.

A worthy alternative to correcting the wavelength calibrations is trimming the edges of orders. The first step to examine how to do this is to find the routines that work with the order individual, assuming that these routines are most likely within ch\_cals. Once found, we can then hopefully use basic array operations within IDL to trim the edges of the orders. The reason we believe this will work is because the spectrum orders in the red are in agreement with adjacent orders, but have less overlap between orders. The opposite is true with the spectrum orders in the blue where there is a lack of agreement with more overlap. So if we can trim the orders that approach the blue then we may be able to produce a stronger wavelength solution.

References:

* Tokovin, A.: CHIRON basic data reduction, http://www.ctio.noao.edu/noao/sites/default/files/telescopes/smarts/tele15/chireduce.pdf.
* Tokovin, A., Fischer, D., Bontani, M., Giguere, M.J., Moore, P., Schwab, C., Spronk, J., Szymkowiak, A.: CHIRON-A Fiber Fed Spectrometer for Precise Radial Velocities. (2013).
* Walter, F.: An Alternative Scheme for Chiron Fiber and Slicer Mode Data Reductions.
* SMARTS 1.5-m CHIRON - Yale University, www.astro.yale.edu/smarts/1.5m.html.