modsIDL:

MODS Spectral Data Reduction Pipeline

Document Number:	OSU-MODS-2014-002
Version:	1.0.2
Date:	2019 February 20
Prepared by:	Croxall, Pogge, & Berg OSU

THE OHIO STATE UNIVERSITY

Distribution List		
Recipient	Institution/Company	Number of Copies
Kevin Croxall Richard Pogge Danielle Berg	The Ohio State University The Ohio State University The Ohio State University	1 (.docx + pdf file) 1 (.docx + pdf file) 1 (.docx + pdf file)

Document Change Record			
Version	Date	Changes	Remarks
0.1	2014-03-07	n/a	Template from Comm Report
0.2	2014-04-15	Many and various	Second Draft
0.3	2014-04-15	Updated installation section and various misc.	
0.4	2014-04-16	Added worked example	First Released Version
0.6	2014-08-25	Numerous Updates based on feedback	
0.8	2016-03-09	Restructuring of science and standard processing including wrappers to simplify command lines. Bug fixes and clarifications. Improved Prism modules.	Second Released Version
1.0	2019-02-20	MODS2 Grating mode release	Binocular-capable Release

Contents

1 Introduction

1.1 Scope

This document describes the tools and procedures for reducing spectral data obtained with the MODS instruments at the Large Binocular Telescope.

1.2 Document and Software Version

The version of this document is 1.0, the first release that includes working code for reducing MODS1 and MODS2 grating mode data. It corresponds to modsIDL pipeline software version 1.0. We have used the first formal release for binocular MODS grating spectral reductions to (finally) reconcile the version numbering which has long been a source of confusion.

1.3 References

1. *On-Sky Performance of the Multi-Object Double Spectrograph on the Large Binocular Telescope*, Pogge. R.W., Atwood, B., O'Brien, T.P., Byard, P.L., Derwent, M.A., Gonzalez, R., Martini, P., Mason, J.A., Osmer, P.S., Pappalardo, D.P., Zhelem, R., Stoll, R.A., Steinbrecher, D.P., Brewer, D.F., Colarosa, C., & Teiga, E.J., 2012, SPIE, 8446, 84460G
2. *MODS Instrument Manual*, 2012, Pogge, R.W., OSU-MODS-2011-003.
3. *MODS Basic CCD Reduction with* modsCCDRed, 2012, Pogge, R.W., OSU-MODS-2012-002.
4. Bohlin, R.C., Colina, L., & Finley, D.S. 1995, AJ, 110, 1316, *White Dwarf Standard Stars: G191-B2B, GD 71, GD 153, HZ 43*
5. Hubble Space Telescope Calibration Database System, (CALSPEC), www.stsci.edu/hst/observatory/crds/calspec.html
6. Massey, P., Strobel, K., Barnes, J.V., & Anderson, E., 1988, ApJ, 328, 315. *Spectrophotometric Standards*.
7. Oke, J.B. 1990, AJ, 99, 1621. *Faint spectrophotometric standard stars*,
8. Oke, M, et al. 1994, PASP, 106, 566. *Southern Spectrophotometric Standards, II*.
9. Massey, P. & Gronwall, C. 1990, ApJ, 358, 344, *The Kitt Peak spectrophotometric standards - Extension to 1 micron*
10. van Dokkum, P.G. 2001, PASP, 113, 1420. *Cosmic-Ray Rejection by Laplacian Edge Detection*
11. Kelson, D.D. 2003, PASP, 115, 688. *Optimal Techniques in Two-Dimensional Spectroscopy: Background Subtraction for the 21st Century*.

1.4 List of Abbreviations and Acronyms

ADC	Analog-to-Digital Converter
ADU	Analog-to-Digital Converter Units
CCD	Charge Coupled Device
DETXY	Detector X,Y coordinate system
FITS	Flexible Image Transport System
Gb	Gigabyte
GHz	Giga-Hertz
IDL	Interactive Data Language
IRAF	Image Reduction and Analysis Facility
LBT	Large Binocular Telescope (the telescope itself...)
LBTO	Large Binocular Telescope Observatory (operations arm)
LDG	Left Direct Gregorian (LBT telescope focal station)
Mb	Megabyte
MGIO	Mt. Graham International Observatory
MODS	Multi-Object Double Spectrograph
MOS	Multi-Object Spectra/Spectroscopy
ND	Neutral Density
NIST	National Institute of Standard and Technology (United States)
NSF	National Science Foundation (United States)
OSU	The Ohio State University
PA	Position Angle (celestial coordinates)
PSF	Point Spread Function
RAM	Random-Access Memory
RMS	Root Mean Square
ROI	Region-Of-Interest
SVN	Apache Subversion
WCS	World Coordinate System
XIDL	A package of IDL codes assembled by Jason X. Prochaska

2 Overview

This document describes how to perform basic 2D processing of MODS science data and the associated calibration data, and then describes how to extract 1D calibrated spectra that can then be measured using any spectral analysis package that can examine flux-calibrated 1D spectra on a linearized wavelength scale. We describe the Python programs that are part of `modsCCDRed` package for basic 2D reductions common to all MODS data, and the IDL programs that use the XIDL library to carry out reduction and calibration of long-slit and multi-slit MODS spectra. Note that long-slit spectra are considered by the pipeline to be a logical subset of multi-object spectroscopy, so the basic procedures for both are essentially the same.

Currently the MODS pipeline assumes that full-frame, unbinned 8K×3K MODS CCD images are supplied for Grating mode, and unbinned 4K×3K images for Prism mode. The pipeline uses geometric transformation maps computed for these standard readout configurations and there is no simple transformation that can be applied for other readout formats. If your data use non-standard readout modes you must rely on other means to reduce your data.

2.1 The MODS Spectral Data Reduction Pipeline

The MODS spectral data reduction pipeline was developed originally to reduce data obtained for multi-slit mask spectroscopy of extragalactic HII regions for an NSF-sponsored research project. However, there being no observatory pipeline, and because US and German LBT partners do not have access to INAF's LBT facility instrument pipelines, word spread that OSU had a "MODS pipeline", and the modsIDL code began to leak out into the community. Once it got into the wild, we expanded our documentation beyond what was used internally to our research group. The pipeline has been designed to reduce multi-slit and long-slit data in both the grating and prism modes, though because the prism modes are less commonly used the level of development for prism reductions is not as advanced as for the grating modes.

The MODS pipeline has been implemented using a combination of Python and IDL. This hybrid approach was made because it seemed better to adapt an existing IDL package that was close (but not 100%) what we needed for MODS, and we had existing python code for the basic 2D reductions already in hand and tested for the oddities of the first-light MODS CCD detector controllers. Given more resources and a broader mandate, a more sensible approach would have been to develop completely in open-source Python, but here we are.

The Python portions are the `modsCCDRed` package developed during early MODS1 commissioning to provide basic 2D CCD reduction tools common to all MODS data. These tools prepare raw MODS data for further reduction with the spectral reduction pipeline.

The IDL portions of the pipeline were designed as an add-on to the XIDL software libraries developed by Jason X. Prochaska and Joe Hennawi (github.com/profxj/xidl). This design decision was one of basic economy: the XIDL library had already developed many of the basic functions we needed, allowing us to concentrate on the reduction challenges peculiar to the MODS instruments. This greatly reduced the development process, and we were able to build on the proven (and already mostly debugged!) XIDL routines.

MODS2 support was added late due to a number of contributing factors. For the v1.0 release only grating spectra will be supported initially, with (lesser used) prism modes to follow as human resources to calibrate and test this little-used mode become available.

While every effort has been made to make these scripts straightforward and simple to use, a basic working knowledge of Python and IDL scripting, as well as an understanding of the basics of spectroscopic data processing, will be helpful.

2.2 Terms and Conditions of Support

We are providing this pipeline and software as-is, with no warranty or offer of user support. We are willing to fix bugs and answer basic questions, and will do our best to keep the code updated, but step-by-step instruction or hand-holding is beyond our ability to provide. We cannot support all combinations of versions of IDL, Python, or various operating systems. We expect the user to have at least a basic working knowledge of Python and IDL as a user (if not as a programmer), be acquainted with the MODS and their operational characteristics, have some experience with the basics of CCD spectroscopy, and have an understanding of the basic principles of operation of astronomical CCDs.

The NSF funding that supported the development of this pipeline ended in July 2015, the postdoc who did most of the development (KVC) left astronomy to become a data scientist in May 2016, and the MODS PI (RWP) has moved on to other projects after MODS handover, which means that without substantial infusion of new resources and an IDL Jedi on staff, we will only make changes as absolutely required to get our science done, and are not open to providing new features. If OSU students and staff aren't using a particular MODS mode (e.g., prism), chances that new features or refinement will happen are unlikely. We had hoped that LBT Observatory partner users and staff would have become proficient in the use of the pipeline over time to spread out this lead, but this has not really happened, and there has been no systematic effort on the part of the LBT Observatory to provide support for reduction pipelines.

2.3 Acknowledging use of the MODS Pipeline

The MODS reduction pipeline was developed independently of the MODS instrument project as part of NSF Grant AST 1108693 which supported the Chemical Abundances of Spirals (CHAOS) project headed by Evan Skillman (Minnesota) and Rick Pogge (OSU). MODS pipeline author Kevin Croxall was a postdoctoral fellow supported by this grant and an OSU Price Fellowship from 2012 through mid-2016.

If you publish MODS data reduced with the modsIDL pipeline, we ask that you add this line to the acknowledgments section of your paper:

> This paper made use of the modsIDL spectral data reduction reduction pipeline developed in part with funds provided by NSF Grant AST-1108693 and a generous gift from OSU Astronomy alumnus David G. Price through the Price Fellowship in Astronomical Instrumentation.

A software DOI has been registered with Zenodo (https://zenodo.org/record/2561424). This should be cited in the bibliography of your paper. The link should provide you with information on the citation, including template BibTeX entries to use.

3 Installing the Software

3.1 System Requirements

The modsIDL pipeline was developed on a Linux machine with 8 Intel Core i7 3.4 GHz processors and 8 Gb of RAM. It has also been tested and run on a MacBook Air with 2 GHz Intel Core i7 and 8 Gb of RAM.

We therefore recommend your system have at least a 3 GHz processor and 8 Gb of RAM.

The modsIDL pipeline has also been run successfully on machines with slower processors and less memory than this recommended limit. However, given the large size of processed MODS files (~300 Mb after sky subtraction), a *minimum* of 4 Gb of RAM is recommended.

You need your own copy of and license for IDL. We developed and tested modsIDL with IDL version 8.1.

For Python we strongly recommend the latest Anaconda distribution of Python 2.7 (www.anaconda.com), which we have found works best for most Linux distributions and Mac OS/X computers. None of the programs for the pipeline have been ported or tested under Python 3 at the time of this writing, but with Python 2 end-of-support approaching in 2020, this may change.

3.2 Installing modsCCDRed

The modsCCDRed programs are written for python v2.7, and requires the numpy and astropy modules. We strongly recommend that you use the Anaconda distribution of Python (www.anaconda.com). Anaconda comes with the latest stable versions of numpy and astropy, and we've found that it works out of the box on Linux and Mac. If you are using some other Python distribution, you will need to separately install and verify numpy and astropy and all of the modules they depend on. Better to install Anaconda and get everything you need in one distribution.

To obtain modsCCDRed, please visit

https://github.com/rwpogge/modsCCDRed

Starting in 2018 we are maintaining modsCCDRed on a public GitHub page. The older webpage at OSU has been phased out, and links to GitHub, though it is available for getting older versions.

3.3 Installing modsIDL

The modsIDL programs are incorporated into the XIDL package developed principally by Jason X. Prochaska and Joe Hennawi (github.com/profxj/xidl). We have chosen to build the pipeline upon the many tried and true tools that have already been developed for other multi-object spectrographs. modsIDL was developed using the SVN installation of XIDL from the fall of 2012. We are currently providing the entire XIDL package that was used for the modsIDL development so you do not need to separately obtain and install this package. In time, we expect the MODS components we have developed to become part of the public XIDL package, but we have a ways to go yet.

Advanced XIDL users may wish to install the MODS scripts into their personal versions of XIDL. This is only recommended for power IDL users who are familiar with the structure of

XIDL and are aware that changes may have occurred in XIDL that may be incompatible with the current pipeline. For those of you electing to follow this path, the most current stable MODS scripts are located in the appropriate directory under the Longslit branch of XIDL. If you don't know what this means, you shouldn't try this. Note, however, until the modsIDL package matures to the point that we can fully integrate it into XIDL and move from development to maintenance as a part of the public XIDL, we can offer only limited support if you go this route.

3.3.1 Downloading xidl and modsIDL

If you are doing a new installation and need both xidl and modsIDL, the distribution tar file is available at

www.astronomy.ohio-state.edu/MODS/Software/modsIDL

This webpage has links to the latest release and older versions, and includes a full snapshot (~250Mb!) of the entire xidl distribution we know works with the modsIDL code.

In addition to the pipeline software, this webpage will also include links to the manual and set of training data discussed in later chapters.

If you already have a working version of the modsIDL pipeline from a pre-1.0 release, you can upload just the changes to the core MODS code from GitHub:

github.com/rwpogge/modsIDL

and follow the instructions for installing the updated code. None of the modsIDL code changes any of the xidl package proper. This is a much faster way to update modsIDL (~290kb instead of ~250Mb).

3.3.2 Unpacking & Installation

If doing a full installation, unpack the tarball (e.g., `modsIDL_linux_v1.0.tgz`) in the usual way

```
tar -xvzpf modsIDL_v0.1.tgz
```

This will create the `modsIDL/` directory and fill it with the requisite programs and support files for the entire xidl package plus modsIDL. The `modsIDL` directory has five (5) subdirectories:

`coyote/`	a version of the Coyote IDL Library (in case the user does not have it)
`idlspec2d/`	IDL 2D spectral package by David J. Schlegel & Scott Burles
`idlutils/`	IDL utilities package by David J. Schlegel
`xidl/`	the XIDL package by Jason X. Prochaska
`Docs/`	a copy of this manual (PDF) and supplementary documents

If you already have a working Linux or OSX version of the modsIDL pipeline from before, you can update to the latest version on this repository by downloading the new code specific to MODS instead of having to re-install the entire xidl snapshot (~290k vs ~250M for the whole thing).

Download the tar file in a safe place, say

```
/path/to/tarball/modsIDL.justMODS.v1.0.tgz
```

Then make a backup of your old version

```
cd /xidl/Spec/Longslit/pro/LBT/
```

and make sure that when you type ls you see the MODS folder like this:

```
LUCIFER/ MODS/
```

Then make a copy the old MODS folder as a backup in case you need to roll back % tar cvzf oldMODS.tgz MODS and replace the contents of the MODS folder with the new code:

```
tar xvzpf /path/to/tarball/modsIDL.justMODS.v1.0.tgz
```

and remember to say "yes" if asked to overwrite any existing files.

3.3.3 Environment

To run the modsIDL package, the modsIDL directory must be in your IDL path. In general, most IDL codes are kept together in a top level directory such as `~/myidl`. All code is then placed under this directory. The user will also need to define the necessary environment variables and update their `IDL_PATH` as necessary. The following is recommended to be added to your .cshrc file:

```
setenv modsIDL ~/myidl/modsIDL
if ( -d $(modsIDL) then
    setenv IDLUTILS_DIR $(modsIDL)/idlutils
    setenv IDLSPEC2D_DIR $(modsIDL)/idlspec2d
    setenv XIDL_DIR $(modsIDL)/xidl
    setenv LONGSLIT_DIR $XIDL_DIR/Spec/Longslit
    setenv IDL_PATH \
    $IDL_PATH+$IDL_DIR\/lib:+$IDL_DIR\/examples:+${modsIDL}:+pro
endif
```

Or alternatively, add the following to your .bash file:

```
if [ -d ~/myidl/modsidl ] then
    export IDLUTILS_DIR=~/myidl/modsidl/idlutils
    export IDLSPEC2D_DIR=~/myidl/modsidl/idlspec2d
    export XIDL_DIR=~/myidl/modsidl/xidl
    export LONGSLIT_DIR=$XIDL_DIR/Spec/Longslit
    export IDL_PATH=+$IDL_DIR\/lib:+$IDL_DIR\/examples:+~/myidl:+pro
endif
```

To verify that modsIDL has been included in your path you can type:

```
env | grep –i modsIDL
```

At this point you should be ready to use the modsIDL pipeline.

3.3.4 Software Updates

Because modsIDL is built on xidl, we will continue to provide a full snapshot of the entire xidl package + modsIDL on the modsIDL webpage at OSU for at least the foreseeable future. Once you have the xidl package, updates to the core modsIDL code will be available from GitHub:

github.com/rwpogge/modsIDL

which just distributes the contents of the MODS branch of xidl. A bit more sophisticated way to update using git tools will be described in future releases.

4 Basic MODS Reduction Procedures

The basic outline of the MODS spectral reduction pipeline is shown in Figure 1.

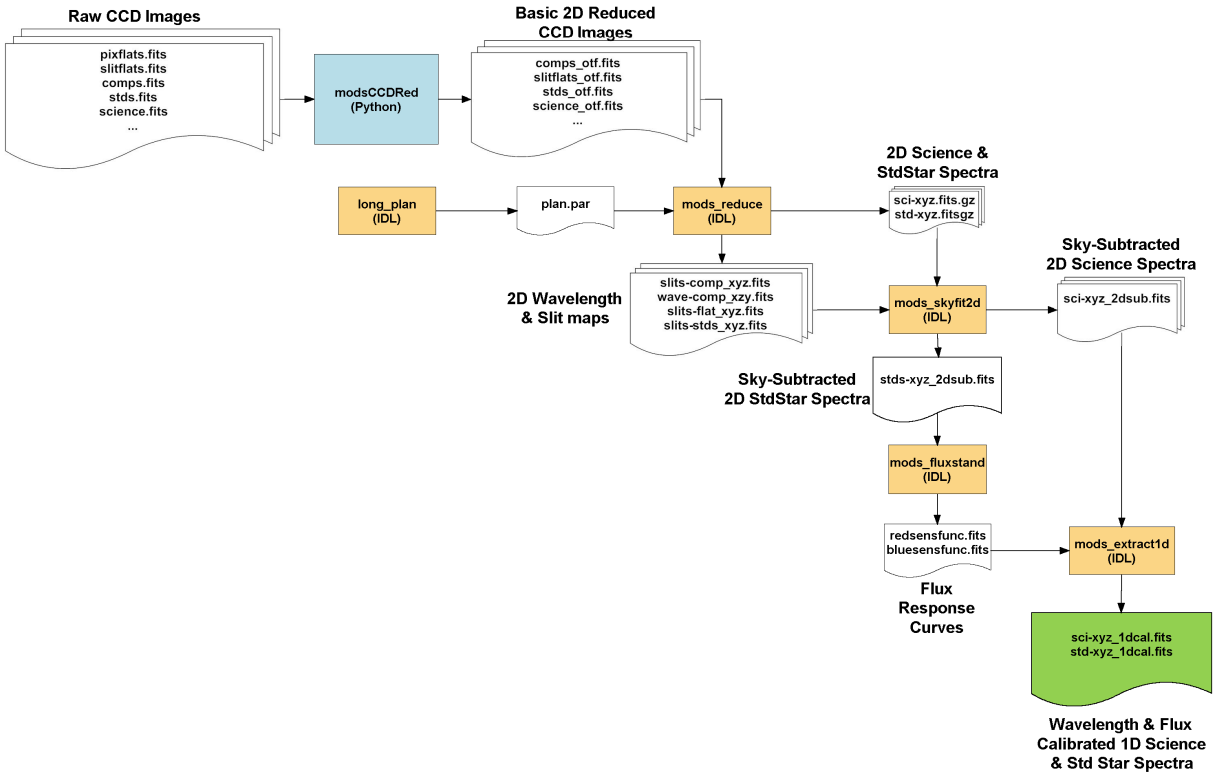

Figure 1: Workflow of the MODS reduction pipeline. Python programs (modsCCDRed) are in light blue, IDL routines are in pale orange, intermediate data files are in white, and the final output, 1D wavelength and flux-calibrated sky-subtracted science and standard-star spectra, are in green.

MODS data reduction Proceeds in two steps:

1. Basic 2D reductions common to all MODS CCD images (bias and flat fielding) applied to all raw data.
2. Reduction of the Basic 2D reduced images to 2D and 1D wavelength and flux calibrated spectra with sky subtraction

The first step uses a set of custom Python scripts (modsCCDRed).

All subsequent steps use a set of custom IDL routines (modsIDL) to trace slits, compute and apply wavelength and flux calibrations, subtract sky, and extract 2D and 1D spectrophotometry.

5 Preparing for MODS Reduction

Organizing your MODS data at the start is essential for a smooth reduction process. While the XIDL scripts allow a fair amount of latitude for how you set things up, we recommend the following organization.

Start with a top-level working directory. Because a lot of the reduction steps are specific to a single target (e.g., custom mask, particular calibrations, etc.), we recommend that you use one top-level working directory per target or group of targets that share common properties. For this manual we will generically refer to this as the "working directory" and for our examples will be given the generic name "`myProject`".

Inside `myProject/` you will need to create three directories:

`Raw/`	For raw, unprocessed MODS data
`Proc/`	For basic 2D images created by modsCCDRed, ready for modsIDL
`Science/`	For reduced/extracted "science" spectra created by modsIDL.

The contents of `Raw/` are not used by modsIDL proper, but is where you keep the original data and perform the basic 2D reductions using modsCCDRed. It is where you prepare flat fields and process comparison lamps ("comps"), standard stars ("stds"), and science target spectra to prepare them for reduction with modsIDL.

Once you have 2D spectra ready for modsIDL they are copied from `Raw/` into `Proc/` where modsIDL will be told to look for them. A good practice is to keep untouched copies of these spectra in your `Raw/` directory in case you have to backup and redo a reduction (and while learning the first time, you will...). They way to think of this is that we keep protected "master" copies of 2D spectra in `Raw/`, and work off copies in `Proc/`.

`Science/` is where the modsIDL pipeline will write all of the reduced science data and standard star data. If you forget to create it, modsIDL will create it for you.

In addition to these three folders, the working directory above them needs to contain working files created and/or used by modsIDL. Principal among these are:

MODS Mask Specification (MMS) files for your data

Runtime parameter files created by modsIDL (e.g., plan.par)

Calibration images such as slit and wavelength maps that are created by modsIDL during processing.

A good initial organization of the data will make the reduction process run smoothly. As you gain more experience using modsIDL, you will find ways to customize your workspace to suit your needs. For this manual we will stick to the basics and concentrate on reductions of data of single targets, both long-slit and multi-slit.

5.1 Basic 2D CCD Image Reductions

Basic 2D CCD reduction is performed using the modsCCDRed package. These steps include removal of the bias and overscan regions as well as the creation and application of flat fields. In this manual we will only outline the application of the modsCCDRed tools. For detailed instructions on the use of modsCCDRed, we refer the reader to the modsCCDRed manual which can be found at github.com/rwpogge/modsCCDRed.

We will assume below you have read the modsCCDRed manual for the details of the various steps outlined below.

All of these steps will occur in the `myProject/Raw/` directory.

5.2 Gather your Data

Retrieve your raw science and calibration data from the LBT archives.

The minimum calibration data needed are:

1. Flat fields appropriate to the instrument mode. For example, for long-slit or multi-slit grating spectra you need pixel flats, and slit mask flats taken during the run, including slits flats taken for the flux calibration standard stars. Flats are generally stable on month timescales, so finding a suitable set of flats nearest your night of observing is good enough.
2. Comparison lamp spectra for wavelength calibration. If using long-slit masks, find the set of comparison lamps taken during the observing run. If using a custom multislit mask, find the comparison lamp spectra taken through your mask(s) and for the long-slit masks as the latter are used for reducing the flux calibration standard star spedtra. MODS wavelength calibrations are stable on long timescale, and any zero-point shifts are resolved in processing using night-sky lines.
3. Biases are only needed for Prism spectra. Grating spectra use the overscan regions for bias removal (future MODS prism spectra will also support overscan).
4. Spectra of flux standard stars taken during the run (or during the night).

For science targets, you want to retrieve all science spectra and target acquisition images from the archive. The acquisition images are useful for confirming target placement in the slit(s), and if attempting to derive radial velocities, can be used to resolve residual velocity zero-point shifts due to the detailed location in the slit proper during final wavelength calibration.

All of these data should be copied into your Raw/ data directory. We find it helpful to divide up the raw data by type. For example, for reducing long-slit grating spectra, our Raw/ directory would start out looking like this:

```
Raw/  Flats/
      Comps/
      Stds/
      Object1/
      Object2/
      ...
```

This lets us keep similar calibration and science data together and not all mixed together. We then put copies of the final 2D reductions in the top level of the Raw/ directory where they can be easily found.

5.3 Create a Normalized Color-Free Pixel Flat

For this procedure you will need to have taken standard slitless spectral flats of the internal continuum lamps. If for some reason you do not have these data, you can use any calibration data taken within a few months of your observations in the LBT archives. In practice, MODS flat fields have proven to be remarkably stable run-to-run.

First you must bias-correct the raw spectral flats and remove the overscan region, then median combine the flats, repair bad pixels, and create normalized pixel flats. The steps are:

Red flats:

```
modsBias.py mods1r.20140122.001[0-4].fits
modsMedian.py mods1r.20140122.001[0-4]_ot.fits rflat_med.fits
modsFixPix.py rflat_med.fits rflat_fix.fits
modsPixFlat.py rflat_fix.fits pixflat_m1r.fits
```

Blue flats:

```
modsBias.py mods1b.20140122.002*.fits
modsMedian.py mods1b.20140122.002[0-4]_ot.fits bclr_med.fits
modsMedian.py mods1b.20140122.002[5-9]_ot.fits bug5_med.fits
modsAdd.py bclr_med.fits bug5_med.fits bflat_med.fits
modsFixPix.py bflat_med.fits bflat_fix.fits
modsPixFlat.py bflat_fix.fits pixflat_m1b.fits
```

This gives us normalized color-free pixel flats that we need to process all of the other raw calibration and science images.

Note that for the blue channel we take two types of slitless flat fields, one through a clear filter, the other through a UG5 red-blocking filter. The UG5 flats are used to boost the signal at the far blue/UV end of the spectral range without saturating the CCD at the red end of the blue channel range.

5.4 OTF Process the 2D Calibration and Science Images

The next step is to reduce the remaining raw calibration and science image with the pixel flats created above.

This step uses the `modsProc.py` script to bias correct, trim, flat field, and fix bad pixels to create OTF ("overscan, trim, and flat") processed images that are the primary input to the modsIDL pipeline. An additional feature of `modsProc.py` for red-channel spectra is that it flips the spectra about the Y (vertical) axis so that wavelengths are increasing (blue to red) with increasing X pixel coordinate, like the blue spectra.

For example, to OTF process a series of blue dual-grating spectra using a blue pixel flat with bad pixel repair, the command would be:

```
modsProc.py -b mods1b.20140506.003[5-9].fits bpixFlat.fits
```

This bias subtracts, flat fields, and fixes bad pixels in five (5) images 0035 thru 0039. Unlike older version of modsProc.py, version 2 of the package allows multiple raw images on the command line. The output of the example above would be 5 images with names like

```
mods1b.20140506.0035_otf.fits
mods1b.20140506.0036_otf.fits
...
```

The `_otf` suffix indicates it is an OTF-processed spectrum.

Some people find it useful to create shell scripts to run modsProc.py over groups of images that share flat fields if a large number of images with non-sequential filenames must be processed.

This processing is done for all comparison lamps, slit flats, standard stars, and science images.

5.5 Combine Calibration Spectra

Once individual bias and flat-field corrected OTF images are obtained for the comparison lamps and slit flats, each set needs to be combined for use by the modsIDL pipeline.

For comparison lamps, we add the individual lamp spectra together into a single 2D spectrum for each channel using modsAdd.py. For example, if the reduced 2D spectra of the Hg(Ar), Xe+Kr, and Ar comparison lamps for the blue grating mode and a long slit mask are in images mods1b.20140203.0014_otf.fits through .0016_otf.fits, you'd use the command:

```
modsAdd.py mods1b.20140203.001[4-6]_otf.fits comp_m1b_ls.fits
```

to create the combined comp_m1b_ls.fits file used by the pipeline. Similarly, if reducing data for a custom multi-slit mask, you'd combine files like these (here 3 comparison lamp spectra taken with the red grating channel):

```
modsAdd.py mods1r.20140524.002[1-3]_otf.fits comp_m1r_id514234.fits
```

Note that the format of the combined comparison lamp 2D spectrum filename is:

```
comp_<instID>_<maskname>.fits
```

where `<instID>` includes the instrument and channel used in abbreviated form, thus `m1r` = MODS1 Red channel and `m2b` = MODS2 blue channel, etc., and `<maskname>` is either the value of the `MASKNAME` header keyword if using a custom multi-slit mask or "`ls`" (for "long-slit") if using one of the facility long-slit masks (e.g., the LS5x60x1.0 slit mask). Adopting this uniform filename convention will make for shorter, easier to read and type filenames later in the reduction process.

Slit flats (spectra of continuum lamps taken through long-slit or custom multislit masks) are not actually used as flat fields, but rather are used to map the spectral traces cast by each mask aperture. In principle, we could use the information in the slit flats to compute and apply along-slit illumination corrections, fringe pattern corrects, etc., but these appear to not be indicated by MODS spectra in our experience.

Combine slit flats using modsAdd.py like for comparison lamp spectra. For example:

```
modsAdd.py mods1r.20140314.003[4-6]_otf.fits flat_m1r_id514123.fits
```

where like for comparison lamps the filename format is:

```
flat_<instID>_<maskname>.fits
```

which includes the instrument/channel information in `<instID>` and the maskname if a custom multi-slit mask or "`ls`" if using any of the facility long-slit masks in `<maskname>`.

For a dual-mode spectral reduction (e.g., dual grating spectra), you will have at least eight (8) calibration images, a long-slit comparison lamp and flat for standard stars, and a multi-slit mask comparison and flat for each of blue and red channels.

5.6 Optional Processing of Standard Star and Science Images.

After OTF processing of the raw standard star and science images, a final decision is whether you wish to perform any additional processing on the OTF images before entering the modsIDL pipeline. Examples of such additional processing is combining multiple images of

a target into a single cumulative 2D spectrum (e.g., via summation, median, or pixel-average combination), or cleaning of cosmic rays.

The primary issue is usually the desire to suppress radiation event artifacts (aka "cosmic rays") on the images. Such artifacts can make the sky subtraction difficult during subsequent pipeline processing, and can affect the final extracted spectra, so removal of such artifacts is desirable. They are especially a concern for spectra with long cumulative integration times.

Cosmic ray removal is NOT included as part of the MODS pipeline.

At a basic level, after OTF processing there are no particular differences between MODS 2D spectra and any other CCD images or 2D spectra that dictate the adoption of one method over another. Our advice is to follow your best practices for image combination and cosmic ray suppression, since the downstream modsIDL pipeline processing is agnostic about this choice **provided that those methods do not alter the size or orientation of the final 2D spectra**.

One common option is to median combine images to eliminate cosmic rays. The `modsMedian.py` program is provided that can help with this, but any median combination algorithm (e.g. in IRAF) will work as well. Issues to beware of with median combination are the introduction of artifacts into the data if the seeing or transparency is variable between images or if the object has moved substantially along the slit.

Another option is to sum spectra and then remove cosmic rays separately using a third-party cosmic ray rejection algorithm like `L.A.Cosmic` (van Dokkum 2001). Some algorithms work better than others on spectral data, and the choice of cosmic-ray identification parameters is complicated and outside the scope of this pipeline.

Some observers opt to dither targets along the slit, so it is up to them to decide how to combine (or not) their dithered spectra. A bit of guidance is to point out that at this stage you still have more-or-less raw pixels on the as-observed sampling grid, so it is best to do any data combination at the bare-pixel stage before you get deep into the pipeline where geometric effects complicate the combination of dithered images.

For standard star spectra, which have short integrations and high per-pixel signals, we have found that simply adding all of the spectra together works well since the number of cosmic rays is generally small.

Finally, if you elect to combine images, be sure to change the exposure time (if summing instead of averaging or median combining) as needed, and you need to change the airmass (SECZ header keyword) to a value appropriate to the median for the entire cumulative exposure sequence.

5.7 Custom Slit Mask Files

The modsIDL pipeline uses the MODS Mask Specification (MMS) files used to create the slit masks if you used custom multi-slit masks. Copies of these must be placed in the working directory before you start running the pipeline. The MODS pipeline will matches MMS files to the appropriate science image using the MASKNAME keyword in the header of each fits image, such that files should be named:

```
<maskname>.mms
```

The maskname should match the mask ID that is used by LBTO to identify each custom mask.

If your data were taken using one of the facility long-slit masks, a set of stock MMS files are automatically available as part of the pipeline. You do not need to secure your own copies.

5.8 Organize the 2D Spectra

The final preparation step is to cleanup and organize the 2D spectra and associated slit mask (.mms) files to prepare for modsIDL pipeline reduction.

We find it very helpful as the last step in the OTF processing step to rename the science and standard star image files to reflect the names of the objects, rather than working with the long raw date/sequence filenames created by the MODS data-taking system. We recommend you adopt simple names like:

```
ngc5548_m1r_001.fits ... ngc5548_m1r_004.fits
```

if reducing individual integrations separately, or

```
ngc1068_m1b_med.fits
```

for reducing combined 2D spectra (e.g., "med" = median combined). Similarly for standard stars, shortened names like

```
feige34_m1r.fits
bd28_m1b.fits
```

etc. While in practice these are "`_otf.fits`" images, adding this only lengthens the name without adding useful information for the by-hand processing described here. In a future automated observatory-scale pipeline adding `_otf.fits` makes more sense. We consider it optional in this context.

In general, short names are easier to work with than long names, because as the pipeline processing progresses, it creates a number of intermediate files based on these root name by adding suffixes and prefixes descriptive of the data product, which means long names at the start will become even longer (and harder to type) filenames by the end of the process. Try to keep things as simple as practical.

Once you have the final data ready, put **copies** of the images to be reduced into the `Proc/` directory and move unused raw data and copies of reduced images into the `Raw/` directory to keep them out of the way. Also have copies of any custom multi-slit mask MMS files in the `Work/` directory (not in `Proc/`!).

6 modsIDL Pipeline Processing

Once your 2D images are processed, combined, sorted, and ready for further reduction, you will need to start an IDL session to run the modsIDL pipeline. This is done by typing

```
idl
```

at the system prompt while in the working directory. All subsequent reductions in this section are performed by commands issued in the IDL command shell.

All modsIDL scripts accept optional parameters separated by commas on the command line. There are four basic types of command-line options available:

`/<option>`	for a Boolean on/off (true/false) parameter
`option=###`	for a numerical parameter
`option=[#,#,#]`	for a numerical array parameter
`option='string'`	for a string parameter

This manual assumes only a passing acquaintance with IDL. You do not need to know IDL programming or have much IDL experience to use the modsIDL routines effectively.

6.1 Create the input parameter file (`mods_plan`)

First, you need to create an input parameter file that will tell modsIDL which images are to be processed and which calibration files are associated with these images. This is done using the `mods_plan` script:

```
mods_plan,'<fileexpr>','<indir>',[planfile='<name>']
```

For example:

```
mods_plan,'*.fits*','Proc/'
```

reads the headers of all `.fits` files in the `Proc/` directory and creates an input parameter file named "`plan.par`".

The optional `planfile=` argument may be used to create this file with a different name, for example:

```
mods_plan,'*.fits*','Proc/',planfile='n5548.par'
```

creates `n5548.par` in the current directory.

`mods_plan` will label each type of image based on the contents of the `IMAGETYP` keyword it finds in the FITS headers:

FITS Header `IMAGETYP`	`plan.par` Image Type	Description
OBJECT	science	science-target spectra
FLAT	domeflat	spectral slit flats
STD	std	flux standard stars
COMP	arc	wavelength calibration lamps

The contents of the `plan.par` file may be edited later as needed to modify runtime parameters or remove images you don't want to examine, but in general you will rarely need to do this for routine reductions.

6.2 Construct Slit Maps, Wavelength Maps, and 2D Spectra (mods_reduce)

You are now ready for the first pipeline step.

The mods_reduce script computes the slit location maps for the observation's slit masks, computes a 2D wavelength map giving the wavelength associated with each raw image pixel, and creates multi-extension FITS files from the science and standard star 2D spectra. A number of optional keywords are provided to enable specialized processing. We will only describe the options most applicable to routine MODS data reduction.

Run the mods_reduce script by typing:

```
mods_reduce,<parameter file>,[/autotune, /interactive_slits,
/check_complete]
```

For example:

```
mods_reduce,'plan.par',/autotune,/interactive_slits
```

Optional Parameters:

- autotune – Automatically tune the location of the slits to allow for slight shifts based on how the mask is actually mounted in its cell. This option should be used for all MODS reductions.
- interactive_slits – Interactively adjust where the slits are located, provides a visual inspection. This is recommended as a way to make sure the slit location maps are computed correctly.
- check_complete – Run mods_reduce without stopping if encountering already processed files. This will not overwrite already processed files, but merely checks that all files are processed.
- justcalib – just reduce the calibration images (comparison lamps and slit flats) listed in the parameter file.
- justsci – just reduce the science images listed in the parameter file. This is useful if calibration images are already reduced.
- juststd – just reduce the standard star observations listed in the parameter file.
- clobber – allow overwrite of any existing output file leftover from a previous reduction run. By default, mods_reduce runs in "no-clobber" mode to protect data.
- calibclobber – only allow overwrite of existing slit location maps and wavelength map files.
- sciclobber – only allow overwrite of existing science and standard star images.

The mods_reduce script creates a number of output files in the working and Science/ directories.

In the working directory you will find the slit-position and wavelength maps associated with the science and standard star spectra:

slits-<name>.fits	slit position maps associated with <name>.fits
wave-<name>.fits	wavelength maps associated with <name>.fits

`wave-<name>.sav`	copy of the wavelength maps for re-reduction
`wave-<name>.ps`	PostScript plots of the wavelength solutions

The `.sav` files are copies to be used if future reductions are performed in the working directory (these prevent `mods_reduce` from unnecessarily repeating intermediate calibration steps).

Examples of slit location maps are shown in Figure 2 thru Figure 4.

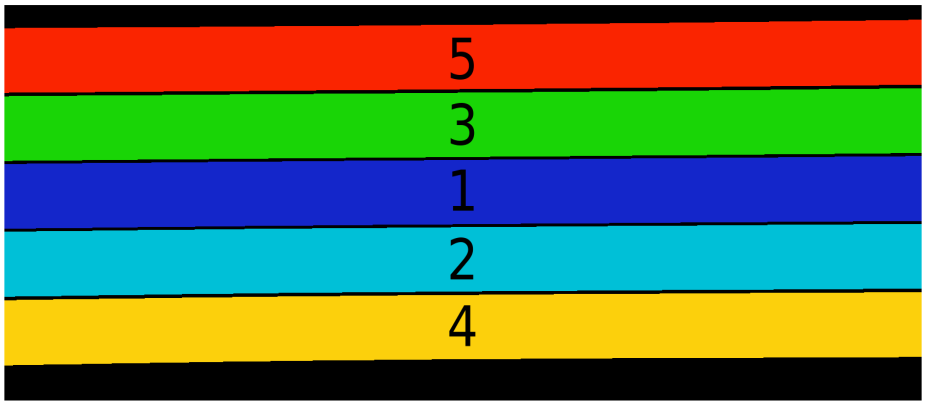

Figure 2: Slit location map for a grating spectra using a facility long-slit mask.

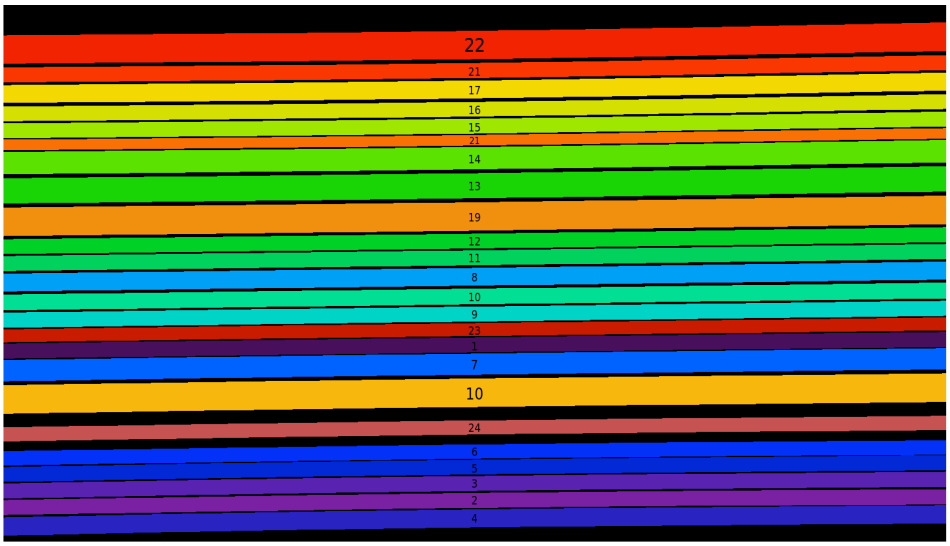

Figure 3: Slit location map for a multi-object grating spectrum.

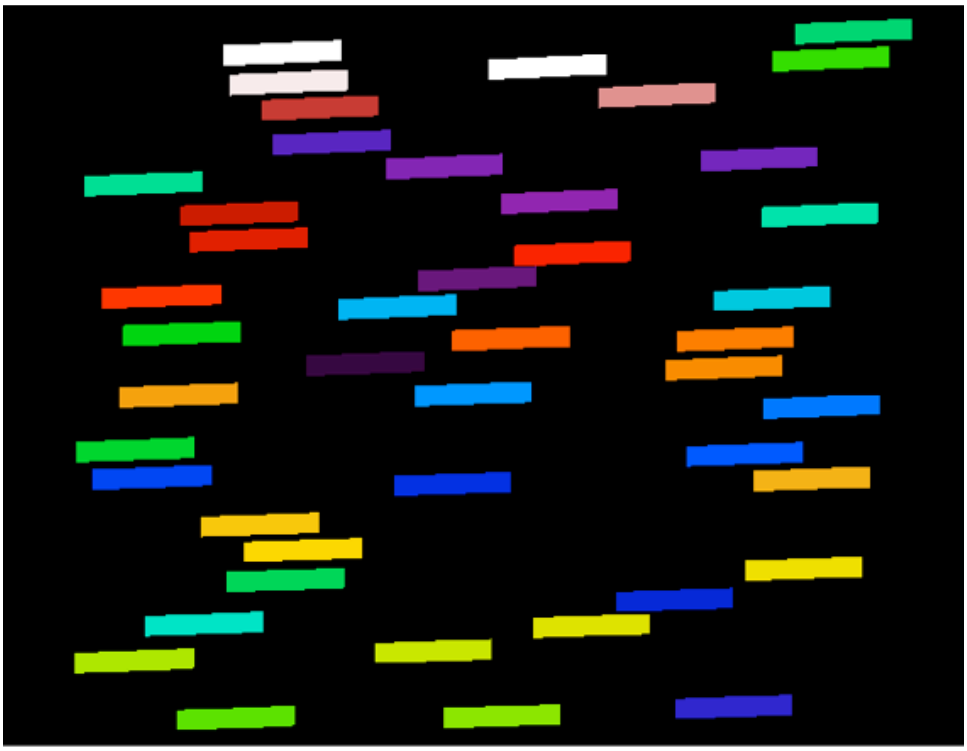

Figure 4: Slit location map for a multi-object prism spectrum.

Examples of 2D wavelength maps are shown in Figure 5 through Figure 7. These show how each pixel in each slit maps into wavelength. These are still "raw" unrebinned pixels preserving the original image pixilation - we do not rectify 2D images to avoid introducing unwanted artifacts into the data.

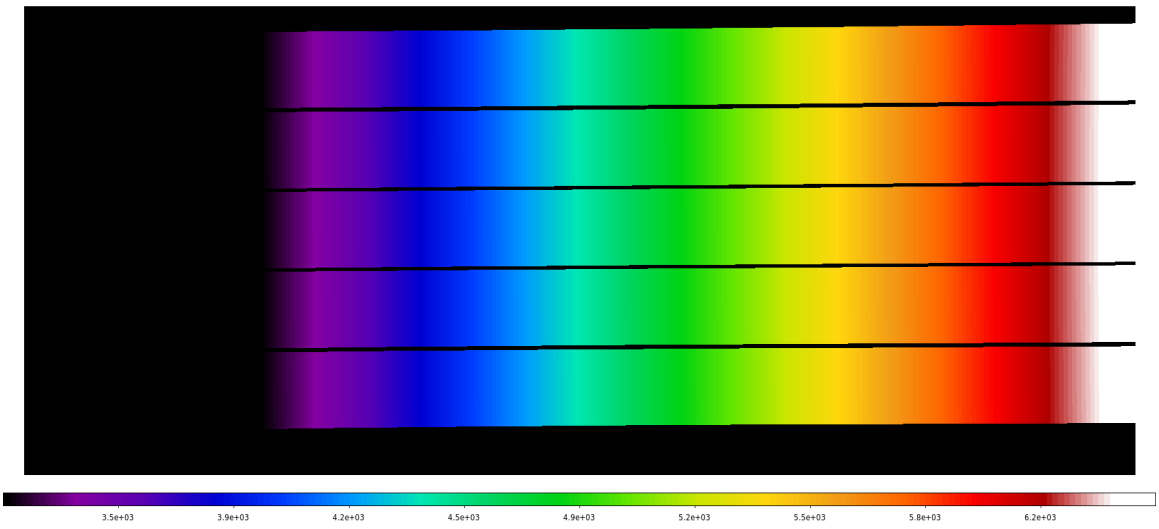

Figure 5: Wavelength map for a blue long-slit grating spectrum.

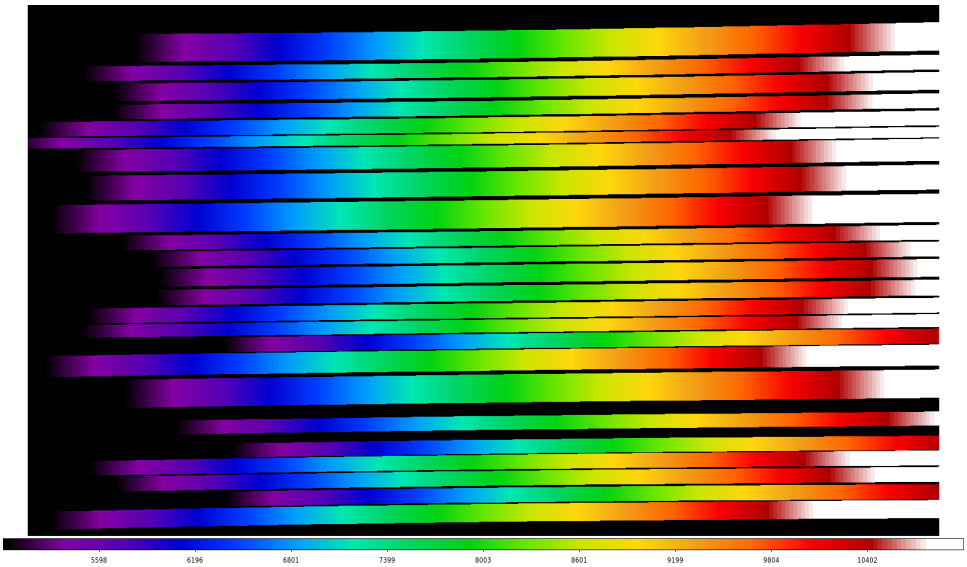

Figure 6: Wavelength map for a red multi-object grating spectrum.

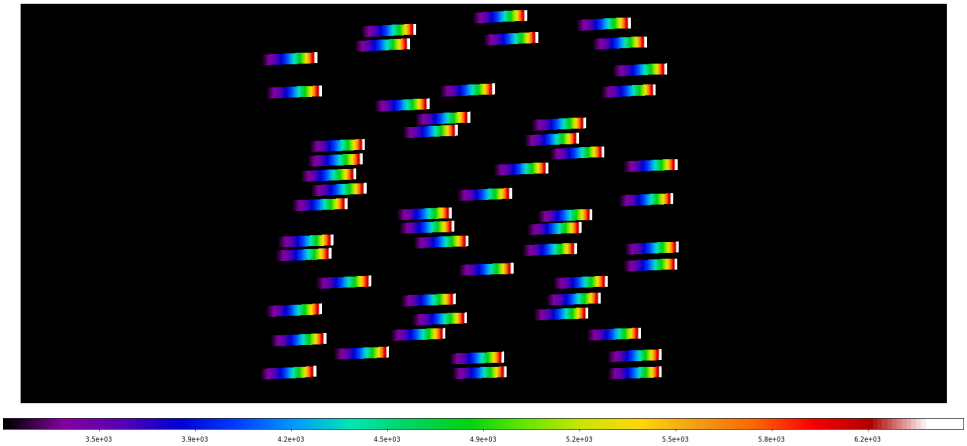

Figure 7: Wavelength map for a blue multi-object prism spectrum.

The `Science/` directory is where `mods_reduce` writes the science and standard star spectra ready for further processing. These will be given names like:

`sci-<name>.fits.gz`	2D science target spectra
`std-<name>.fits.gz`	2D flux calibration standard star spectra

There will be one file for every standard star and science image listed in the `plan.par` file. They are written as gzip-compressed Multi-Extension FITS files that contain the processed data. There are two image extensions:

0 – Science image ready for sky subtraction and 1D spectrum extraction
1 – Inverse-variance weights of the science image pixels

6.2.1 Alternative empirical wavelength maps (`mods_empwave`)

If contemporaneous wavelength calibration comparison lamp spectra are not available for a given observing run, or when examining prism-mode data, 2D wavelength maps can be generated using the empirical mapping of the MODS channels:

```
mods_empwave,<slit image>,[filename='outName']
```

For example:

```
mods_empwave,'slits-flat_mb1_LS.fits',
             filename='wave-comp_m1b_emp.fits'
```

The empirical mapping is remarkably good for most applications, and a good wavelength solution can be obtained with additional correction of the wavelength scale using night-sky emission lines. It will not, however, return the higher precision possible with contemporaneous wavelength comparison lamp spectra.

6.2.2 Adjusting the 2D Slit Location Maps (`mods_slitadjust`)

If you find there are issues with the locations of the slit edges found on the first pass with `mods_reduce`, these may be adjusted interactively using `mods_slitadjust`:

```
mods_slitadjust,<slitImage>,<referenceImage>,[apertures=[#,#],/Raw]
```

For example:

```
mods_slitadjust,'slits-comp_m1b_ls.fits','Proc/comp_m1b_ls.fits',
                apertures=[2,4],/Raw
```

If only certain apertures need adjustment, they can be specified on the command line with the `apertures=[]` array. The optional `/Raw` argument needs to be used if a raw image (e.g., an OTF file) is used as a reference as modsIDL works with a transpose (i.e., 90° rotation) of the original 2D images (the transpose aligns the data in memory for optimal processing speed when performing summations or fits along the slit direction).

IMPORTANT NOTE:

> The pipeline for MODS prism mode is still a work in progress. As a result you will likely see some errors. In particular, you can expect messages from `long_slits2x` which will likely note that prism slits are goofy. This is a result of the shorter span of the 2D spectrum. Cleaning up the prism reductions is a work in progress.

6.3 Processing of Standard Stars (`mods_standard`)

After the running `mods_reduce` to produce wavelength and slit maps and initial processing of each image, the standard stars should be processed to create a spectral response curve. Reduction of standards is performed by running:

```
mods_standard,[planfile='<name>', paramfile='<name>',
outname='<name>',/onlyplan,/logprofile,/no_sky_sub]
```

The `mods_standard` script will parse the contents of the `Science/` directory and subsequently run sky subtraction, extraction, and fluxing routines on the standard star spectra. Given standards are fairly uniform, most users will find executing the command:

```
mods_standard
```

to be sufficient for running this task, although some may wish to add `/logprofile` to change the profile plot to a logarithmic scale.

We will here describe basic usage of this script and the scripts called by `mods_standard`. For additional details on each individual script, please see the appendix.

6.3.1 Create and edit parameter files (`mods_editparfile`)

To generate the appropriate commands to run, `mods_standard` must first parse the `Science/` directory and each file therein. It will then create two parameter files:

`std.par`

`stdparam.par`

unless other names were supplied using `planfile='<name>'` and `paramfile='<name>'`. These files contain the relevant information gleaned from the headers that will be used in processing the standard stars. Following a header that defines how the file is read, the `std.par` file lists for each file:

Tag: A tag describing each type of observation. These should all read `STD`.

Filename: The name of each file.

Instrument: The instrument code read from the header of each file (MODS1B, MODS1R, MODS2B, MODS2R).

Wavefile: The associated wavelength file as assigned by `mods_reduce` and saved in the header.

Slitfile: The associated slit map, as assigned by `mods_reduce` and saved in the header.

Std_name: The name of the standard file from the `CALSPEC` library that will be used to create a response curve. This is chosen based on the name of the standard listed in the header. If the name is not recognized, you will be prompted with a list of recommended standard file names.

Apertures: The apertures to be processed. As standard stars are taken through the wide-slit, which has only one aperture, they should all read `[1]` .

Maskname: The name of the mask associate with the observation.

Following a header that defines how the file is read, the `stdparam.par` file lists for each file:

Tag: A tag describing each type of observation. These should all read `STD`.

Filename: The name of each file.

Boxcar: A Boolean flag (0/1) that uses boxcar smoothing for the sky-subtraction.

Centerline: Central wavelength, in angstroms, for the spectral window that will be used in selecting sky-regions and spectral extractions. Default is the central wavelength for each grating.

Centerwidth: Width, in angstroms, of the spectral window that will be used in selecting sky-regions and spectral extractions.

Centersum: Number of pixels summed to create the slit profile.

Trim_top: Number of pixels to trim off the top of the slit when searching for maxima and display parameters.

Trim_bot: Number of pixels to trim off the bottom of the slit when searching for maxima and display parameters.

Mask_lines: A Boolean (0/1) parameter to mask standard strong emission lines.

The files can be edited using standard text editors, or using the `mods_editparfile` script.

6.3.2 Sky Subtraction (`mods_skyfit2d_singlechan`)

Sky subtraction of MODS spectra is accomplished by fitting the night-sky emission lines with a B-spline (aka "Basis spline") in the unrebinned pixel space (see Kelson 2003 for the classic implementation of this method). Using the two-dimensional wavelength image, a B-spline is fit to the dispersion direction as a function of the wavelength while a low order polynomial is fit along the slit direction (orthogonal to dispersion).

Two graphical display panels, examples shown below, are used to guide the selection of sky areas.

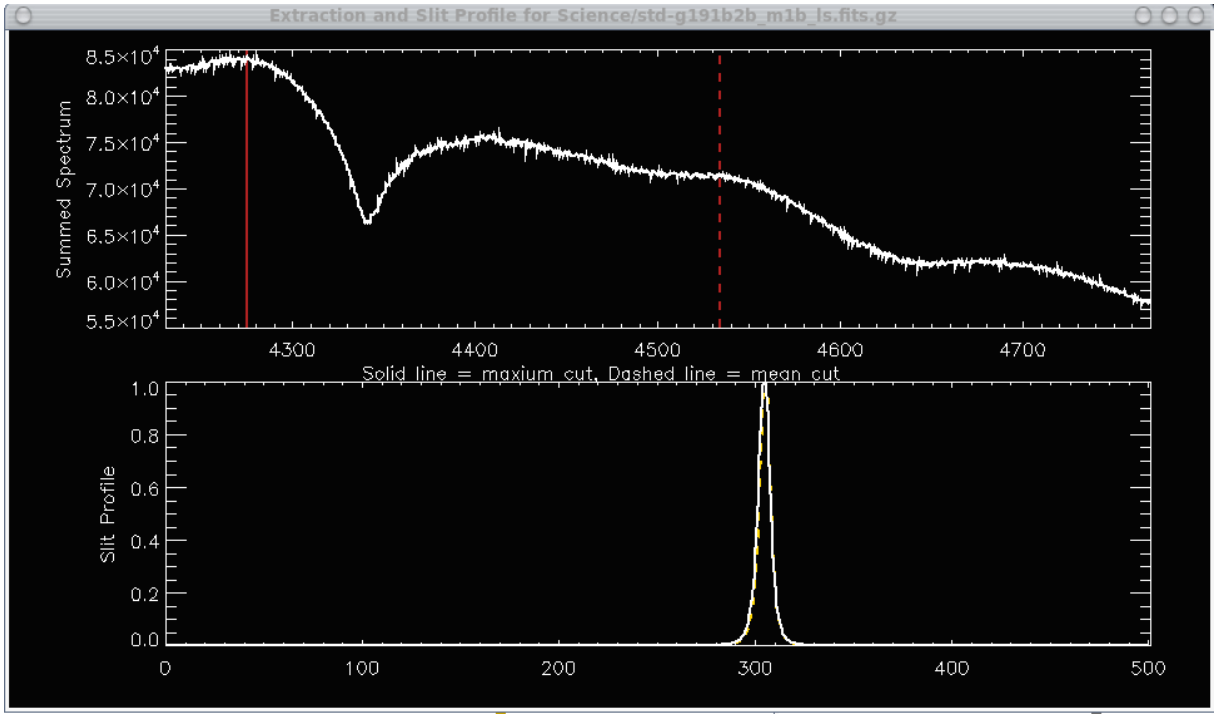

Figure 8: Example of a diagnostic plot produced by the `mods_skyfit2d_singlechan` procedure as part of `mods_standard` that is used to interactively select object and sky regions for computing "clean" sky fits.

In the first window, the top row initially shows the average spectrum for the entire slit in a given spectral range (shifted using the by centerline parameter input). The brightest portion of the spectrum and the median value of the spectrum in the window are selected to make profiles of the slit, as marked by the solid and dashed lines respectively. The bottom panel shows the intensity profile of the aperture averaged over the extraction region shown in the bottom panel, solid white for the maximum extraction and dashed yellow for the median extraction.

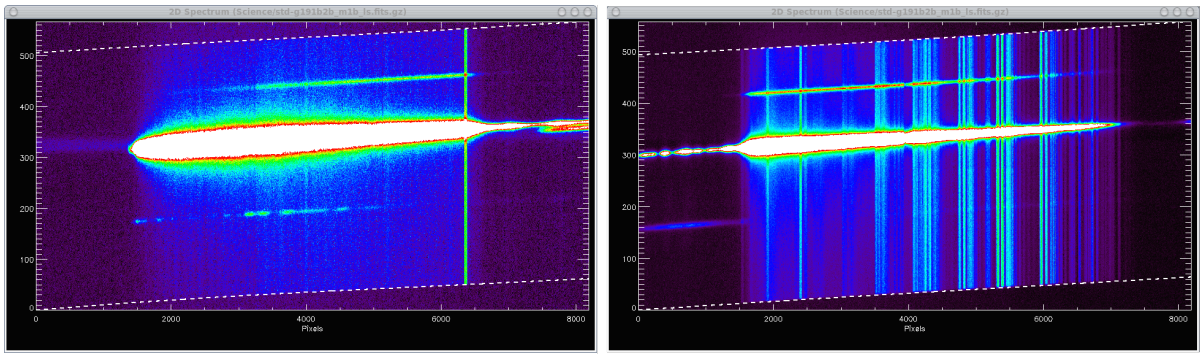

Figure 9: 2D spectral plots created by `mods_skyfit2d_singlechan` to help interactively select sky and object regions for a standard star spectrum.

In the second window the two-dimensional spectrum is displayed. The upper and lower boundaries of the aperture are displayed as dashed white lines.

After an aperture is presented, you will be asked to click on, first, a lower bound and, secondly, an upper bound for a sky to used as a sky background. This is done by left clicking the mouse on the slit profile where each boundary should be drawn. Once the boundaries have been selected, the region marked will be highlighted in red in the slit profile, and the corresponding boundaries will be displayed on the 2D spectral plot. You will be asked to confirm this selection, answer 'y', 'n', 'nosky, or 'abort. Entering no (`n`) will allow you to re-mark the selection. Entering abort will immediately exit the program. Entering nosky will indicate that there is no suitable sky in the aperture and apply the solution from an indicated sky-slit, this requires a sky-slit to have been designated and is not applicable for standard stars. Finally, entering yes (`y`) will allow you to move forward.

You will then be asked if you would like to add an additional sky region to be included in the modeling. Old sky designations will remain marked in a darker red (firebrick) while the current sky selections will be in bright red. This process will continue until you indicate that no additional sky regions should be included.

For best results, you may want to avoid edges of the slit so that no severe drop-off from the slit is seen, i.e., you are not including inter-aperture pixels, as this will cause ringing in the sky-lines. Currently, the edges are brought in a bit from the defined slit edges, but there can be a little variation that may require additional masking.

6.3.3 Extraction and Calibration (`mods_fluxstand_singlechan`)

Finally, `mods_standard` uses the sky-subtracted standard star observations to construct a spectral response curve for flux calibration using `mods_fluxstand_singlechan`.

The `mods_fluxstand` procedure creates the same plots that were created by `mods_skyfit2d`. However, in this case, rather than selecting sky regions you are marking the region to be extracted, and the top plot shows a portion spectrum to be extracted.

In addition to the extracted spectrum of your standard star (see 7.3 for details), `mods_fluxstand` will save two files, `std-<name>_m1b_ls_ssf.fits` and `std-<name>_m1r_ls_ssf.fits`. These are the spectral sensitivity functions that will be applied as the flux calibration when extracting 1D spectra.

6.4 Processing of Science Fields (mods_science)

After running mods_reduce and mods_standard we are ready to reduce the science images. Reduction of science objects is performed by running:

```
mods_science,[/dual, /redonly, /blueonly, planfile='<name>',
paramfile='<name>', infofile='<name>', outname='<name>', /onlyplan,
/logprofile, /no_sky_sub, /clobber, /linemask, /em_line, /logprofile,
/man_scale, /planonly]
```

The mods_science script will parse the contents of the Science/ directory and subsequently run sky subtraction, extraction, and fluxing routines on the science spectra. It behaves very similar to mods_standard, however, as science observations are less uniform than standard stars, more options are available.

6.4.1 Create and edit parameter files (mods_editparfile)

To generate the appropriate commands to run, mods_science must first parse the Science/ directory and each file therein. It will then create three parameter files:

```
sci.par
sciinfo.par
sciparam.par
```

unless other names were supplied using planfile='<name>', infofile='<name>', and paramfile='<name>'. These files contain the relevant information gleaned from the headers that will be used in processing the science files. The files can be edited using standard text editors, or using the mods_editparfile script.

sci.par: Following a header that defines how the file is read, a line for each file lists:

Tag: A tag describing each type of observation. These should all read SCI.

Filename: The name of each file.

Instrument: The instrument code read from the header of each file (MODS1B, MODS1R, MODS2B, MODS2R).

Wavefile: The associated wavelength file as assigned by mods_reduce and saved in the header.

Slitfile: The associated slit map, as assigned by mods_reduce and saved in the header.

Std_name: The name of the standard file from the CALSPEC library that will be used to create a response curve. This is chosen based on the name of the standard listed in the header. If the name is not recognized, you will be prompted with a list of recommended standard file names.

Apertures: The apertures to be processed. As long-slit exposures have only one aperture and should read [1]. All apertures for multislit masks will be processed if this parameter reads [ALL], or the desired individual apertures may be selected, e.g., [1,2] for slits 1 and 2.

Skyslit: The number of the aperture you wish to use as a sky slit, it any. This aperture

will be fit before the others to create a general sky fit that can be applied to any other slit if desired. See 7.1 for more details.

Redshift: Redshift of the science target.

Dual: Default is `0`. Set to `1` if you want to extract red and blue spectra together.

Maskname: The name of the mask associate with the observation.

`sciinfo.par`: Following a header that defines how the file is read, a line for each file lists:

Tag: A tag designating fields as information about the exposure, should read `INFO`.

Filename: The name of each science file to process.

Obsdate: The date of the observation, in YYYY-MM-DDTHH:MM:SS.SSS format.

Object: Name of the science target.

ExpTime: Duration of the exposure. This number should be edited to reflect any summing or median combining of science exposures that was performed.

ObsMode: Channel mode of the observations: `Dual`, `Red`, or `Blue`.

Maskname: The name of the mask associate with the observation.

`sciparam.par`: Following a header that defines how the file is read, a line for each file lists:

Tag: A tag designating fields as parameters for extraction, should read `PARAMS`.

Filename: The name of each science file.

Boxcar: A Boolean flag (0/1) that uses boxcar smoothing for the sky-subtraction.

Centerline: Central wavelength, in angstroms, for the spectral window that will be used in selecting sky-regions and spectral extractions. Default is the central wavelength for each grating. It is recommend that a more intentional wavelength is designated appropriate to the science case. For an HII region example, `4861` for blue and `6563` for red extractions allows us to use an extraction width encompassing the majority of the nebular emission.

Centerwidth: Width, in angstroms (Å), of the spectral window that will be used in selecting sky-regions and spectral extractions. For nebular emission-line spectra, a width of a few Å is sufficient.

Centersum: Number of pixels summed to create the slit profile. Default is 10.

Trim_top: Number of pixels to trim off the top of the slit when searching for maxima and display parameters.

Trim_bot: Number of pixels to trim off the bottom of the slit when searching for maxima and display parameters.

Mask_lines: A Boolean (0/1) parameter to mask standard strong emission lines.

7 Details on Sky Subtraction and Extraction scripts

7.1 2D Sky Subtraction (`mods_skyfit2d`)

Sky subtraction of MODS spectra is accomplished by fitting the night-sky emission lines with a B-spline (aka "Basis spline") in the raw-image pixel space (see Kelson 2003 for the classic implementation of this method). Using the two-dimensional wavelength image, a B-spline is fit to the dispersion direction as a function of the wavelength while a low-order polynomial is fit along the slit direction (orthogonal to dispersion).

For the best sky subtraction, your slits should be long enough that every slit has regions of sky free of stars or emission-line regions. At the same time, you should select a sky-extraction region that is as close to your object as practical.

Run `mods_skyfit2d` by typing:

```
mods_skyfit2d,<red_image>,<blue_image>
```

For example:

```
mods_skyfit2d,'Science/sci-ngc5194_m1r.fits.gz',
              'Science/sci-ngc5194_m1b.fits.gz',
              outname='M101'
```

Optional Parameters:

`skyslit` – The number of the aperture you wish to use as a sky slit. This aperture will be fit before the others to create a general sky fit that can be applied to any other slit if desired. The sky slit solution will also be applied to fill in the sky under the strongest nebular emission lines that often are difficult to mask (Hα, Hβ, Hγ, Hδ, [O III] 5007, [O III] 4959, [O II] 3727, [S III] 9069, [S III] 9532, [S II] 6716,31, [N II] 6548,84).

`lw` – linewidth in Angstroms to use in blanking strong lines when employing a sky slit. The default value is 5Å.

`z` – The redshift of the object. This will be used to shift the wavelength windows used.

`convbeam` – Convolution beam used to smooth the sharpness of the night sky lines. Given as an array [dispersion axis, non-dispersion axis]. The default is `convbeam=[2,1]`.

`centerLine` – Boundaries for where in wavelength space `mods_skyfit2d` will search for emission lines to center on. Default value is `centerLine=[4850,5070,6650,6800]`.

`clobber` – overwrite output files if they already exist.

`noPlotAp` – Suppress the display of apertures you are subtracting sky from.

`trim_??` – Automatic regions to mask at the top and bottom of every slit when fitting the sky. Replace `??` with: rt = red top, rb = red bottom, bt = blue top, bb = blue bottom.

For the best sky subtraction, your slits should be long enough that regions of sky clear of stars or emission-line regions appear in every slit. At the same time, you should select a sky extraction region that is as close to your object as practical. However, this is not always

feasible. In cases where "clean" sky is not obtained in each slit, a dedicated sky slit can be employed, however, you should be aware of that *where* a sky slit is placed on the mask is important as optical aberrations in the MODS optics are a serious concern at the extreme edges of the field of view. These aberrations (primarily astigmatism from the off-axis paraboloid collimator mirrors) have the effect of distorting the night-sky emission-line profiles, making the sky fit less robust and resulting in noisy sky subtractions. Including dedicated sky slits on multi-object masks is a good idea, but requires planning and care. This is most important if your targets are extended (like HII regions in galaxies) or point sources in crowded fields (e.g., low-latitude star fields).

`mods_skyfit2d` creates two Multi-Extension FITS files that contain the processed data. The extensions are:

0 – Sky subtracted image.
1 – Inverse-variance weights of the science image pixels
2 – Binary FITS table containing info on regions masked for each slit.
3 – Two dimensional image of the fit to sky emission.

If a sky slit was fit, then the images will have these additional extensions:

4 – Two dimensional image of the fit to sky emission from the sky slit.
5 – Sky subtraction using only the sky fit from the identified sky-slit.

A note of caution: You will want to be careful around the quadrant jumps as small shifts may exist here that can throw off the fit. For single channel data, the same procedure can be accomplished using the command `mods_skyfit2d_singlechan`. For prism mode, you will follow the same procedure but employ `mods_skyfitprism`. Each of these routines use the same keywords and parameters, although in `mods_skyfit2d_singlechan` the keywords do not have the associated channel ID (i.e., `wave_red` becomes simply `wave`)

Examples of two dimensional spectra before and after sky subtraction are shown in Figure 10 through Figure 13.

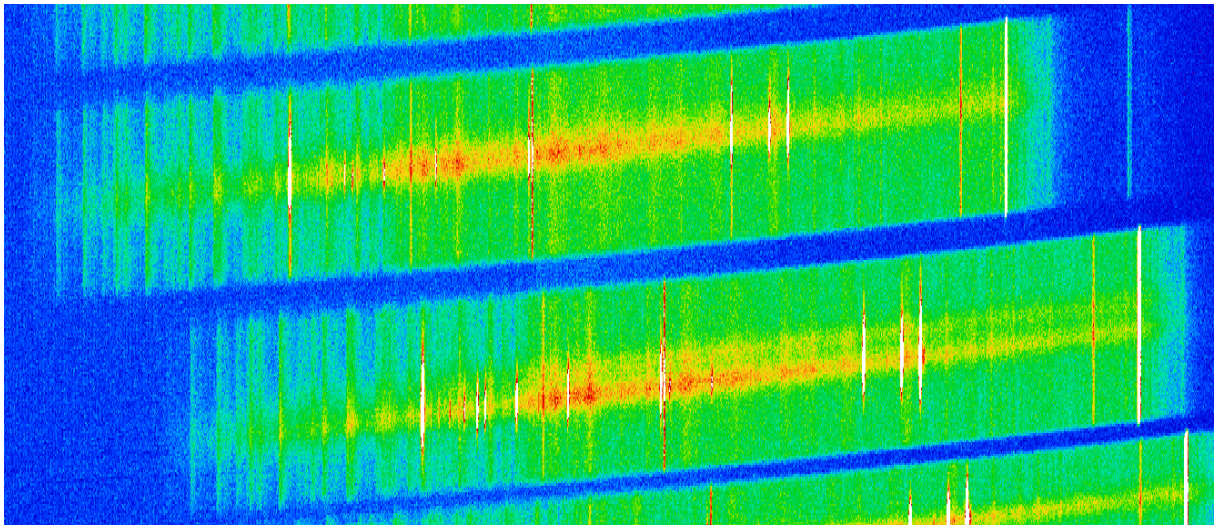

Figure 10: A portion of a blue grating multi-object spectrum before sky subtraction.

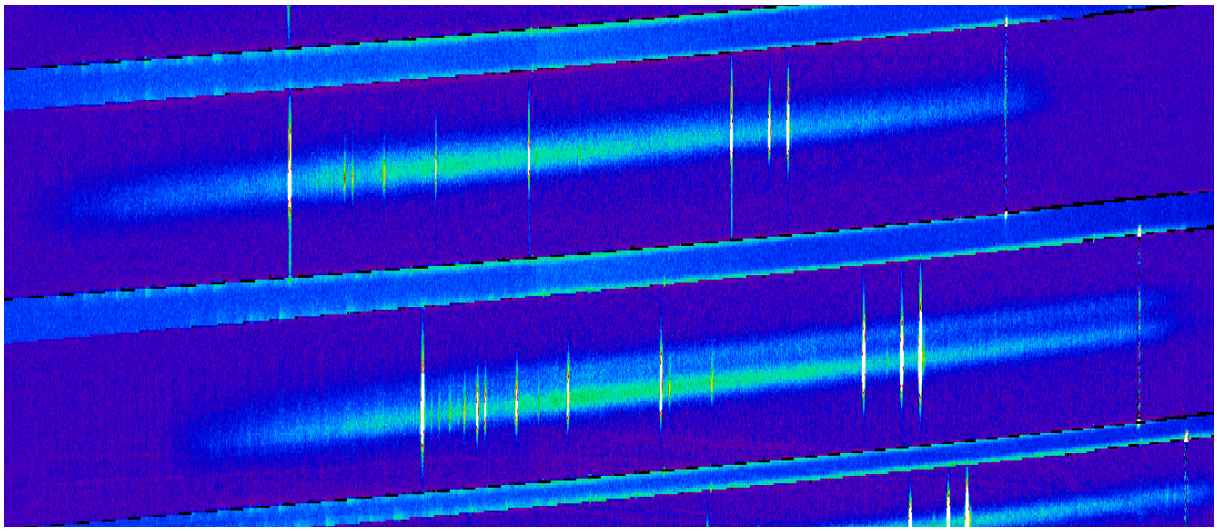

Figure 11: A portion of a blue grating multi-object spectrum after sky subtraction. The underlying HII regions emission lines are readily visible.

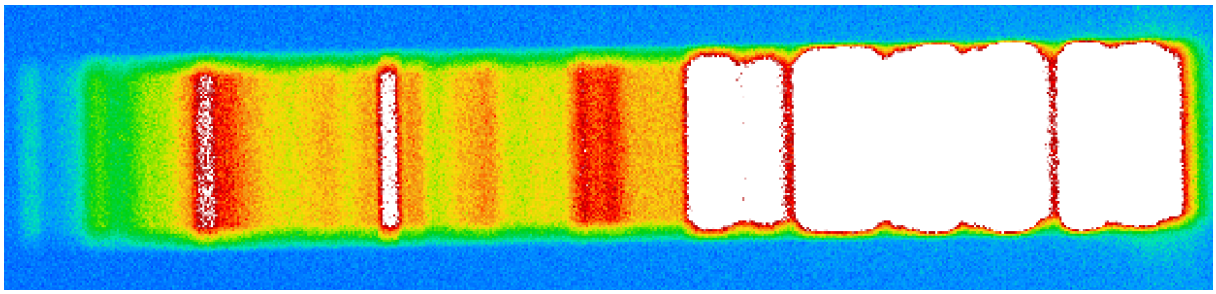

Figure 12: A portion of a red prism multi-slit spectrum before sky subtraction.

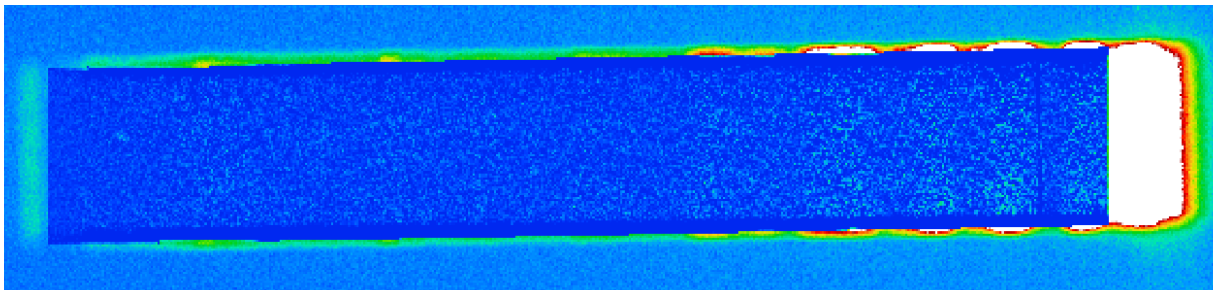

Figure 13: A portion of a red prism multi-slit spectrum after sky subtraction.

Finally, be aware that sky subtraction in the red channel can be difficult due to the presence of the many emission lines red-ward of about 7000Å from atmospheric OH Meinel Bands. These closely-spaced emission lines can leave residuals in the sky subtraction, particularly if the applied sky was not locally selected.

7.2 Measuring Flux Calibration Response Curves (`mods_fluxstand`)

Using the sky-subtracted standard star observations we can construct a spectral response curve for flux calibration using `mods_fluxstand`:

```
mods_fluxstand,<red_image>,<blue_image>,std_name=<name>
```

Ex: `mods_fluxstand, 'Science/std-hz44_m1r_2ds.fits', 'Science/std-hz44_m1r_2ds.fits', std_name='hz44_stis_001'`

where `std_name` is the name of one of the standard files in the XIDL library database. Standards stars have been selected from the HST Primary calibration star list. If you run `mods_fluxstand` without giving the `std_name` keyword, you will be instructed to use it and given a list of recommended standard files. When fitting the standard templates, we mask strong Balmer lines and telluric features.

The HST calibration star files for std_name are stored in the

xidl/Spec/Longslit/calib/standards/calspec

folder and are part of XIDL. In the example above, 'hz44_stis_001' corresponds to the file 'hz44_stis_001.fits.gz' in the calspec/ folder.

Optional Parameters:

`outname` – desired output name. This string will be appended with '1Dsub_color_'.

`wave_blue` – name of the blue wavelength image you wish to use. Default value is `wave-comp_m<1|2>b_<maskname>.fits`

`wave_red` – name of the red wavelength image you wish to use. Default value is `wave-comp_m<1|2>r_<maskname>.fits`

`centerLine` – Boundaries for where in wavelength space `mods_skyfit2d` will search for emission lines to center on. Default is value is `centerLine=[4850,5070,6650,6800].`

`blue_slits` – name of the blue slit image you wish to use. Default value is `slits-flat_m<1|2>b_<maskname>.fits`

`red_slits` – name of the red slit image you wish to use. Default value is `slits-flat_m<1|2>r_<maskname>.fits`

The `mods_fluxstand` procedure creates the same plots that were created by `mods_skyfit2d`. However, in this case, rather than selecting exclusion regions you are marking the area to be extracted, and the top plot shows a portion of the extracted spectrum.

In addition to the extracted spectrum of your standard star (see 7.3 for details), `mods_fluxstand` will save two files, `redsensfunc.fits` and `bluesensfunc.fits`, in the current working directory. These are applied as the flux calibration when extracting 1D spectra.

7.3 Extracting 1D Calibrated Spectra (`mods_extract1d`)

Extraction of one dimensional MODS data uses traces derived from the mapping of the optical system updated for atmospheric distortions using information in the header of the FITS files (e.g., HA, EXPTIME). To extract spectra, load IDL from the working directory and run mods_extract1d.pro:

`mods_extract1d,<red_image>,<blue_image>`

For example:

```
mods_extract1d,'Science/sci-ngc5194f2_m1r_2ds.fits',
               'Science/sci-ngc5194f2_m1b_2ds.fits',
               apertures=[1,2],outname='ngc5194f2',...
```

Optional Parameters:

`z` – The redshift of the object. This will be used to shift the wavelength windows used.

`apertures` – Array giving the aperture you wish to work on. For example, `apertures=[1,3,5]` will process the first, third, and fifth science slits listed in the associated MMS file.

`outname` – desired output name. This string will be appended with '`1Dsub_color_`'.

`wave_blue` – name of the blue wavelength image you wish to use. Default value is `wave- m<1/2>b_NeXeAr_<maskname>.fits`

`wave_red` – name of the red wavelength image you wish to use. Default value is `wave- m<1/2>r_NeXeAr_<maskname>.fits`

`centerLine` – Boundaries for where in wavelength space `mods_skyfit2d` will search for emission lines to center on. Default value is `centerLine=[4850,5070,6650,6800]`.

`blue_slits` – name of the blue slit image you wish to use. Default value is `slits- m<1/2>b_ill_<maskname>.fits`

`red_slits` – name of the red slit image you wish to use. Default value is `slits- m<1/2>r_ill_<maskname>.fits`

`clobber` – overwrite existing output files.

`noPlotAp` – Suppress the display of apertures you are subtracting sky from.

`trim_??` – Automatic regions to mask at the top and bottom of every slit when fitting the sky. Replace **??** with: rt = red top, rb = red bottom, bt = blue top, bb = blue bottom.

`blue_cal` – name of the blue-channel flux calibration response curve you wish to apply. The default value is `bluesensfunc.fits`

`red_cal` – name of the red-channel flux calibration response curve you wish to apply. The default value is `redsensfunc.fits`

`scale_minmax` – use min/max scaling when displaying the 2D spectra.

`force_blue=[dlam,slope1,slope2]` – force the flexure correction to use the given parameters for the entire blue detector. To suppress any flexure correction altogether, use `force_blue=[0,0,0]`.

`force_red=[dlam,slope1,slope2]` – force the flexure correction to use the given parameters for the entire red detector. To suppress any flexure correction altogether, use `force_red=[0,0,0]`.

`illumination_corr` – extract illumination curves from the slit flats. Not fully implemented at this stage.

Similar to `mods_skyfit2d` and `mods_fluxstand`, the `mods_extract1d` script uses the same two windows for marking which pixels are extracted.

The uncalibrated 1D spectra will be stored in MEF files named:

`<type>-<name>_<channel>_r1d.fits`

with the following image extensions:

0 – sky-subtracted object spectra in raw unrebinned pixels
1 – error spectra in raw unrebinned pixels
2 – sky spectra in raw unrebinned pixels
3 – wavelength map for the spectra
4 – binary FITS table containing the extraction parameters

Within each FITS extension (0-3) the uncalibrated 1D spectra are stored as "row-stacked" spectra with raw unrebinned pixels along the X (column) axis. If you extracted spectra from N regions, the row-stacked spectra will have N+1 rows, organized as follows:

Row 1: unused placeholder (data value 1.0 in all pixels)
Row 2: uncalibrated 1D spectrum (ADU per pixel) of extraction region 1
Row 3: uncalibrated 1D spectrum of extraction region 2
...
Row N+1: uncalibrated spectrum of extraction region N

The calibrated 1D extracted spectra are written to a second MEF file with the name:

`<type>-<name>_<channel>_x1d.fits`

with the following FITS image extensions:

0 – sky-subtracted object spectra
1 – error spectra
2 – sky spectra
3 – binary FITS table containing the extraction parameters
4 – unused (future: illumination correction).

Within each FITS extension (0-2) the calibrated 1D spectra are stored as "row-stacked" spectra with pixels *rebinned* onto a linear wavelength scale along the X (column) axis. If you extracted 1D spectra from N regions, the row-stacked spectrum file will have N+1 rows, organized as follows:

Row 1: flux calibration response function (log response per Angstrom)
Row 2: wavelength- and flux-calibrated 1D spectrum of extraction region 1
Row 3: calibrated 1D spectrum of extraction region 2
...
Row N+1: calibrated 1D spectrum of extraction region N

An example of a blue channel row-stacked spectral file is shown in Figure 14. As a note, Figure 14 show the extracted row-stacked spectral file in ds9 with zoom parameters of 0.2 and 30 applied to the image. Thus, the display shows 3200 pixels along the horizontal dispersion direction while only ~30 pixels are shown along the vertical direction. These row-stacked spectra can be opened and viewed using IRAF or any other FITS-aware software you wish to use.

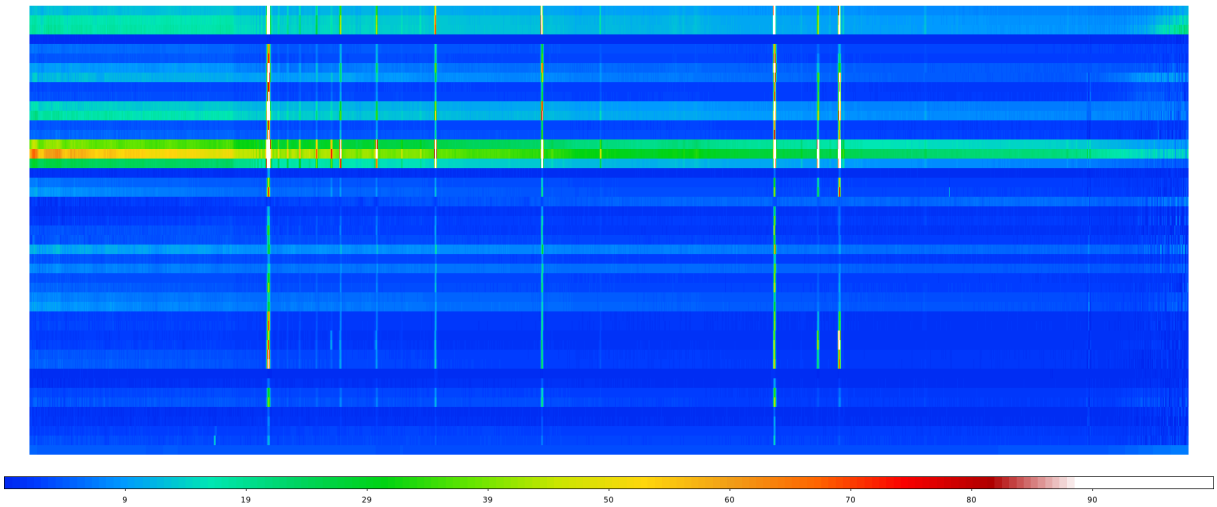

Figure 14: 1D spectra created by `mods_extract1d` are saved as row-stacked spectra. The first (bottom) spectrum in the stack is always the flux calibration response curve applied to the other spectra.

8 Step-By-Step Worked Examples

This section describes a worked example of a long-slit, dual grating mode spectral reduction using the modsIDL package. We have provided a training set of data on the web that you can use containing the raw data below, and a second set of data with the results from running modsIDL on OSU machines so you can compare the output spectra at the end.

This first release just gives a grating long-slit data set. Future versions of this will include examples of MOS grating and prism spectra, once the PIs begin to publish the results and release the data to us.

8.1 Long-slit Grating Mode Observation

8.1.1 Obtaining the Data

You will find five tar files on the modsIDL webpage:

www.astronomy.ohio-state.edu/MODS/Software/modsIDL

These tar files contain the various stages of raw and processed data and calibration files used in this example. To work through this example you will only need `LS_OBS_Raw.tgz`. Other files may be obtained if you wish to compare your results.

8.1.2 Step-By-Step Commands

Make the pixel flats:

```
[Raw]$ modsBias mods1b.20130316.000[3-9].fits
[Raw]$ modsBias mods1b.20130316.001[0-2].fits
[Raw]$ modsBias mods1r.20130316.000[3-7].fits
```

Combine the flats:

```
[Raw]$ modsMedian mods1r.20130316.000[3-7]_ot.fits rFlat_med.fits
[Raw]$ modsMedian mods1b.20130316.000[3-7]_ot.fits bclrFlat_med.fits
[Raw]$ modsMedian mods1b.20130316.000[8-9]_ot.fits
         mods1b.20130316.001[0-2]_ot.fits bug5Flat_med.fits
```

Create the RED normalized pixel flat:

```
[Raw]$ modsFixPix rFlat_med.fits rFlat_fix.fits
[Raw]$ modsPixFlat rFlat_fix.fits pixflat_m1r.fits
```

Create the BLUE normalized pixel flat:

```
[Raw]$ modsAdd bclrFlat_med.fits bug5Flat_med.fits
         bFlat_med.fits
[Raw]$ modsFixPix bFlat_med.fits bFlat_fix.fits
[Raw]$ modsPixFlat bFlat_fix.fits pixflat_m1b.fits
```

Apply pixel Flats to Science and calibration data:

```
[Raw]$ modsProc -b mods1b.20130319.001[2-4].fits pixflat_m1b.fits
[Raw]$ modsProc -b mods1b.20130319.002[1-6].fits pixflat_m1b.fits
[Raw]$ modsProc -b mods1b.20130319.003[3-9].fits pixflat_m1b.fits
[Raw]$ modsProc -b mods1b.20130319.004[0-1].fits pixflat_m1b.fits

[Raw]$ modsProc -b mods1r.20130319.0007.fits pixflat_m1r.fits
```

```
[Raw]$ modsProc -b mods1r.20130319.0008.fits pixflat_m1r.fits
[Raw]$ modsProc -b mods1r.20130319.0009.fits pixflat_m1r.fits
[Raw]$ modsProc -b mods1r.20130319.0015.fits pixflat_m1r.fits
[Raw]$ modsProc -b mods1r.20130319.0016.fits pixflat_m1r.fits
[Raw]$ modsProc -b mods1r.20130319.0017.fits pixflat_m1r.fits
[Raw]$ modsProc -b mods1r.20130319.0018.fits pixflat_m1r.fits
[Raw]$ modsProc -b mods1r.20130319.0019.fits pixflat_m1r.fits
[Raw]$ modsProc -b mods1r.20130319.0020.fits pixflat_m1r.fits
[Raw]$ modsProc -b mods1r.20130319.0030.fits pixflat_m1r.fits
[Raw]$ modsProc -b mods1r.20130319.0031.fits pixflat_m1r.fits
[Raw]$ modsProc -b mods1r.20130319.0032.fits pixflat_m1r.fits
[Raw]$ modsProc -b mods1r.20130319.0033.fits pixflat_m1r.fits
[Raw]$ modsProc -b mods1r.20130319.0034.fits pixflat_m1r.fits
[Raw]$ modsProc -b mods1r.20130319.0035.fits pixflat_m1r.fits
[Raw]$ modsProc -b mods1r.20130319.0038.fits pixflat_m1r.fits
[Raw]$ modsProc -b mods1r.20130319.0039.fits pixflat_m1r.fits
[Raw]$ modsProc -b mods1r.20130319.0040.fits pixflat_m1r.fits
```

Combine Calibration Spectra:

```
[OTF]$ modsMedian mods1b.20130319.002[1-3]_otf.fits blue_ND.fits
[OTF]$ modsMedian mods1b.20130319.002[4-6]_otf.fits
         blue_ug5.fits
[OTF]$ modsMedian mods1r.20130319.001[5-7]_otf.fits red_ND.fits
[OTF]$ modsMedian mods1r.20130319.001[8-9]_otf.fits
         mods1r.20130319.0020_otf.fits red_clr.fits
[OTF]$ modsAdd red_ND.fits red_clr.fits m1r_ill_LS.fits
[OTF]$ modsAdd blue_ND.fits blue_ug5.fits m1b_ill_LS.fits
[OTF]$ modsAdd mods1b.20130319.001[2-4]_otf.fits
         m1b_HgXeAr_LS.fits
[OTF]$ modsAdd mods1r.20130319.000[7-9]_otf.fits
         m1r_NeXeAr_LS.fits
```

Combine science and standard stars:

```
[OTF]$ modsMedian mods1b.20130319.0035_otf.fits
         mods1b.20130319.0033_otf.fits
         mods1b.20130319.0034_otf.fits
         mods1b.20130319.0036_otf.fits
         mods1b.20130319.0037_otf.fits
         mods1b.20130319.0038_otf.fits mods1b.median.fits
[OTF]$ modsMedian mods1r.20130319.0032_otf.fits
         mods1r.20130319.0030_otf.fits
         mods1r.20130319.0031_otf.fits
         mods1r.20130319.0033_otf.fits
         mods1r.20130319.0034_otf.fits
         mods1r.20130319.0035_otf.fits mods1r.median.fits

[OTF]$ modsMedian mods1b.20130319.0040_otf.fits
         mods1b.20130319.0039_otf.fits
         mods1b.20130319.0041_otf.fits mods1b.Feige66.fits
[OTF]$ modsMedian mods1r.20130319.0039_otf.fits
         mods1r.20130319.0038_otf.fits
         mods1r.20130319.0040_otf.fits mods1r.Feige66.fits
```

Sort files for modsIDL processing:

```
[LS_OBS]$ cp Raw/OTF/m1b_ill_LS.fits Proc/.
[LS_OBS]$ cp Raw/OTF/m1r_ill_LS.fits Proc/.
[LS_OBS]$ cp Raw/OTF/m1r_NeXeAr_LS.fits Proc/.
[LS_OBS]$ cp Raw/OTF/m1b_HgXeAr_LS.fits Proc/.
[LS_OBS]$ cp Raw/OTF/mods1b.median.fits Proc/.
[LS_OBS]$ cp Raw/OTF/mods1r.median.fits Proc/.
[LS_OBS]$ cp Raw/OTF/mods1r.fits Proc/.
[LS_OBS]$ cp Raw/OTF/mods1r.2.fits Proc/.
[LS_OBS]$ cp Raw/OTF/mods1r.20130319.003[0-5]_otf.fits Proc/.
[LS_OBS]$ cp Raw/OTF/mods1b.20130319.003[3-8]_otf.fits Proc/.
[LS_OBS]$ cp Raw/OTF/mods1b.Feige66.fits Proc/.
[LS_OBS]$ cp Raw/OTF/mods1r.Feige66.fits Proc/.
```

Create input parameter file:

```
IDL> long_plan,'*fits','Proc/'
```

Run modsIDL:

Perform Initial Reduction

This generates slit maps, wavelength solutions, and creates sky-subtraction ready 2D science and standard star spectra:

```
IDL> mods_reduce,'plan.par',/autotune,/interactive_slits
```

Reduce Standard Star Spectra and Create Response Curves.

We now perform sky subtraction on the 2D standard star spectra (`mods_skyfit2d`), then extract a 1D spectrum and use it to create the flux calibration response function (`mods_fluxstand`). Note that below we have artificially broken the IDL commands into multiple lines for improved readability, but they must be typed all one line.

```
IDL> mods_skyfit2d,
     'Science/std-feige66_m1r.fits.gz',
     'Science/std-feige66_m1b.fits.gz',outname='Feige66',...

Parameters used for mods_skyfit2d:
  BlueCenter=311, RedCenter=305,    lower=50.,  upper = 50.
  BlueCenter=0,   RedCenter=0,      lower=5.,   upper = 200.
  BlueCenter=501, RedCenter=489,    lower=50.,  upper = 5.


IDL> mods_fluxstand,
     'Science/std-feige66_m1r.2dsub.fits',
     'Science/std-feige66_m1b.2dsub.fits',
     std_name='feige66_002',outname='Feige66'
```

Reduce the Science Spectra.

Now perform 2D sky subtractions on the science images and extract 1D spectra. As above, we will artificially break the IDL commands into multiple lines for improved readability.

```
IDL> mods_skyfit2d,
    'Science/sci-median_m1r.fits.gz',
    'Science/sci-median_m1r.fits.gz',outname='median',aperture=[1]
```

```
Parameters used for mods_skyfit2d:
  BlueCenter=316,  RedCenter=307,    lower=100,  upper=100.
  BlueCenter=0,    RedCenter=0,      lower=5.,   upper=170.
  BlueCenter=501,  RedCenter=489,    lower=40.,  upper=5.

IDL> mods_extract1d,
     'Science/sci-median_m1r_median.2dsub.fits',
     'Science/sci-median_m1b_median.2dsub.fits',
     z=0.003156,apertures=[1],outname='xyz'
```

8.2 MOS Grating Mode Observation

Coming in the future.

8.3 MOS Prism Mode Observation

Coming in the future.