FOCAS IFU

Reduction software manual

August 31, 2023

1. Installation

FOCAS IFU reduction software uses Python. It is confirmed to work under both Python 2 and 3.

1.1. Required Linux libraries

Before installing PyRAF, you need to install following Linux libraries.

- Xlib
- Tcl
- Tk
- Tkinter.

For Ubuntu, you can install the following command.

    apt install libx11-dev tcl-dev tk-dev python-tk

1.2. Required python modules

The reduction software has been confirmed to work well under the environments shown in Table 1.

Table 1. Python versions and the module versions.

Python	2.7.12	3.6.7
NumPy	1.14.2	1.13.3
SciPy	1.0.1	0.19.1
matplotlib	2.2.2	2.1.1
Astropy	2.0.5	3.0
Astro-SCRAPPY	1.0.8	1.0.5
stsci.tools	3.4.11	3.4.12
PyRAF	2.1.14	2.1.14
Photutils	0.4	0.4

In Ubuntu 16.04, PyRAF for Python 3 cannot be installed, but it for Python 2 can be install with the following command.

    pip install numpy, scipy, matplotlib, astropy, astroscrappy, stsci.tools==3.4.11,
            photutils

Note: In Ubuntu 16.04, stsci.tools can be installed only for 3.4.11, and not for >=3.4.12. So, you have to install the specific version like above.

In Ubuntu 18.04, all of the modules for both Python 2 and 3 can be installed from the Ubuntu package manager. But their versions tend to be slightly older than those installed with pip.

To improve smoothness of the PyRAF graphics, set the following environment variable.

setenv PYRAFGRAPHICS matplotlib

1.3. IRAF

IRAF is also needed for the software.

In Ubuntu 18.04, it is also available as a part of Ubuntu packages. Note that the IRAF install directory is "/usr/lib/iraf/" in this case, and the environment variable of "iraf" should be set to the directrory.

setenv iraf /usr/lib/ira/

1.4. Installation of FOCAS IFU reduction software

● Download FOCASIFU.tar.gz from the following URL.

✧ https://www2.nao.ac.jp/~shinobuozaki/focasifu/software.html

● Decompress it in a desired directory. The directory tree is shown in Figure 1.
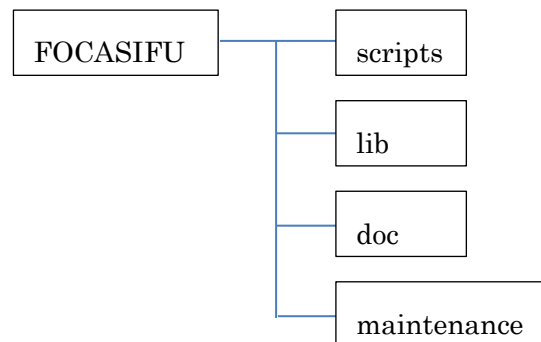
Figure 1. Directory tree

➢ "scripts" directory contains python scripts for the reduction.

➢ "lib" directory contains some data referred by the reduction scripts.

➢ "doc" directory contains this manual.

➢ "maintenance" directory contains some scripts for maintenance of this reduction software. Users usually do not need to use them.

● Add the path of "FOCASIFU/scripts" to the "PATH" environment variable.

2. Data

FOCAS has two CCDs, and each data consists of two FITS files for the two CCDs. These two files have the name with odd and even frame ID numbers (for example, FCSA01234567.fits and FCSA01234568.fits). In this manual, FCSA01234567 or FCSA01234568 is called as the frame ID.
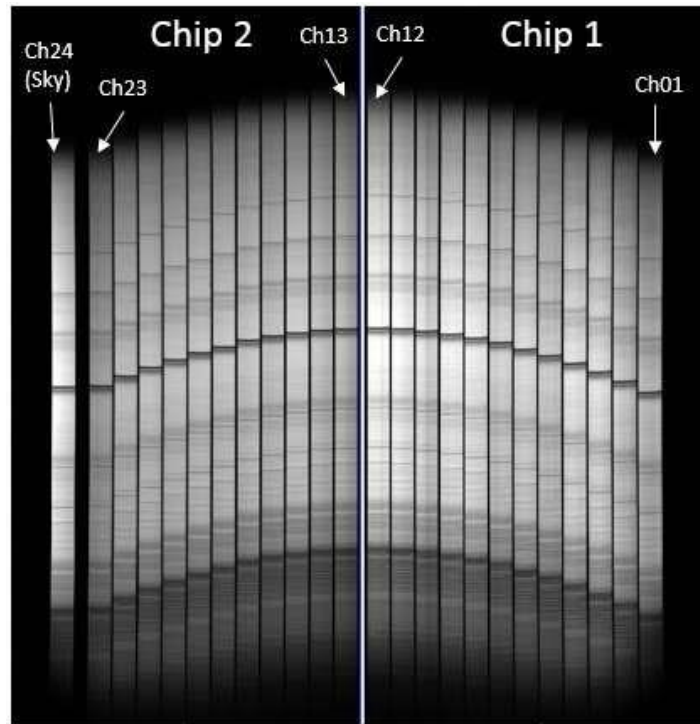
Figure 2. Obtained image with FOACS IFU.

Data cube FITS files created by this software have the following format.

      NAXIS1: slice length direction

      NAXIS2: another spatial direction perpendicular to NAXIS1
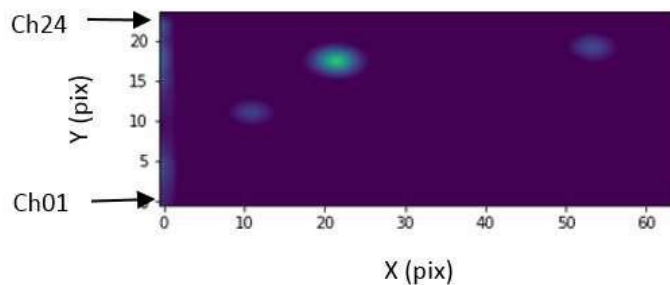
      NAXIS3: wavelength direction

Figure 3. Reconstructed image.

3. Reduction flow

Typical procedure is as follows.

1. First of all, create template bias images using mkbiastemplate.py.
2. Reduce dome flat images using domeflat.py.
3. Reduce CAL flat images for both object and standard star using calflat.py.
4. Reduce comparison images for both object and standard star using comparison.py.
5. Reduce standard star images using standardstar.py.
6. Reduce object images using object.py.
7. Make an error data cube for the object using objecterr.py.

3.1. Examples of the commands

```
# Bias template
mkbiastemplate.py FCSA00200797.fits -d ../20180302/

# Dome flat domeflat.py
FCSA00200723.fits,FCSA00200725.fits,FCSA00200727.fits,FCSA00200729.fits,FCSA0
0200731.fits -d ../20180302/

# CAL flat for object
calflat.py FCSA00201317.fits,FCSA00201319.fits,FCSA00201321.fits -d ../20180302/
# CAL flat for standard star
calflat.py FCSA00200939.fits,FCSA00200941.fits,FCSA00200943.fits -d ../20180302/

# Comparison for object
comparison.py FCSA00201315.fits FCSA00201317 -d ../20180302/
# Comparison for standard star
comparison.py FCSA00200987.fits FCSA00200939 -d ../20180302/

# Standard star
# NOTE: FCSA00200721 is the Frame ID of the comparison image for dome flat. This #
image does not need to be processed by comparison.py.

standardstar.py FCSA00200903.fits FCSA00200723 FCSA00200939 FCSA00200721
```

FCSA00200987 -d ../20180302/

Object
NOTE: FCSA00200721 is the Frame ID of the comparison image for dome flat. This
image does not need to be processed by comparison.py.
object.py   FCSA00201303.fits   FCSA00200723   FCSA00201317
FCSA00201315 FCSA00200721 FCSA00200903 -d ../20180302/

4. Scripts

All scripts have two same options, "-h" and "-o".

    -h : Display a help screen.

    -o : Overwritten output files.

All low-level scripts check whether the output files already exit or not. If exits, the scripts are skipped. If you want to apply a certain script again, you must remove or rename the output file beforehand.

4.1. High-level scripts

4.1.1. domeflat.py

USAGE: domeflat.py [-h] [-o] [-d RAWDATADIR] [-ql] <FITS files>

    ARGUMENT

        FITS files: Comma-separated file names of dome flat images. OPTION

        -d RAWDATADIR: Raw data directory

        -ql: Quick look mode

    OUTPUT

        XXXX.ov.fits : bias-structed and overscan-region-removed image

   XXXX.fcmb.fits : combined flat image

        XXXX.fcmb_low.fits: low spatial frequency image

        XXXX.fcmb_high.fits : high spatial frequency image

        (XXXXX is the frame ID of the input file.)

This script is for dome-flat images and applies the following low-level scripts; bias_overscan.py, flat_combine.py, and divhighlow.py. You have only to input FITS files with odd-number frame IDs. The script deduces FITS file names with even-number frame IDs. See descriptions about each script for more details.

When "-ql" option is attached, this script runs in a quick look mode. This mode skips high and low spatial frequency decomposition for a dome flat image because it takes long time. This is usually used in the observation run.
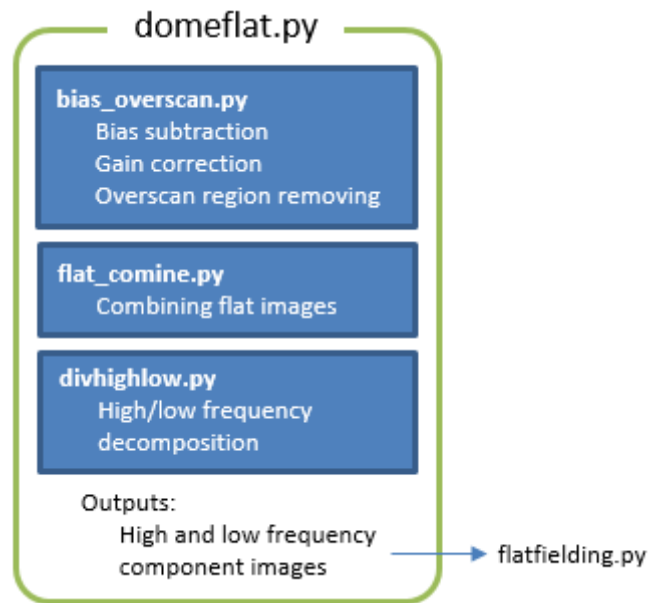
Figure 4. Script structure of domeflat.py.

4.1.2. calflat.py

USAGE: calflat.py [-h] [-o] [-d RAWDATADIR] <FITS files>

ARGUMENT

FITS files: Comma-separated file names of CAL flat images. OPTION

-d RAWDATADIR: Raw data directory

OUTPUT

XXXX.ov.fits : bias-structed and overscan-region-removed image

XXXX.fcmb.fits : combined flat image

idXXXX.fcmb : spectrum gap location database in "database" directory

XXXX.fcmb_gapcoef : coefficient database for spectrum gap position

XXXX.chNNedge.fits : edge-enhanced images in "chimages" directrory.

idXXXX.chNNedge : edge location database in "chimages/database" directory.

idXXXX.ch12edge_org : edge location database for Ch12 in "chimages/database" directory only for VPH650 data.

fcXXXX.chNNedge : spatial coordinate transform function database in "chimages/database" directory

(XXXXX is the frame ID of the input file, and NN is the channel number.)

This script is for CAL-flat images and applies the following low-level scripts; bias_overscan.py, flat_combine.py, identify_gap.py, fit_gap_coordinate.py, mkedgeimage.py, identify_edge.py, correct_ch12_edge.py only for VPH650 data, and

fitcoord_edge.py. You have only to input FITS files with odd-number frame IDs. The script deduces FITS file names with even-number frame IDs. See descriptions about each script for more details.
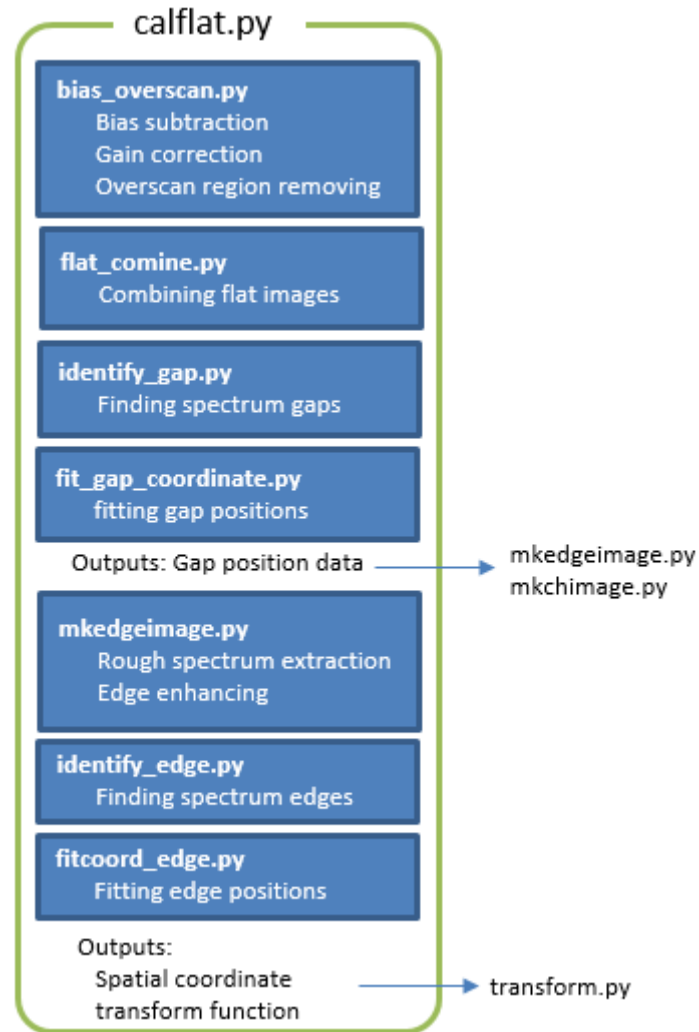
Figure 5. Script structure of calflat.py.

4.1.3. comparison.py

USAGE: comparison.py [-h] [-o] [d RAWDATADIR] [-ql] <FITS files> <CAL flat ID>

    ARGUMENT

        FITS files: Comma-separated file names of comparison images.

        CAL flat ID: Frame ID of the CAL flat image created by calflat.py.

    OPTION

        -d RAWDATADIR: Raw data directory

        -ql: Quick look mode

    OUTPUT

XXXX.ov.fits： bias-structed and overscan-region-removed image

XXXX.chNN.fits：each spectrum image in "chimages" directrory

idXXXX.chNN：emission line location database after bad point elimination in
        "chimages/database" directory

idXXXX.chNN.org：original emission line location database in
        "chimages/database" directory

fcXXXX.chNN：wavelength coordinate transform function in
        "chimages/database" directory

(XXXXX is the frame ID of the input file, and NN is the channel number.)

This script is for comparison images and applies the following low-level scripts; bias_overscan.py, mkchimage.py, identify_dispersion.py, rmidpoints.py, fitcoord_dispersion.py, transform.py, mkcube.py, and get_sky_shift.py. You have only to input FITS files with odd-number frame IDs. The script deduces FITS file names with even-number frame IDs. See descriptions about each script for more details.

When "-ql" option is attached, this script runs in a quick look mode. This mode skips derivation of the wavelength difference between sky and object spectra because it takes long time. This is usually used in the observation run.
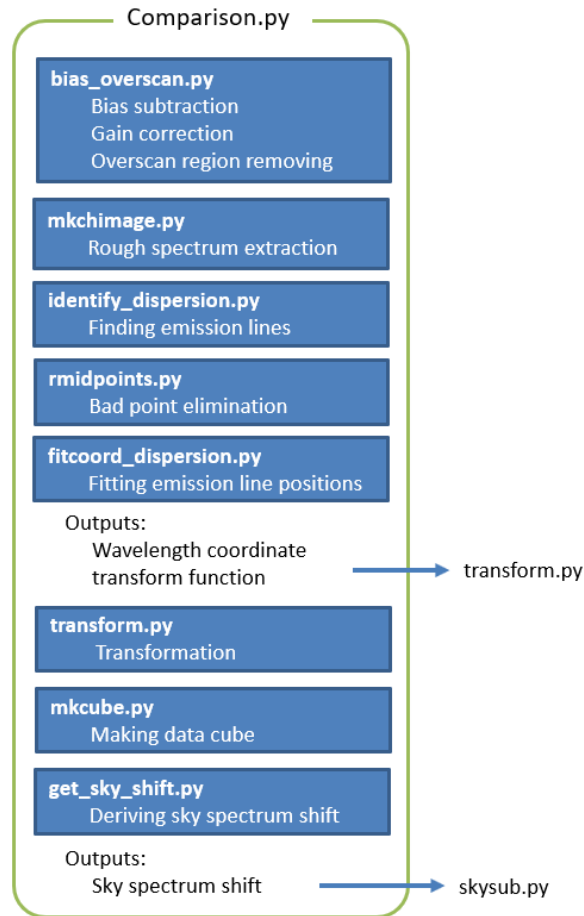
Figure 6. Script structure of comparison.py.

4.1.4. standardstar.py

USAGE: standardstar.py [-h] [-o] [-d RAWDATADIR] <FITS file> <Dome flat ID>

        <CAL flat ID> < Comparison ID for object flat>

        < Comparison ID for dome flat>

  ARGUMENT

    FITS file: Object image

    Dome flat ID: Frame ID of the dome flat image combined by domeflat.py.

    CAL flat ID: Frame ID of the combined CAL flat image created by calflat.py.

    Comparison ID for object flat: Frame ID of the comparison image for object

          processed by comparison.py.

    Comparison ID for dome flat: Frame ID of the comparison image for dome flat.

        This image does not need to be processed by comparison.py.

  OPTION

-d RAWDATADIR: Raw data directory

OUTPUT

XXXX.ov.fits : bias-structed and overscan-region-removed image

XXXX.cr.fits and XXXX.mask.fits : cosmic-ray-removed image and mask image

XXXX.ff.fits : flat-fielded image

XXXX.chNN.fits : each spectrum image in "chimages" directrory

XXXX.chNN.wc.fits : transformed images in "chimages" directrory

XXXX.xyl.fits : data cube

XXXX.ss.fits : sky-subtracted data cube

XXXX.1dspec.fits : 1D spectrum of the standard star

XXXX.std : standard star database

XXXX.sens.fits : sensitivity function image

This script is for an standard star image and applies the following low-level scripts; bias_overscan.py, cosmicrays.py, flatfielding.py, mkchimage.py, transcorm.py, mkcube.py, skysub.py, std1dspec.py and standard_sens.py. You have only to input a FITS file with an odd-number frame ID. The script deduces a FITS file name with an even-number frame ID. See descriptions about each script for more details.

Figure 7. Script structure of standard.py.

4.1.5. object.py

USAGE: object.py [-h] [-o] [d RAWDATADIR] [-domecompid DOMECOMPID]

　　　　　[-stdstarid STDSTARID] [-ql] <FITS file> <Dome flat ID>

　　　　　<CAL flat ID> < Comparison ID for object flat>

　　ARGUMENT

　　　　FITS file: Object image

　　　　Dome flat ID: Frame ID of the dome flat image combined by domeflat.py.

　　　　CAL flat ID: Frame ID of the combined CAL flat image created by calflat.py.

　　　　Comparison ID for object flat: Frame ID of the comparison image for object

processed by comparison.py.

OPTION

-domecompid DOMECOMPID: Frame ID of the comparison image for dome flat.

This image does not need to be processed by comparison.py.

-stdstarid STDSTARID： Frame ID of the standard star image.

-d RAWDATADIR： Raw data directory

-ql： Quick look mode

OUTPUT

XXXX.ov.fits : bias-structed and overscan-region-removed image

XXXX.cr.fits and XXXX.mask.fits : cosmic-ray-removed image and mask image

XXXX.ff.fits : flat-fielded image

XXXX.chNN.fits : each spectrum image in "chimages" directrory

XXXX.chNN.wc.fits : transformed images in "chimages" directrory

XXXX.xyl.fits : data cube

XXXX.ss.fits : sky-subtracted data cube

XXXX.fc.fits : flux-calibrated data cube

This script is for an object image and applies the following low-level scripts; bias_overscan.py, cosmicrays.py, flatfielding.py, mkchimage.py, transform.py, fluxcalib.py, mkcube.py, skysub.py and skysub_old.py. You have only to input a FITS file with an odd-number frame ID. The script deduces a FITS file name with an even-number frame ID. See descriptions about each script for more details.

In a sky subtraction process, this uses skysub_old.py for old data using the old sky slit.

When "-ql" option is attached, this script runs in a quick look mode. This mode does not use high and low spatial frequency flat images (see also domeflat.py). This is usually used in the observation run.

Figure 8. Script structure of object.py.

4.1.6. objecterr.py

USAGA: objecterr.py [-h] [-domecompid DOMECOMPID] [-stdstarid STDSTARID]

         [-d RAWDATADIR] [-o] [-ql] <Object FITS file> <Bias FITS file>

        <Dome flat ID> <CAL flat ID> < Comparison ID for object flat>

   ARGUMENT

      Object FITS file: Object image

      Bias FITS file: Bias image

      Dome flat ID: Frame ID of the dome flat image combined by domeflat.py.

      CAL flat ID: Frame ID of the combined CAL flat image created by calflat.py.

      Comparison ID for object flat: Frame ID of the comparison image for object

             processed by comparison.py.

   OPTION

-domecompid DOMECOMPID: Frame ID of the comparison image for dome flat.
This image does not need to be processed by comparison.py.

-stdstarid STDSTARID： Frame ID of the standard star image.

-d RAWDATADIR： Raw data directory

-ql： Quick look mode

This script is for creating an error data cube and applies the following low-level scripts: bias_overscan.py, cosmicrays.py, errorestimate.py, flatfielding.py, mkchimage.py, transform.py, fluxcalib.py, and mkcube.py. You have only to input a FITS file with an odd-number frame ID. The script deduces a FITS file name with an even-number frame ID. See descriptions about each script for more details. You should use same arguments and options as those for object.py.

When "-ql" option is attached, this script runs in a quick look mode. This mode does not use high and low spatial frequency flat images (see also domeflat.py).
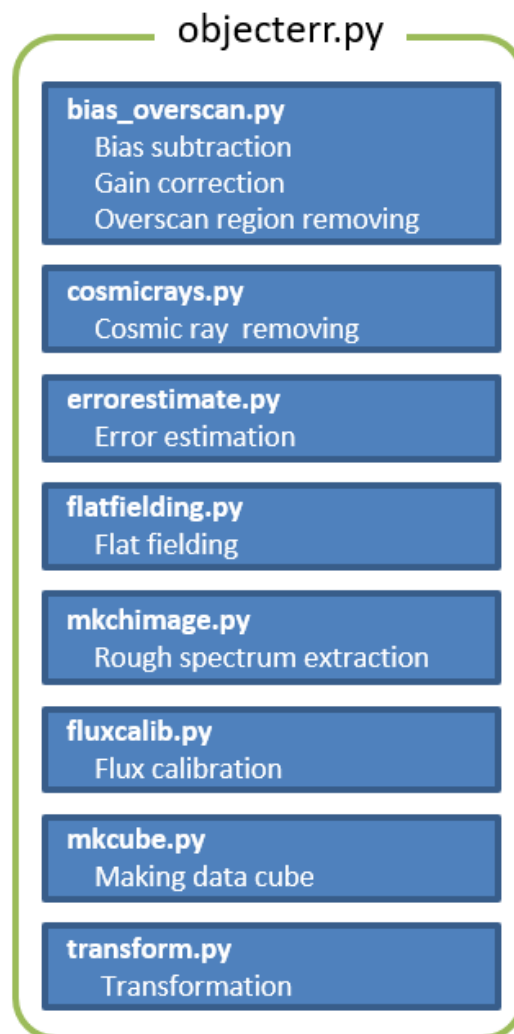
Figure 9. Script structure of objecterr.py.

4.2. Low-level script

4.2.1. bias_overscan.py
Usage: bias_overscan.py [-h] [-o] [-d RAWDATADIR] [-bo] <FITS file>

ARGUMENT

FITS file: Input FITS file with an odd-number frame ID OPTION

-d RAWDATADIR: Raw data directory

-bo: Upper overscan region of the input FITS file is used for the bias subtraction instead of the bias template files.

OUTPUT

XXXXX.ov.fits（XXXXX is the frame ID of the input file.）

This script

- subtracts the bias,
- removes the over scan regions,
- converts the unit from ADUs to electrons,
- corrects the bad pixels, and
- merges the two images for two CCDs into one image.

You have only to input one FITS file with an odd-number frame ID. The script deduces another FITS file name with an even-number frame ID. Unless the option -bo is not used, this needs the bias template files created by mkbiastemplate.py. If you can find the systematic noise after bias subtraction reported here, please use this option.

4.2.2. correct_ch12_edge.py
USAGE: correct_ch12_edge.py [-h] [-o] <CAL flat ID>

ARGUMENT

CAL flat ID: Frame ID of the CAL flat image combined by flat_combine.py.

OUTPUT

idXXXX.ch12edge and idXXXX.ch12edge_org in "chimages/database" directory

(XXXXX is the frame ID of the input file, and NN is the channel number.)

When VPH650 is used, the left edge of the Ch12 spectrum is in the gap between the two CCDs. This script corrects the left edge position data detected in identify_edge.py. The left edge position is estimated from the spectrum width interpolated from other channels and the detected Ch12 right edge. This renames the old id-file to "idXXXX.ch12edge_org" and creates a new id-file, "idXXXX.ch12edge." This is not needed for the other grisms.

4.2.3. cosmicrays.py

USAGE: cosmicrays.py [-h] [-o] [-sigclip SIGCLIP] [-sigfrac SIGFRAC]

[-niter NITER] <FITS file>

ARGUMENT

FITS file: Input FITS file name

OPTION

-sigclip SIGCLIP: Laplacian-to-noise limit for cosmic ray detection. (default: 5.0)

-sigfrac SIGFRAC: Fractional detection limit for neighboring pixels. (default: 0.4)

-niter NITER: Number of iterations of the LA Cosmic algorithm to perform. (default: 4)

OUTPUT

XXXX.cr.fits and XXXX.mask.fits

(XXXXX is the frame ID of the input file)

This script removes cosmic rays and creates a mask image using the Python module, Astro-SCRAPPY, based on the L.A.Cosmic algorithm (Pieter G. van Dokkum, 2001, PASP, 113, 1420). SIGCLIP, SIGFRAC and NITER are the options for Astro-SCRAPPY. For more details about the options, see the Astro-SCRAPPY web page.

4.2.4. divhighlow.py

Usage: divhighlow.py [-h] [-o] <FITS file>

ARGUMENT

FITS file: Dome flat files combined using flat_combine.py.

OUTPUT

XXXX.fcmb_low.fits and XXXX.fcmb_high.fits

(XXXXX is the frame ID of the input file.)

The Ch10 slice mirror has chips at the edge. These causes dark lanes in the Ch10 spectrum. The dark lanes move by some pixels due to flexure. To correct the dark lanes

as well as pixel-to-pixel efficiency variation in a flat fielding procedure, a flat image must be decomposed into two components; One is a component moving due to the flexure, and the other is a stationary component. The former will be shifted before it is used in a flat fielding process.

This script decomposes a dome flat image into the two components. This takes time. For reducing the time, multi-processing using all CPUs is used. For example, the time is about 8 minutes for one 2x1 binning data with Intel Xeon E5-1650 v3 processor having 6 cores.

4.2.5. errorestimate.py

USAGE: errestimate.py [-h] [-d RAWDATADIR] [-o] <Object FITS file> <Bias FITS file>

    ARGUMENT

        Object FITS file: Object image after cosmic ray removing.

        Bias FITS file: Bias image

    OUTPUT

        XXXX.er.fits: Error image (XXXXX is the frame ID of the object FITS file.)

This script creates an error image. Photon noise and readout noise are estimated from the object and bias images, respectively.

4.2.6. fit_gap_coordinate.py

USAGE: fit_gap_coordinate.py [-h] [-o] <FITS file>

    ARGUMENT

        FITS file: CAL flat file combined using flat_combine.py.

    OUTPUT

        XXXX.fcmb_gapcoef (XXXXX is the frame ID of the input file.)

For spectrum extraction, gap positions identified using identify_gap.py are fitted with 2nd order Chebyshev polynomial function. In the fitting, 3-sigma clipping is applied. This result is used in mkchimage.py and mkedgeimage.py. The identified positions and the best fit functions are shown in the left panel of Figure 10. The fitting residuals are shown in the right panel of Figure 10. Deviation within 1 pixel is acceptable. The clipped data points are shown by cross marks. Color code shows difference of the gaps (Some cross marks have different color from the assigned color for the gaps. This might be due to a bug in plotting.). After checking the results, close the window to quit this script.
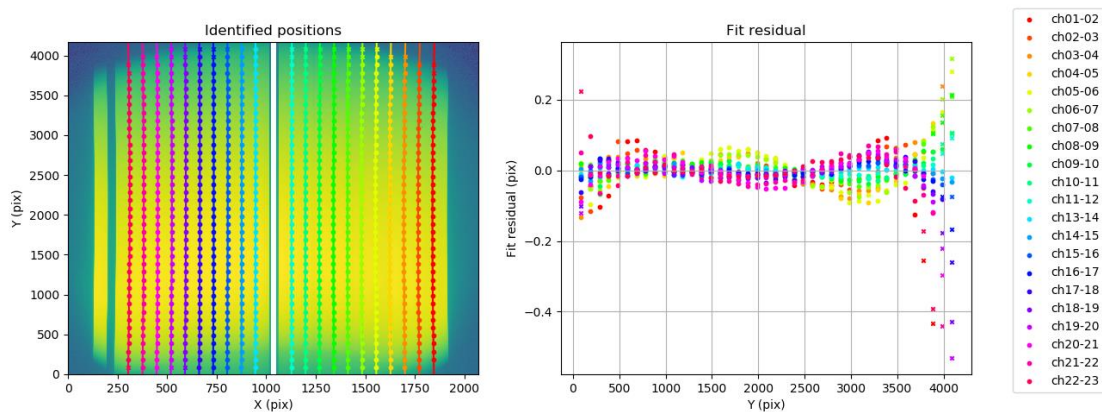
Figure 10.

4.2.7. fitcoord_dispersion.py

USAGE: fitcoord_dispersion.py [-h] [-o] <Frame IDs>

ARGUMENT

Frame IDs: Comma-separated frame IDs of the comparison FITS files.

OUTPUT

fcXXXX.chNN in "chimages/database" directory

(XXXXX is the frame ID of the input file, and NN is the channel number.)

This script creates wavelength coordinate transform functions using the IRAF FITCOORDS task. In the default plot, horizontal axis is Y (dispersion direction). Change it to X (spatial direction) pressing 'x', 'x', and then 'r' in the graphical window. Some data points near the edges might largely deviate (Figure 11). Because those are near the pseudo slit edges, eliminate them pressing 'd' near one of them and then 'p' or 'x'. To fit again, press 'f' key (Figure 12). Go back to the default plow pressing 'x', 'y', and then 'r'. If some data points still significantly deviate, they should be also eliminated. To quit, press 'q' key. See the IRAF manual for more details.
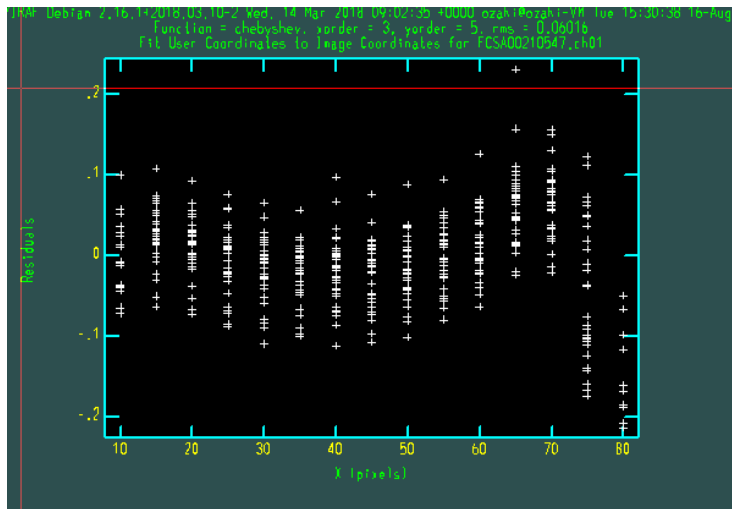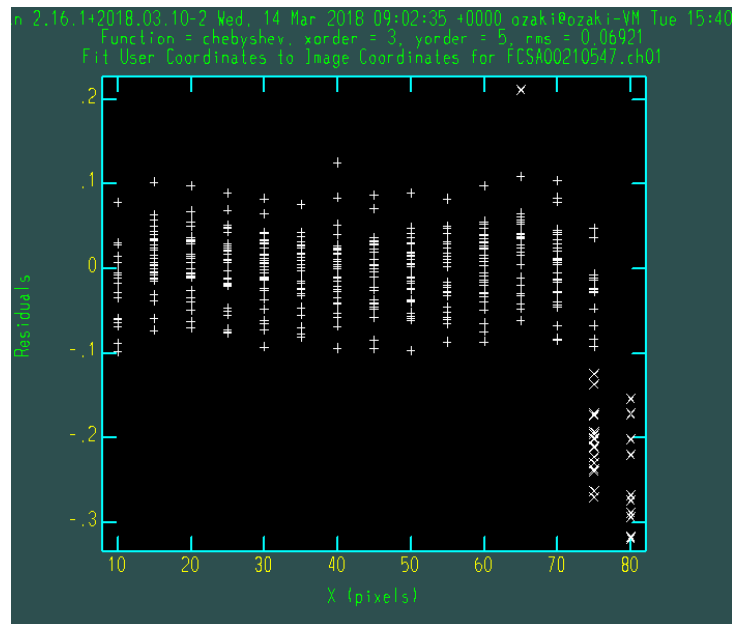
Figure 11.

Figure 12.

4.2.8. fitcoord_edge.py

USAGE: fitcoord_edge.py [-h] [-o] <FITS file>

    ARGUMENT

        FITS file: CAL flat files combined using flat_combine.py.

    OUTPUT

        fcXXXX.chNNedge in "chimages/database" directory

        (XXXXX is the frame ID of the input file, and NN is the channel number.)

This script derives spatial coordinate transform functions using the IRAF FITCOORDS task. In the default plot, horizontal axis is X (spatial direction) (Figure 13). Change it to Y (dispersion direction) pressing 'x', 'y', and then 'r' in the graphical window. Some data points near the Y edges might largely deviate (Figure 14). In that case, eliminate the outermost points even if they do not seem to deviate. pressing 'd' near one of them and then 'y'. To fit again, press 'f' key. If there are still largely deviating points, repeat the above elimination procedure (Figure 15). To quit, press 'q' key. See the IRAF manual for more details.
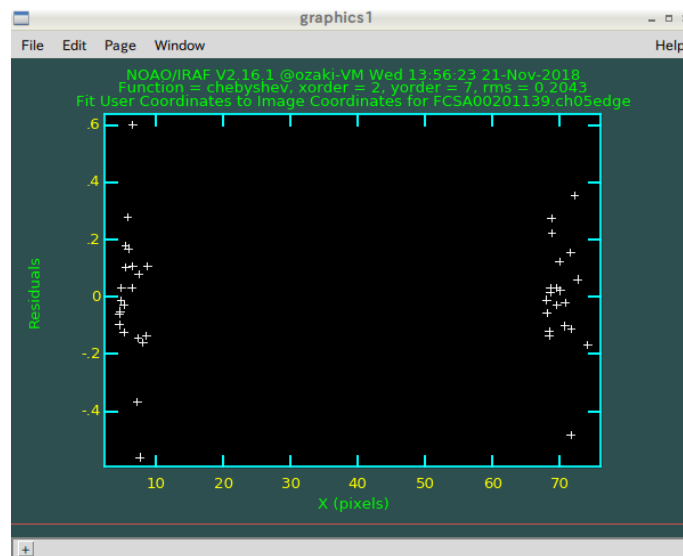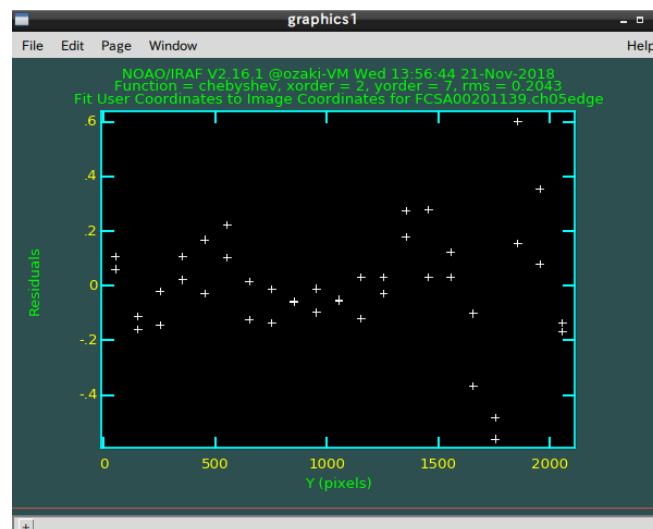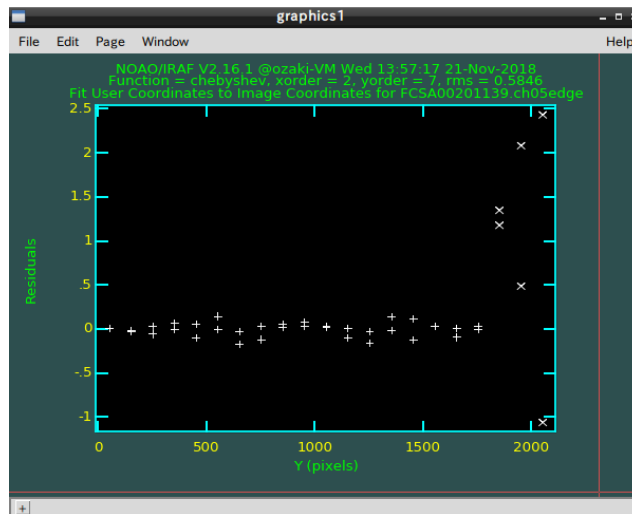

Figure 13.

Figure 14.

Figure 15.

4.2.9. flat_combine.py

Usage: flat_combine.py [-h] [-o] <FITS files>

ARGUMENT

FITS files: Comma-separated flat FITS files created by bias_overscan.py.

OUTPUT

XXXX.fcmb.fits (XXXX is the frame ID of the first input file.)

This script simply combines either dome or CAL flat images using the IRAS task IMCOMBINE with the combine type of 'median'. See the IRAF manual for the more details.

4.2.10. flatfielding.py

USAGE: flatfielding.py [-h] [-o] [-e] [-comp COMP] [-domecomp DOMECOMP]

<FITS file> <Dome flat ID> <CAL flat ID>

ARGUMENT

FITS file: Input FITS file name

Dome flat ID: Frame ID of the dome flat image combined by flat_combine.py.

CAL flat ID: Frame ID of the CAL flat image combined by flat_combine.py.

OPTION

-comp COMP: Fram ID of the comparison image for object.

-domecomp DOMECOMP: Frame ID of the comparison image for dome flat.

-ql: Quick look mode

-e: Put this option for error estimation.

OUTPUT

XXXX.ff.fits　　　(XXXXX is the frame ID of the input file.)

This script performs flat fielding to the input image. An image shift along the X direction is automatically derived from the cross-correlation between the dome flat and the CAL flat (The CAL flat was always taken in the telescope position same as the object). A Y-direction shift is also automatically calculated from the comparison images for the object and the dome flat. The X and Y shifts are shown in plot windows (Figure 16 and Figure 17). After checking, close them.

When "-ql" option is attached, this script runs in a quick look mode. This mode does not use high and low spatial frequency flat images (see also domeflat.py).
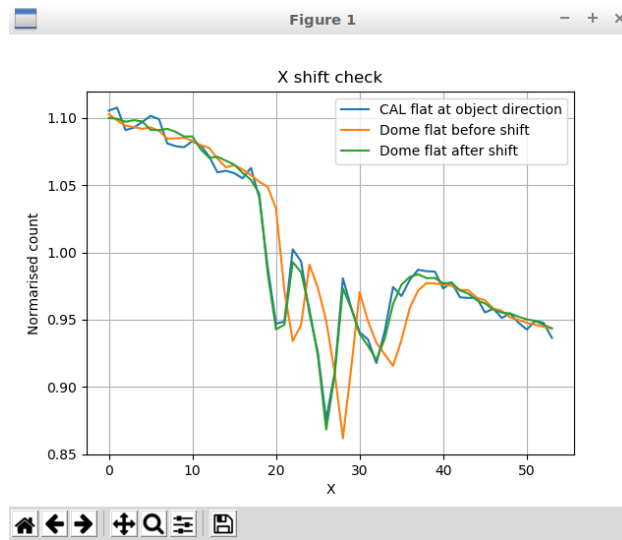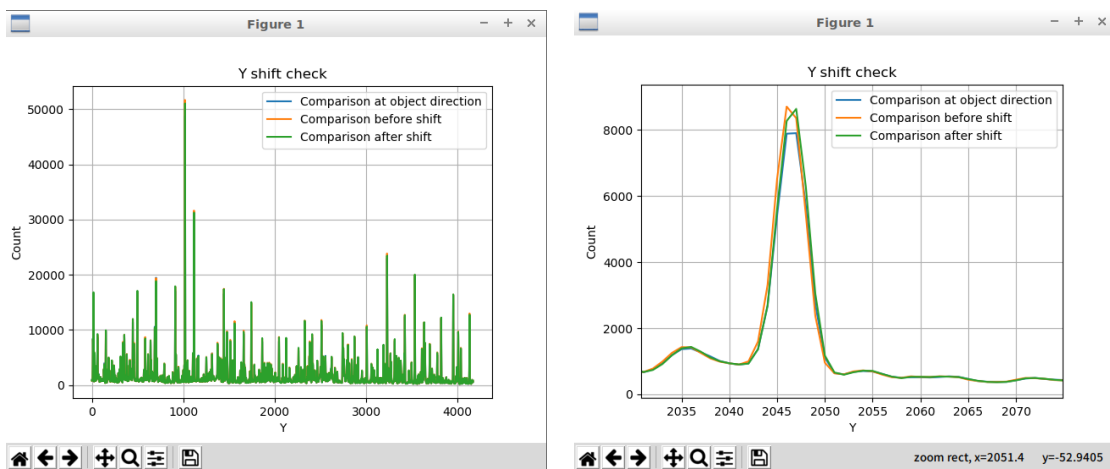

Figure 16. Plot for X shift check.

Figure 17. Plot for Y shift check. Right panel shows a magnified view.

4.2.11. fluxcalib.py

USAGE: fluxcalib.py [-h] [-o] <Input FITS file> <Sensitivity file>

    ARGUMENT

        Input FITS file: Input FITS file name to be calibrated.

        Sensitivity file: Sensitivity function image created by standard_sens.py.

    OUTPUT

        XXXX.fc.fits   (XXXXX is the frame ID)

This script applies flux calibration to the data cube using the sensitivity function derived by standard_sens.py.

4.2.12. get_sky_shift.py

USAGE: get_sky_shift.py [-h] [-o] [-ql] <Comparison ID>

    ARGUMENT

        Comparison ID: Frame ID of the comparison data cube.

    OPTIONS

        -ql: Quick look mode

    OUTPUT

        XXXX.sky_shift.dat   (XXXXX is the frame ID)

This script derives wavelength differences between a sky spectrum and object spectra of each spaxel using the comparison data cube. Results can be plotted using show_skyshift.py.

When the "-ql" option is attached, this script runs in a quick look mode and skips derivation of difference between sky and object spectra.

4.2.13. identify_dispersion.py

USAGE: identify_dispersion.py [-h] [-o] <Frame IDs>

    ARGUMENT

        Frame IDs: Comma-separated frame IDs of the comparison FITS files.

    OUTPUT

        idXXXX.chNN in "chimages/database" directory

        (XXXXX is the frame ID of the input file, and NN is the channel number.)

This script identifies emission line locations using the IRAF IDENTIFY and REIDENTIFY tasks. An object frame with sky emission lines can be used. When the number of frames except an object frame is 2, then the frame for bright lines is followed by the one for faint lines in the argument. An object frame must be at last in the argument. Emission lines are automatically identified. If some lines are not correctly identified, manually identify them with comparing the reference figure shown in the separated window.

comp_emis_position.py would be helpful for distinguishing what data points should be removed.

4.2.14. identify_edge.py

USAGE: identify_edge.py [-h] [-o] <FITS file>

    ARGUMENT

        FITS file: CAL flat files combined using flat_combine.py.

    OUTPUT

        idXXXX.chNNedge in "chimages/database" directory.

        (XXXXX is the frame ID of the input file, and NN is the channel number.)

This script identifies spectrum-edge locations using the IRAF IDENTIFY and REIDENTIFY tasks for each channel. In the graphical window, press m-key at the two edge-peak locations. In the plot, there are some peaks. You should select the inner peak pair like Figure 18. After identifying two peaks, quite the IDENTIFY task with q-key. The REIDENTIFY task starts right after that. If the data is for VPH650, then the left-edge-position data of the Ch12 spectrum is automatically corrected (see correct_ch12_edge.py).
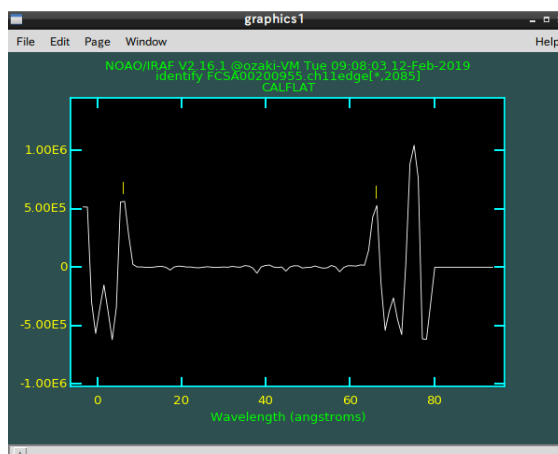
Figure 18.

4.2.15. identify_gap.py

USAGE: identify_gap.py [-h] [-o] <FITS file>

    ARGUMENT

        FITS file: CAL flat file combined using flat_combine.py.

    OUTPUT

        id-files in "database" directory

This script identifies spectrum gap locations using the IRAF tasks, IDENTIFY and REIDENTIFY. Figure 19 shows the PyRAF graphic window for identifying gaps. In the window, press l-key to find the gap locations. The IRAF colon command ":label both" shows the gap names in the window. The number of gaps to be found is 21. If some gaps are not found, manually identify the positions using m-key. See the IRAF manual for the details. The gap positions of Ch12-13 and Ch23-24 are not used in the following procedure, so this script does not find these two gaps. After finding all gaps, quit the IRAF IDENTIFY task by pressing q-key. The IRAF REIDENTIFY task automatically starts.
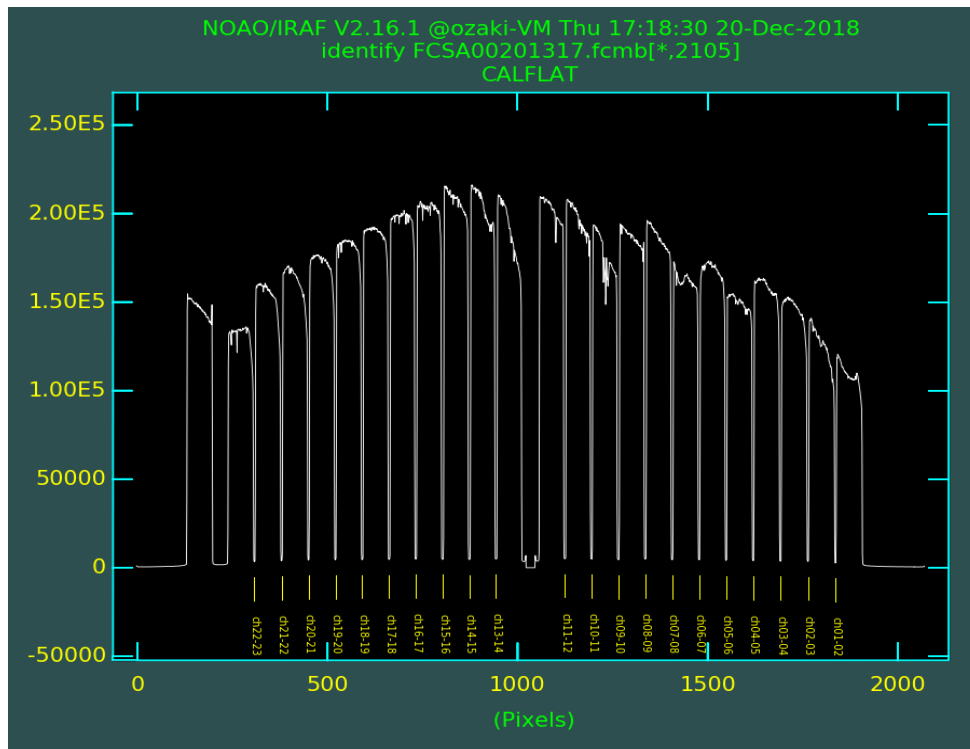
Figure 19.

4.2.16. mkbiastemplate.py

USAGE: mkbiastemplate.py [-h] [-o] [-d RAWDATADIR] <FITS file>

    ARGUMENT

        FITS file: Input bias file name with an odd-number frame ID OPTION

        -d RAWDATADIR: Raw data directory

    OUTPUT

        bias_template1.fits and bias_template2.fits

This script creates 1-D bias template integrating the bias image along Y direction. In the integration, 4-sigma clipping is applied. You have only to input one FITS file with an odd-number frame ID. The script deduces another FITS file name with an even-number frame ID. This script must be run before bias subtraction.

4.2.17. mkchimage.py

USAGE: mkchimage.py [-h] [-o] [-e] <FITS file> <CAL flat ID>

    ARGUMENT

        FITS file: FITS file created by bias_overscan.py.

        CAL flat ID: Frame ID of the relevant CAL flat image

OPTION

   -e: Put this option for error estimation

OUTPUT

   XXXX.chNN.fits in "chimages" directrory for a science data

   XXXX.chNN.err.fits in "chimages" directrory for an error data

   (XXXX is the frame ID of the input FITS file, and NN is the channel number.)

This script extracts each channel spectrum, and stores them in separated FITS files in "chimages" directory. When estimating error, attach the -e option.

4.2.18. mkcube.py

USAGE: mkcube.py [-h] [-o] <Frame ID>

   ARGUMENT

      Frame ID: Frame ID of the object frame.

   OUTPUT

      XXXX.xyl.fits   (XXXXX is the frame ID)

This script creates a data cube from 24 extracted images, and added the WCS headers.

4.2.19. mkedgeimage.py

USAGE: mkedgeimage.py [-h] [-o] <FITS fits>

   ARGUMENT

      FITS file: CAL flat files combined using flat_combine.py.

   OUTPUT

      XXXX.chNNedge.fits in "chimages" directrory.

      (XXXXX is the frame ID of the input file, and NN is the channel number.)

This script extracts each channel spectrum and creates edge-strengthen images. Since Ver. 20210210, neighboring spectrum areas are not padded with zero because the 0-padding causes slight uncertainty (< 1 spatial pixel ~ 0.2 arcsec by default) in defining spectrum edges.

4.2.20. rmidpoints.py

USAGE: rmidpoints.py [-h] [-fac FAC] [-o] <Frame ID>

   ARGUMENT

      Frame ID: Comma-separated Frame IDs

-fac FAC: Threshold factor

OUTPUT

idXXXX.chNN.org

idXXXX.chNN

This script eliminates bad points in the database file, idXXXX.chNN, created by identify_dispersion.py. The original id-file is renamed to idXXXX.chNN.org. When intensity of a data point is lower by the threshold factor with respect to a neighboring data point, the data point is eliminated. After the elimination, this shows the following figure for checking the result. Close this plot window to proceed the script.
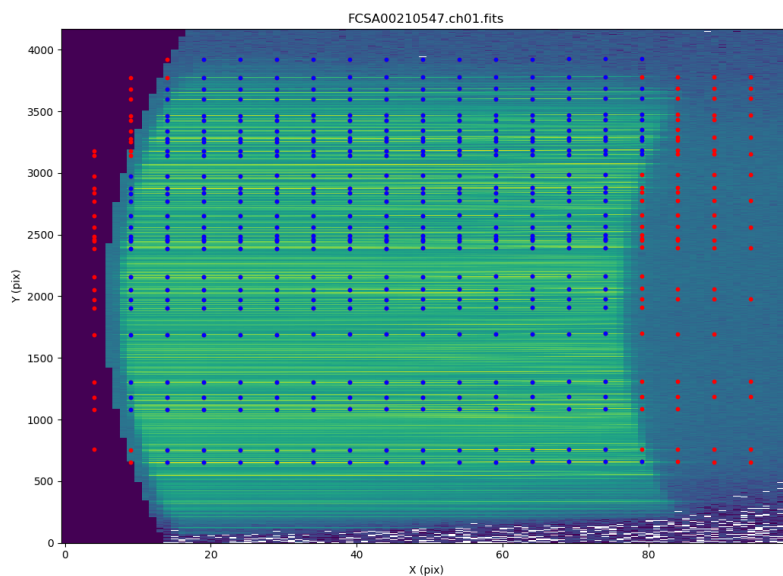
Figure 20. Figure shown by rmidpoints.py. Data points identified by identify_dispersion.py are overplotted on the corresponding comparison image with log scale. Red points show eliminated ones.

4.2.21. skysub.py

USAGE: skysub.py [-h] [-o] [-scale SCALE] <FITS file> <Comparison ID>

ARGUMENT

FITS file: Input FITS file name of the data cube created by mkcube.py.

Comparison ID: Frame ID of the comparison image. OPTIONS

-scale SCALE: Scale factor applied for the sky spectrum. (default: 1.0)

OUTPUT

XXXX.ss.fits   (XXXXX is the frame ID)

This script is applied for data since August 2021 because those data were obtained with the new sky slit. This script subtracts the sky spectrum from the data cube. The sky spectrum is created by integrating the Ch24 (sky) spectrum. The sky slit has graded

width. The width is narrower for larger X coordinate. The sky spectrum is scaled by SCALE. The scaled sky spectrum is shifted for each object spaxel along the dispersion direction by the value derived with get_sky_shift.py.

4.2.22. skysub_old.py
USAGE: skysub_old.py [-h] [-o] [-x1 X1] [-x2 X2] [-scale SCALE]

                                                  \<FITS file\> \<Comparison ID\>

    ARGUMENT

        FITS file: Input FITS file name of the data cube created by mkcube.py.

        Comparison ID: Frame ID of the comparison image. OPTIONS

        -x1 X1: Start pixel for integrating the sky spectrum. (default: 53)

        -x2 X2: End pixel for integrating the sky spectrum. (default: 62)

        -scale SCALE: Scale factor applied for the sky spectrum. (default: 1.0)

    OUTPUT

        XXXX.ss.fits   (XXXXX is the frame ID)

This script is applied for the old data before July 2021 because those data were obtained with the old sky slit. This script subtracts the sky spectrum from the data cube. The sky spectrum is created by integrating the Ch24 (sky) spectrum from X1 to X2. The sky slit has graded width. The width is narrower for larger X coordinate. The sky spectrum is scaled by SCALE. The scaled sky spectrum is shifted for each object spaxel along the dispersion direction by the value derived with get_sky_shift.py.

4.2.23. standard_sens.py
USAGE: standard_sens.py [-h] [-o] \<FITS file\>

    ARGUMENT

                FITS file: FITS file name of the 1D standard star spectrum made by

            std1dspec.py.

    OUTPUT

        XXXX.std and XXXX.sens.fits   (XXXXX is the frame ID)

This script derives the sensitivity function using the IRAF STANDARD and SENSFUNC tasks. STANDARD selects the bands used for deriving a sensitivity function (Figure 21). 'd' key is for deleting the band. SENSFUNC derives the sensitivity function (Figure 22). Also, 'd' key is for deleting the points from the fitting. See the IRAF manual for more details.

Around 9000 – 9500 Å, there is a bump in the sensitivity function. This originates from the absorption feature in the dome flat. To fit this feature, high order function is required in the fitting. In the case of Figure 22, the 50-order spline3 function is used. For such high order fitting, CALSPEC is (hattori@naoj.org) recommended because of the dense spectral sampling. Some data converted to IRAF database format can be found in the FOCAS web page. Please contact to the Subaru telescope for more detail.
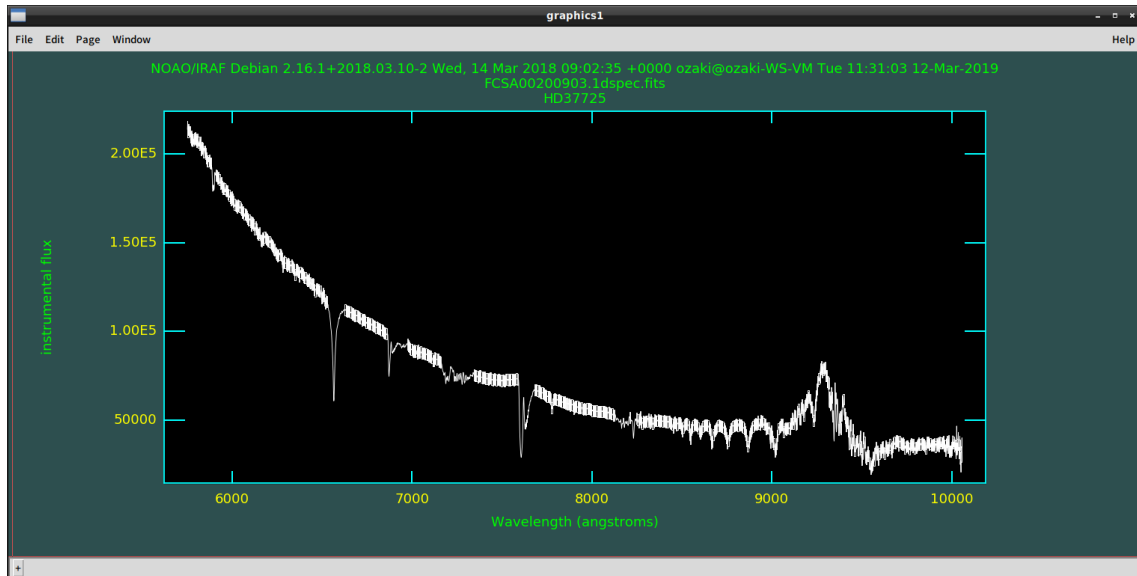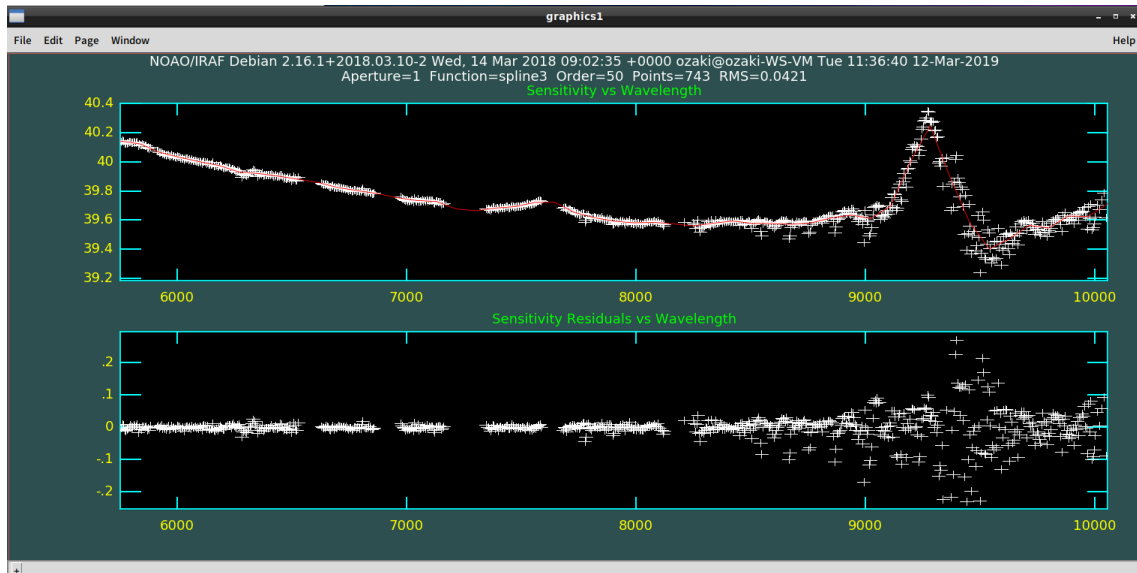

Figure 21. Plot window of the IRAF STANDARD task.

Figure 22. Plot window of the IRAF SENSFUNC task.

4.2.24. std1dspec.py

USAGE: std1dspec.py [-h] [-o] [-startz STARTZ] [-nsigma NSIGMA] <FITS file>

 ARGUMENT

   FITS file: FITS file name of the data cube of the standard star.

 OPTIONS

   -startz STARTZ: Start Z pixel for aperture photometry. (default: 2000)

   -nsigma NSIGMA: Number of sigma for the aperture size. (default: 5)

 OUTPUT

   XXXX.1dspec.fits   (XXXXX is the frame ID)

This script makes a 1D spectrum of the standard star. The procedure is as follows.

1) After an image at Z = STARTZ is shown (Figure 23 left), click around the star center.

2) Standard deviations along X and Y directions are obtained applying 2D Gaussian fitting.

3) A photometric aperture radius is set to NSIGMA times larger than the standard deviations. And the aperture center is set at the 2D Gaussian center.
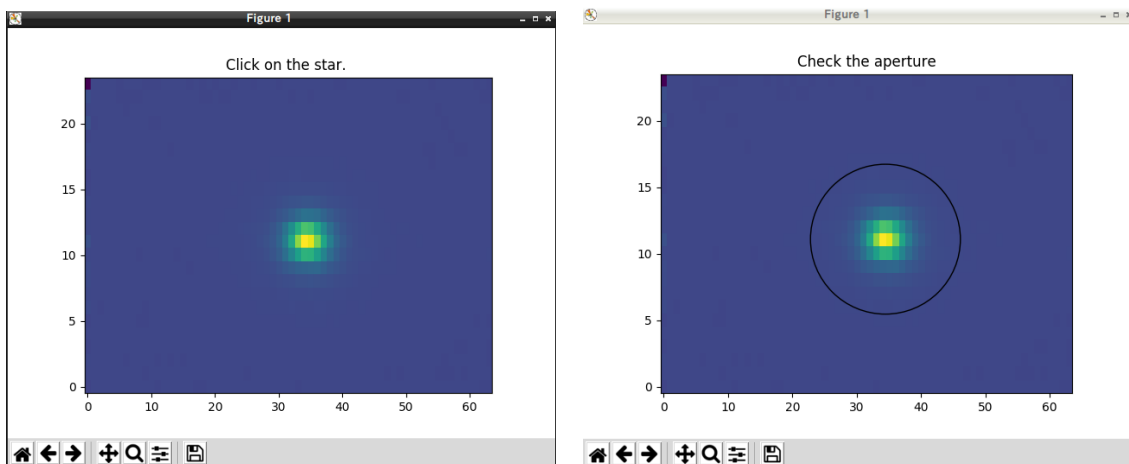

Figure 23.

4) The aperture is overlaid on the image (Figure 23 right). Check the aperture, and close the graphic window.

5) The aperture center is recalculated at every wavelength pixel, and count values are integrated within the aperture.

6) The 1D standard star spectrum is shown (Figure 24 left).

7) Specify the required wavelength range by pushing any key except 'q' at both ends of the range (Figure 24 right). Including rapidly changing region causes difficulty in deriving sensitivity function.
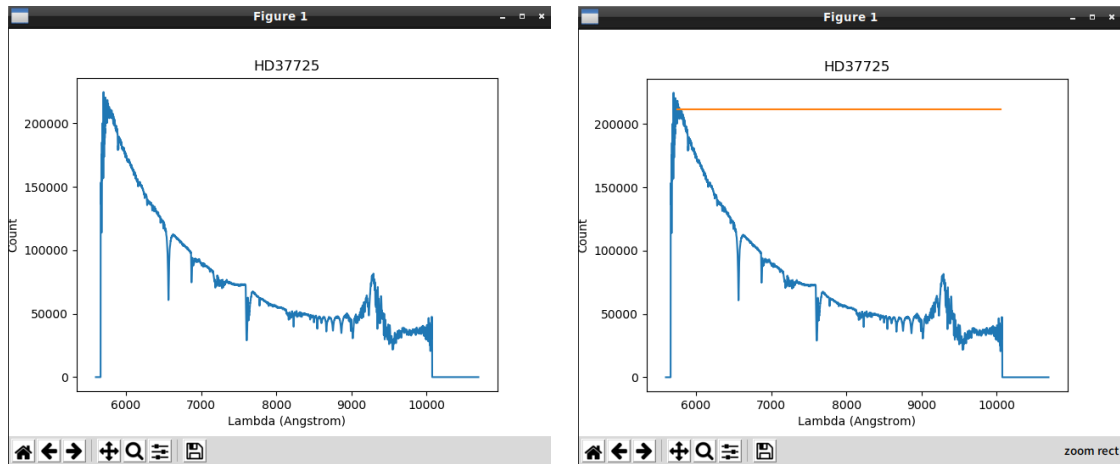
8) Close the plot window.

Figure 24.

4.2.25. transform.py
USAGE: transform.py [-h] [-o] <Frame ID> <Comparison ID> <CAL flat ID>

    ARGUMENT

        Frame ID: Frame ID of the object frame.

        Comparison ID: Frame ID of the comparison frame.

        CAL flat ID: Frame ID of the CAL flat image combined by flat_combine.py.

    OUTPUT

        XXXX.chNN.wc.fits

        (XXXXX is the object frame ID, and NN is the channel number.)

This script transforms each channel spectrum to user coordinate using the wavelength and spatial coordinate transform functions derived from fitcoord_dispersion.py and fitcoord_edge.py, respectively. This uses the IRAF TRANSFORM task internally. NOTE: Since Ver.20201003, blank pixels are not padded with NaN in this script. The function is moved to the separate script, zero_padding.py.

5. Tools

5.1. comp_emis_position.py
usage: comp_emis_position.py [-h] ifnames

This is the script for checking the identified positions of the comparison lines. This is helpful to distinguish what points should be removed in fitcoord_dispersion.py.

positional arguments:

  ifnames   Comma-separated input FITS files
            (e.g. FCSA00210597.ch12.fits,FCSA00210599.ch12.fits)

optional arguments:
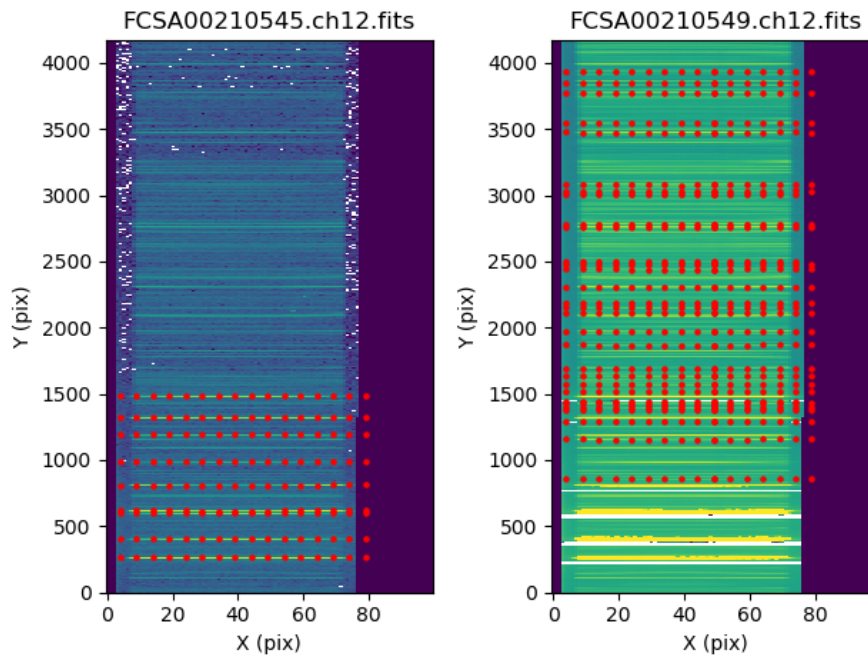
  -h, --help show this help message and exit

Figure 25. Example.

5.2. implot.py

usage: implot.py [-h] [-c C] [-w W] [-flgx] [-flgy] files

This is the script for plotting lines and columns of images. If two or more files are given, those are overplotted.

positional arguments:

  files   Input FITS files (comma separated)

optional arguments:

 -h, --help show this help message and exit

 -c C   Center coordinate (comma separated).

 -w W   Center coordinate (comma separated).

 -flgx   Plot along X axis.

 -flgy   Plot along Y axis.

5.3. monoimage.py

USAGE: monoimage.py [-h] [-c1 CONTINUUM1 CONTINUUM1]
          [-c2 CONTINUUM2 CONTINUUM2] [-en EXTNUM]
          INCUBE OUTFITS ONBAND ONBAND

positional arguments:

 INCUBE   Input data cube

 OUTFILE   Output fits name

 ONBAND   Start and end wavelengths of the on-band (A)

OPTIONS:

 -h, --help   show this help message and exit

 -c1 CONTINUUM1 CONTINUUM1, --continuum1 CONTINUUM1 CONTINUUM1
          Continuum wavelength range of one side (A)

 -c2 CONTINUUM2 CONTINUUM2, --continuum2 CONTINUUM2 CONTINUUM2
          Continuum wavelength range of other side (A)

 -en EXTNUM, --extnum EXTNUM
           FITS extension number to be used. Dfault is 0.

This script creates an image by integrating over ONBAND. When CONTINUUM1 and
CONTINUUM2 are specified, the interpolated continuum level is subtracted.

5.4. show_skyshift.py

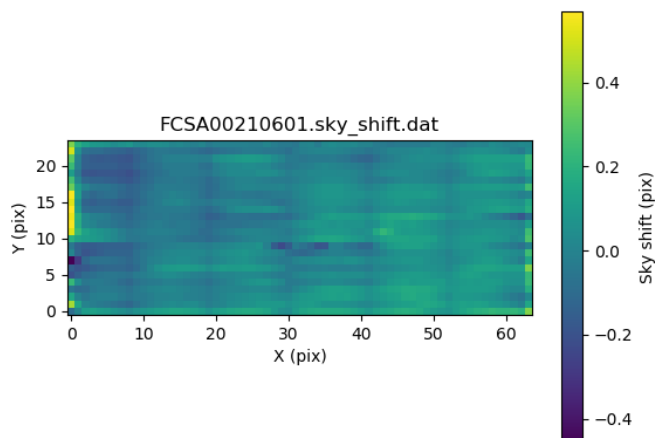usage: show_skyshift.py [-h] [-vmin VMIN] [-vmax VMAX] infile

This is the script for showing sky shift derived with get_sky_shift.py.

positional arguments:

 infile   Sky shift data file

optional arguments:

 -h, --help show this help message and exit

 -vmin VMIN Minimum value (default: None)

 -vmax VMAX Maximum value (default: None)

Figure 26

5.5. zero_padding.py

USAGE: zero_padding.py [-h] <Input cube>

>   ARGUMENT

>>   Input cube: Input data cube file name

This script replaces pixel values of 0.0 around the edge of the wavelength direction to NaN. The input data cube is overwritten.

6. Citing and references

Please cite the following paper.

>   Ozaki, S. et al. 2020, Publications of the Astronomical Society of Japan, 72, 97

Other papers are listed below for reference.

>   Ozaki, S. et al. 2014, Proc. SPIE, 9151, 915149

>   Ozaki, S. et al. 2012, Proc. SPIE, 8450, 84501L

7. Contact

If you have any questions, please don't hesitate to ask.

shinobu.ozaki@nao.ac.jp

8. Changelog

8.1. 20220817 => 20230831

- bias_overscan.py was modified. The previous version showed a systematic error of a few counts after the bias subtraction.

- The new option -bo was added to bias_overscan.py to use an upper overscan region for a bias subtraction instead of template bias files. See the detail here.

8.2. 20210818 => 20220817

- New scripts (objecterr.py and errestimate.py) were introduced for creating an error data cube thanks to Isobe-san's contribution.

- Some scripts were modified for the error estimation.

- rmidpoints.py was newly implemented. This automatically eliminates most of bad data points identified by identify_dispersion.py.

- In the sky subtraction process, object.py uses the old script for old data before August 2021 because they used the old sky slit.

8.3. 20200210 => 20210818

- This version was adapted to the new sky slit.

- The option"ql" was implemented in some scripts.

- This version was adapted to VPH680+SY47.

- ~~From this version, a data cube is trimmed after sky subtraction, and the wavelength range becomes the one which is covered in the whole field of view.~~

8.4. 20201003 => 20210210

- zero_padding.py is now executed at the end of object.py when users demand it.

- In mkedgeimage.py, neighboring spectrum areas are not padded with 0 to avoid slight uncertainty (< 1 spatial pixel ~ 0.2 arcsec by default) of defining spectrum edges.

- Explanations of comp_emis_position.py and show_skyshift.py are added in this manual.

8.5. 20200213 => 20201003

- Sky subtraction method has been modified to decrease residuals of sky emission lines. In the new method, wavelength deviations between a sky spectrum and object spectra of each spaxel are derived from a comparison data. For this purpose, a comparison data cube is created.

- Previously, blank pixels around spectrum edges are padded with NaN in transform.py (see section 8.3). Now this function is moved to the separate script, zero_padding.py.

8.6. No version number change

- Incorrect descriptions about the arguments for standardstar.py and object.py have been corrected. There was no description about the required argument, "Comparison ID for dome flat", in the previous manual.

8.7. 20200210 => 20200213

- In transform.py, blank pixels are padded with NaN. In the previous version, they are padded with 0.

- After flux calibration, unit has been changed from ergs/s/cm2/A to 1E-20 ergs/s/cm2/A. Because of this change, Auto Plot 3D function of SAO image ds9 works well.

8.8. 20190621 => 20200210

- "os.path.join" has been used to add a path to a file name.

- In fitcoord_dispersion.py, xorder and yorder have been changed from 7,7 to 5,3.

- In terms of WCS, only the CD matrix has been used. In the previous version, both the CD matrix and the CDELT keywords were included in the FITS header.

- Some minor bugs have been fixed.

- In Section 4.2.6, the explanation has been slightly changed, and Figure 11 and Figure 12 have been changed.

- In this manual, figure numbers have been corrected.

8.9. 20200210 => 20210818

- This version was adapted to the new sky slit.

- The option "-ql" was implemented in some scripts.

- This version was adapted to VPH680+SY47.

- From this version, a data cube is trimmed after sky subtraction, and the wavelength range becomes the one which is covered in the whole field of view.