`flame` user manual

Sirio Belli

July 7, 2017

1 Introduction

`flame` is a pipeline for reducing near-infrared multi-object spectroscopic data, written in IDL. Although created specifically for the LUCI instrument at LBT, it has been designed in a modular way and can be easily adapted to work with the data produced by different instruments. The present document describes in detail the reduction of LUCI data. Section 6 summarizes the different settings needed for the reduction of data taken with different instruments.

2 Installation

2.1 Requirements

The following three external IDL libraries must be installed before running `flame`.

1. NASA's IDL Astronomy User's Library;
2. David Fanning's Coyote Library;
3. Craig Markwardt's mpfit (only the mpfit.pro and mpfit2dfun.pro files are needed).

Make sure to have the most recent versions of these libraries (at least more recent than the January 2016 release).

Other third-party routines used by `flame` are included in the distribution, in the flame/lib/ directory. These are the B-spline routines from the `idlutils` library, the L.A.Cosmic routine written by Joshua Bloom and Pieter van Dokkum, and the readmhdufits routine written by Marc Kassis.

2.2 Download

Download `flame` from `http://siriobelli.github.io/flame/`. You can either save it in the directory where you keep your IDL code, or save it somewhere else and then add the flame/ directory to the IDL path. No other step is necessary.

3 Setting up the Data Reduction

3.1 Basic input

Go to the directory that you want to use for the reduction of a data set (it is recommended to create a new directory for this purpose). Then:

- Copy the driver file (`flame_driver_luci.pro`) from the flame/pro/ directory to the current directory.
- Create a text file (e.g., science.txt) with, in each line, the full name, including the absolute path, of a raw science frame (either FITS files or gzipped FITS files).
- *(optional)* Create additional text files with the names of the calibration frames (e.g., darks.txt, flats.txt, etc.)

The data reduction is controlled entirely via the driver file. Open the local copy with a text editor and have a look at the code. The driver file is divided into three parts. The first part is where the input parameters are set; the second part is the initialization, and the third part is the data reduction. In this section we discuss the first part.

The first step in the driver file creates the `input` structure:

```
 input = flame_create_input()
```

We now need to populate the fields in the `input` structure with values that are appropriate for the data set that we are reducing.

3.1.1 Science frames

We start by specifying the name of the file containing the science frames; for example:

```
input.science_filelist = 'science.txt'
```

If the file is not in the current working directory, then the absolute path must be provided. However it is recommended to run the data reduction from the same directory in which the input files and the driver file are stored.

3.1.2 Dithering

Near-infrared observations are usually spatially dithered for an effective sky subtraction. `flame` supports a two-point dithering, also called AB dithering, which is activated via the flag:

```
input.AB_subtraction = 1
```

3.1.3 Reference star

When observing faint targets, it is recommended to place a slit on a relatively bright star (hereafter called the reference star) from which the observing conditions and the spatial dithering can be accurately measured. In the AB dithering scheme, the star trace will appear on two different positions, which may be in the same slit or in two different slits. The y pixel position of the reference star in both the A and the B position must be provided:

```
input.star_y_A = 520
input.star_y_B = 560
```

These coordinates do not need to be exact. If the trace of the reference star is very faint in a single frame, a simple A-B subtraction is generally sufficient for a visual detection. When no reference star is available, the dithering pattern must be specified manually (see Section 5). If no AB dithering is performed, the B star position should be set to zero.

3.1.4 Slits to be reduced

While the default behavior is to reduce all the slits in the mask, it is sometimes preferable to reduce only one slit, for example to obtain the reduction of one representative object while observing. This can be achieved by specifying:

```
input.reduce_only_oneslit = 2
```

where in this case we would get the reduction of only the second slit. The default value is zero which means that all the slits will be reduced.

3.1.5 Longslit observations

Finally, to reduce longslit observations, it is sufficient to set the longslit flag:

```
input.longslit = 1
```

Because the longslit typically spans the whole detector in the vertical range, the edges of the longslit are not well defined. The region of interest must be specified by providing the y pixel coordinates of the top and bottom limits:

```
input.longslit_edge = [1050, 1350]
```

If longslit_edge is set to [0,0], the default is to reduce the full frame excluding 10% of pixels at the bottom and 10% at the top. For example, for a frame that is 2048 pixels tall, this would correspond to the [205, 1843] range. Since the slit curvature cannot be traced, the spatial distortion will not be rectified for longslit data.

3.2 Optional input

The `input` structure contains additional fields that can be used to specify optional settings. The full list is reported in Table 1. The expanded driver file `flame_driver_complete.pro` includes all the options: the relevant lines can be copied and then pasted in the local copy of the driver.

3.2.1 Dark frames

Dark frames, if provided, are used to construct the median-combined master dark, which is then subtracted from each science frame. When adopting an AB dithering, the dark current is automatically corrected for during the A-B subtraction, and dark frames are not needed.

The dark frames, as well as all the other calibration frames, are supplied to the pipeline via text files containing a list of FITS files, similarly to what done for the science data:

```
input.dark_filelist = 'dark.txt'
```

The only difference is that all the calibration files are optional, and the default value for the filelist parameter is always `'none'`.

3.2.2 Flat field

A flat field is a calibration frame taken with the purpose of removing unwanted patterns in the data. However, there are many ways in which flat fields can be obtained: using a lamp or the twilight sky; with or without the slitmask; with or without the dispersion element; and with different filters, cameras, and so on.

In `flame` the flat fields are divided into three types, based on how they are used during the data reduction, rather than what instrument configuration was adopted in taking the calibrations. This means that the user is free to choose the actual configuration for each type of flat field.

The **pixel flat field** is used to correct for the pixel-to-pixel variation in sensitivity. First the large-scale variation in the illumination is removed via polynomial fitting, and then the flat field is divided by its median pixel value. Pixel flats work best if taken without slitmask, in order to obtain a smooth illumination of the full detector. If the imaging mode produces small-scale patterns due to the presence of dust, the dispersion element can be used (with or without a long slit). The LUCI detectors present pixel-to-pixel variation of the order of a few percent. These variations vary with time, and often times it is better not to flat field the data rather than using a flat field obtained in a different observing run.

The **illumination flat field** corrects for the uneven illumination of the slit along the spatial direction.

Finally, the **slit flat field** is used to identify the slit traces and trace the geometric distortion of the slit edges.

Each of the three types of flat field can be specified in the usual way, for example:

```
input.pixelflat_filelist = 'pixelflat.txt'
input.illumflat_filelist = 'none'
input.slitflat_filelist = 'slitflat.txt'
```

3.2.3 Arcs

The near-infrared sky spectrum is dominated by OH emission lines, which are ideal for wavelength calibrating the observations. However, in particular cases these OH lines are not strong enough (for example when the exposure time is very short) or do not cover the full observed range (this can happen with high-resolution observations in the K band). Also, spectra taken with optical instruments have very few sky emission lines. In all these cases, it is necessary to observe the spectrum of arc lamps in order to obtain a reliable wavelength calibration. These arc frames can be specified via:

```
input.arc_filelist = 'arcs.txt'
```

Table 1. Content of the `input` structure

Field	Default Value	Description
science_filelist	science.txt	name of ASCII file containing the list of science frames
AB_subtraction	1	flag to enable AB subtraction
input.star_y_A	0	pixel y-coordinate of the A position of the star trace
input.star_y_B	0	pixel y-coordinate of the B position of the star trace
reduce_only_oneslit	0	if non-zero, reduce only that one slit
longslit	0	set to one to reduce long-slit data
longslit_edge	[0, 0]	when reducing long-slit data, only consider this range of pixel y-coordinate
dark_filelist	none	name of ASCII file containing the list of dark frames
pixelflat_filelist	none	name of ASCII file containing the list of frames for pixel flat field
illumflat_filelist	none	name of ASCII file containing the list of frames for illumination flat field
arc_filelist	none	name of ASCII file containing the list of lamp arc frames
slitflat_filelist	none	name of ASCII file containing the list of slit flats
slitflat_offset	0	vertical offset in pixels between the slit flat and the science frames
slit_position_file	none	name of ASCII file containing the approximate y-coordinates of the slit edges
dither_file	none	name of ASCII file containing the list of dither positions
max_slitwidth_arcsec	0	threshold value of the slit width to identify the alignment boxes
intermediate_dir	intermediate/	directory that will contain the intermediate products
output_dir	output/	directory that will contain the final output

3.2.4 Dithering positions

It is advisable to always have a reference star on the multi-slit mask. When this is not possible, `flame` cannot measure the dithering offset for each frame, and the dithering pattern needs to be saved in an ASCII file. The file must contain a number of lines equal to the number of science frames. Each line will have the value of the vertical offset, in arcseconds, corresponding to that frame (for example 5, -5, 5, -5, etc). Then the file name needs to be input in the `input` structure:

```
input.startrace_y_pos = [0, 0]
input.dither_file = 'dither.txt'
```

3.2.5 Directories

Next, we need to define the directories for intermediate and final outputs:

```
input.intermediate_dir = 'intermediate/'
input.output_dir = 'output/'
```

If these directories do not exist, they will be created by `flame`. Note that these definitions use relative paths, but absolute paths can also be used.

3.3 Initialization

The data reduction in `flame` consists of a series of steps, which are represented by individual routines. The information is carried from one routine to the next via the use of only one structure, named `fuel`, which contains all the input parameters set by the user, together with all the internal variables.

Once the `input` fields have been specified, the `fuel` structure must be created:

```
fuel = flame_initialize_luci(input)
```

This will also check for potential errors in the inputs, and create the necessary directories.

The `fuel` structure consists of five substructures:

- fuel.input: a copy of the `input` structure created in the previous step;
- fuel.util: utility variables needed by `flame`;
- fuel.instrument: instrument-specific information;
- fuel.diagnostics: diagnostics such as seeing and transparency for each of the science frames;
- fuel.slits: information relative to each slit, such as position on the detector and expected wavelength range.

Some of the fields in the `fuel` structure are initially set to null pointers and will be created by successive steps.

During the initialization, relevant fields from the FITS header of the science files, such as band, central wavelength, and slit positions are read and saved. This is the only step during the data reduction where instrument-specific operations are performed. Since the remaining steps are common to all near-infrared MOS data, it is in principle easy to adapt `flame` to a different instrument by simply writing the appropriate initialization module.

4 Running the Data Reduction

Once the first part of the driver file has been edited, the data reduction is fully automatic. One way to run `flame` at this point would be to execute the driver directly from the command line:

```
> idl flame_driver.pro
```

Another possibility, which allows some interaction and testing, is to copy line-by-line the content of the driver file onto an interactive IDL session. In this way one can explore the output files and the fuel structure after every step.

The second part of the driver file contains eleven steps or modules, which we are going to examine in detail in this section. Every module is an IDL routine whose name starts with `flame_` and which is saved in a .pro file with the same name in the directory flame/pro/. All modules accept (and require) only one argument, which is the `fuel` structure, so that the IDL code for each module looks the same:

```
flame_modulename, fuel
```

Every module edits the fuel structure and/or outputs files in the intermediate directory. Only the last module, which creates the final results, saves the output files in the output directory.

4.1 `flame_diagnostics`

The first step produces important diagnostics which will be used by the other modules. These diagnostics can also be helpful to the user to assess the quality of the observations, and can be run in real time while observing, ideally after every new frame or frame pair has been taken.

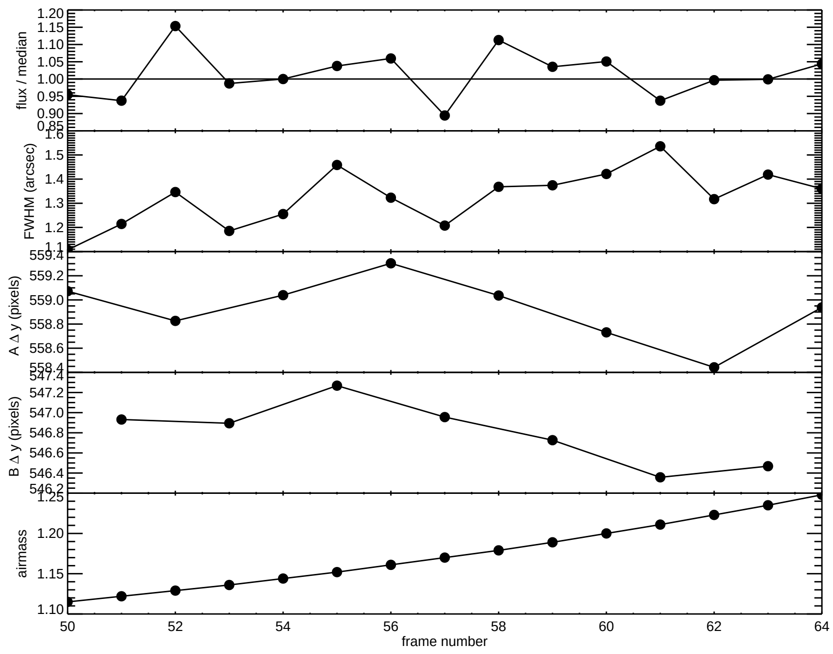

Figure 1 Example of diagnostic plots. From the top: Flux of the star trace, normalized by the median value; FWHM in arcsec; vertical position of the trace in A frames; vertical position of the trace in the B frames; airmass value read from the FITS header.

Starting with the y coordinates specified by the user in fuel.startrace_y_pos, this module detects the reference star position on each frame, and identifies which frames belong to the *A* and which to the *B* position (and assign an *X* position when no star trace is found). Then, for each frame a Gaussian profile is fit to the star trace, from which the flux, vertical position and FWHM are measured. In addition, the airmass value is read from the FITS header of each frame.

All these quantities are saved in the substructure fuel.diagnostics. They are also output in the ASCII file diagnostics.txt, and plotted as a function of the frame number in the file diagnostics.ps. One example of such file is shown in Figure 1. Because of the dithering, the vertical position is shown in different panels for the *A* and the *B* frames. If there are multiple frames with same frame number, which can happen when combining data from different nights, then the x axis of the plot will show a sequential number (but the original frame numbers are used in the ASCII file).

The diagnostic plots can be used to judge the trend of transmission (including the effect of cloud cover), seeing, and drift. In addition to help the user make informed decisions while observing, these measurements are also useful to identify the bad frames that should be excluded from the final data reduction.

4.2 `flame_quickstack`

The second step produces a FITS file which is the difference of a stack of the A frames and a stack of the B frames (as defined in the previous module). This step is also useful while observing, since the simple $A - B$ subtraction often yields a decent sky removal which allows the detection of faint continuum traces or emission lines.

The sole goal of this module is to allow a quick look at the data. This step can be skipped with no consequences for the remaining parts of the data reduction.

4.3 `flame_correct`

This module applies a number of corrections to the raw science frames. First, the bad pixel mask is generated from the input dark and/or flat frames. If neither dark nor flat field frames were supplied, then the default bad pixel mask, stored in flame/data/, is used.

The following corrections are applied:

- Cosmic rays removal: this step is not recommended for near-infrared data, but can be necessary for observations in the optical.
- Linearity correction: the value of each pixel is corrected to take into account the non-linear response of the detector. The correction is a polynomial function with instrument-specific coefficients, which are determined during the initialization.
- Pixel flat field correction, if required.
- Bad pixels are set to NaN.
- The flux of each pixel is multiplied by the gain and divided by the exposure time. This converts the flux units from ADUs to electrons/second.

The corrected science frames are then saved in the intermediate/frames/ directory. Each FITS file contains a second extension with the error spectrum, which is calculated as the combination of read-out noise and Poisson noise.

4.4 `flame_getslits`

The module `flame_getslits` is responsible for detecting the slits from the multi-object slitmask. The vertical coordinates of the slit edges calculated during the initialization are used. However, the physical position of the mask with respect to the detector is typically difficult to predict and varies slightly from one night to the other. To account for this, a vertical shift is calculated by identifying the slit edges.

Once the approximate pixel position of each slit has been calculated, a second, more refined slit edge detection is run. This is based on the identification of individual OH emission lines, which are typically very bright and can be reliably used to trace the edge of a slit. For each slit, a low-order polynomial is fit to the y coordinate of the edge as a function of x position, and the results are saved to the fuel structure. Note that this procedure is more likely to fail when the slits are short or when there is not much space between one slit and the next one. When observing in the K, the low number of OH lines and the strong continuum sky emission makes it convenient to use the sky background instead (see Section 5.2.2)

The output file slitim.fits is a mask image where the value of each pixel is an integer indicating the slit number to which that pixel belongs to, with zero indicating pixels that do not belong to any slit. Also, the file slits.reg is saved, which is a ds9 region file that can be loaded on top of any science frame and will show the slit edges (see Figure 2). This can be used to check the slit detection and also to identify which slit on the science frame corresponds to which target.

Explain the option of manually specifying the slit edges for difficult cases.

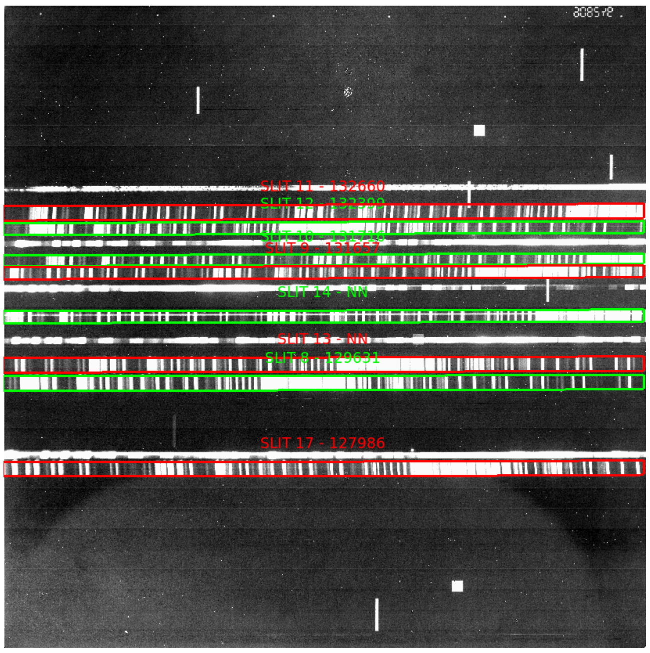

Figure 2 An example of LUCI raw frame, with the regions marking the slits identified by `flame` shown in green. The slits that are not marked are the boxes for the alignment stars, which are ignored by the pipeline.

4.5 `flame_cutout_slits`

Using the slit definition from the previous step, the slits are extracted from each corrected frame and saved as FITS files. All the cutouts corresponding to the same slit are saved in the directory slitxx, where xx is the slit number.

4.6 `flame_wavecal_rough`

The wavelength calibration is arguably the most important and difficult step in the data reduction. An accurate calibration is not only useful for the scientific interpretation of the data, but is also necessary for a correct sky subtraction.

In `flame`, the wavelength calibration is split into three steps. The first step finds an approximate wavelength calibration for the central part of the slit, using only the first science frame. The sky spectrum is extracted from the central five pixels of the slit, and is compared to a model spectrum. The comparison consists of two successive steps:

1. Both the observed and the model sky spectra are smoothed to an intermediate spectral resolution ($R = 500$, but this value is stored in the input structure and can be changed by the user), and a series of values for the pixel scale (in micron per pixel) are tested. At each loop, cross-correlation is used to find the initial wavelength, and the pixel scale that gives the largest value of the cross-correlation is selected.

2. Once the pixel scale and the initial wavelength are roughly known, a finer comparison is performed. In this case the spectra are not heavily smoothed, so that the narrow OH lines can be used for a very effective comparison. The relation between x pixel coordinate and wavelength is typically not linear, and a second-order polynomial is used: $\lambda(x) = a_0 + a_1 x + a_2 x^2$. A fine grid of values for a_1 and a_2 is used, and at each loop cross-correlation is used to determine the best value for a_0.

3. Finally, another cross-correlation is performed at the native spectral resolution.

For each slit, plots illustrating the successive comparisons of observed and model sky spectra are saved as ps files (wavelength_solution_estimate.ps). An example is shown in Figure 3.

If the sky spectrum includes a strong contribution from continuum emission, as is often the case for the thermal emission in the K band, it may be necessary to fit and subtract a low-order polynomial to the observed spectrum before comparing it to the model. This can be done be setting `util.wavecal_rough_continuum_degree` to a number different from zero, which will be taken as the degree of the polynomial representing the continuum (typically about five or less).

4.7 `flame_identify_lines`

The previous procedure gives only a rough wavelength calibration that is valid for the pixel rows at the center of the slits. Since the wavelength solution typically varies with spatial coordinate within the slit (particularly for tilted slits), the calibration needs to be calculated for each individual pixel row, which is done in this step.

Initially, the spectrum of the central pixel row is extracted, and the individual OH emission lines are identified using the rough wavelength solution. A Gaussian fit is performed on each line, and the x pixel coordinate corresponding to the center is stored. It is therefore necessary that the calibration found in the previous step is accurate to within the window used in the Gaussian fit,

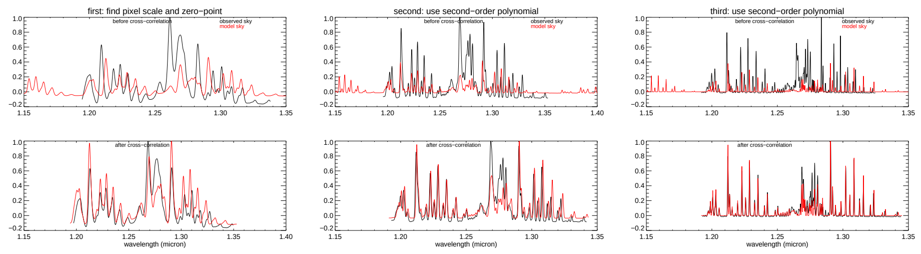

Figure 3 Approximate wavelength calibration. The sky spectrum (in black) is extracted from the central region of the slit, and is compared to a template spectrum (in red). The top panels show the initial guess, and the bottom panels show the results of the cross-correlation. The cross-correlation is repeated three times using successively higher spectral resolution.

otherwise the identification of the OH lines will not be possible. The relation between the x coordinate and the expected wavelength of each OH line is then fit with a low order polynomial (see example in Figure 4). This procedure is then repeated for the next pixel row, using the wavelength solution from the previous pixel row as first guess. This is always a good assumption as long as the wavelength solution varies smoothly with position, without abrupt changes from one pixel position to the next.

After all the pixel rows have been processed, the final result is a number of x, y coordinates of detected OH lines, and the corresponding theoretical wavelength. A ds9 region file (OHlines.reg) showing the detected location of the OH lines is produced.

4.8 `flame_wavecal_accurate`

For each sky line identification, we have a matching set of *observed* pixel coordinates (x, y) and *rectified* coordinates (λ, γ), where λ is the theoretical wavelength of the OH line and γ is simply the vertical distance (in pixels) from the bottom edge of the slit, calculated using the polynomial fits obtained by `flame_getslits`. The wavelength solution and the

4.9 `flame_skysub`

Because the OH emission lines vary on short timescales, the A-B subtraction typically leaves strong residuals. In order to improve the sky subtraction, `flame` adopts the method of Kelson 2003. Briefly, the wavelength solution is used to map each pixel from the observed, distorted frame to the output, rectified frame. The distortion in the observed frame (which is produced by the optics and/or by the tilted slits) produces a very fine sampling of the sky lines in the rectified frame. Because of this sub-pixel sampling, it is possible to obtain an excellent B-spline fitting of the emission lines. `flame` performs this operation on each frame for every slit, and saves a FITS file of the sky-subtracted data. A graphic window shows, during this step, the sky emission spectrum (collapsed onto one dimension) and the B-spline fit for every frame.

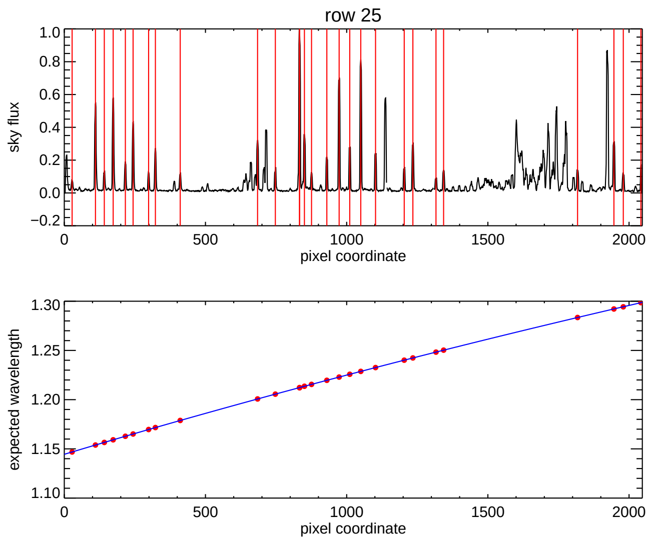

Figure 4 Identification of sky lines. Top panel: The sky spectrum is extracted from a single pixel row, and using the approximate wavelength calibration the OH lines are identified and fit with a Gaussian profile. Bottom panel: the measured position of each OH line, together with their known wavelength, allow an accurate wavelength solution, shown in blue.

4.10 `flame_rectify`

Using the 2D rectification coefficients calculated in `flame_wavecal_accurate`, this step performs both the wavelength calibration and the spatial rectification simultaneously, using only one interpolation. In practice, for each slit and each frame a polynomial warping is applied to the observed 2D spectrum, and a rectified FITS file is produced.

4.11 `flame_combine`

This step consists in the stacking (after sigma-clipping) of all the A and B frames. The individual A and B stacks, and the difference A-B, are output as FITS file for each slit. Non-skysubtracted versions of the stacks are also saved. Also, an ABcombined FITS file is produced by summing A-B to the shifted B-A images, so that one positive central trace, corresponding to the total observing time, can be extracted for the science analysis.

Sometimes the dithering is such that one object spends half of the time in one slit and half of the time in another slit. This is particularly useful when the slits need to be tilted in directions that are different from the dithering direction (as is the case when different objects in the mask require different position angles). If the dithering length, as calculated from the star traces, matches the distance between any two slits, then these are assumed to be matching slits and their A-B stacks are combined together. It is worth pointing out that this assumption breaks down if the star traces for the A and B positions actually belong to different stars.

4.12 `flame_checkdata`

Finally, this optional module performs a quality check on the reduced data set. If present, the reference star spectrum is analyzed and the effective seeing calculated. The file reference_star.ps is written, showing the spatial profile, the width and position of the stellar track as a function of wavelength, and the xtracted 1D spectrum of the star. Then, a ps file is written for each slit. The first plot shows the stacked sky spectrum, from which the sky lines are identified and fit, and the wavelength residuals and spectral resolution R are shown. Stats on the wavelength solution and the velocity resolution are shown at the bottom. The remaining two plots show respectively the line width and the wavelength residuals for each of the science frames reduced. The gray points are individual measurements for one pixel row of one sky line; the red points show the median value for each frame and the error bars include the central 68% of points. Looking at these figures it is easy to see if one of more of the frames do not have a good wavelength solution. In such cases the user can decide to run the reduction again excluding those frames, or trying to improve the wavelength solution by tweaking the settings in `fuel.util`.

5 Troubleshooting

5.1 Bad fit to sky lines

Sometime the sky lines are not bright or numerous enough for a reliable wavelength calibration. This can happen especially in the K band, where the OH lines are less numerous and the spectrum is typically dominated by thermal emission. Another situation where this issue can arise is when the OH lines are faint due to very short exposure times. In this cases it is always advisable to obtain arcs. There are however a few settings that can be optimized to obtain a satisfactory wavelength solution.

During the `flame_identify_lines` step, at each pixel row a polynomial is fit to the position of the sky lines (see Figure 4). This part is critical and is often the first to fail in sub-optimal conditions. If the number of sky lines identified in one row is too low, the fit will not be considered valid. It is possible to change this threshold number by changing the variable `fuel.util.identify_lines_Nmin_lines`. However, with a low number of lines it is also necessary to lwer the degree of the polynomial, to avoid over-fitting and obtaining meaningless wavelength solutions. This can be done by changing the variable `fuel.util.identify_lines_poly_degree`.

Another possibility is to manually exclude those sky lines that are clearly problematic. This can be done easily by commenting out one or more entries in the line list, which is stored as a text file in the flame/data/ directory.

5.2 Slit identification

If the slit identification fails or is not accurate, one alternative is to use the longslit option to manually select the region within one slit to reduce. By default the longslit option assumes that the slit is spatially centered on the mask. If this assumption causes a wildly wrong initial guess for the wavelength range, it is possible to override this value by changing the `fuel.slits.range_lambda0` variable, which is a 2-element array containing the minimum and maximum values to be considered for the wavelength of the first pixel.

Sometimes using a different science frame gives a better slit identification. The default frame used for the slit identification is the one in the middle; this file name is stored in `fuel.util.master_getslit`. This can be manually changed after initialization; or a different science frame can be specified via the `input.slitflat_filelist`, see below.

5.2.1 Manual identification of slit edges

Description of the manual input for the slit edges

When using curved slit with LUCI at LBT, the header keywords are not sufficient to correctly predict the position and size of the slits, and it is often necessary to specify the slit edges manually.

5.2.2 Detection of slit edges with sky background

In some cases, particularly in the K band, there are not enough bright OH lines for identifying the slit edges (see Section 4.4). It is possible to use the continuum emission instead, by specifying

```
input.trace_slit_with_skylines = 0
```

5.2.3 Detection of slit edges with slit flat field

If the slit edges are not well defined in the science frames, it is possible to use a flat field for the slit identification:

```
input.trace_slit_with_skylines = 0
input.slitflat_filelist = 'slitflat.txt'
input.slitflat_offset = 4
```

where the file slitflat.txt contains a list of file names corresponding to the slit flat field data. These can be the same as the pixel and/or illumination flats. It is often the case that the science frames present a vertical shift compared to the calibration frames because of flexure. In this case the shift needs to be measured manually and specified via the `slitflat_offset` setting.

5.3 Specify the x coordinates for the reference star

When the reference star is present, its trace is extracted and used as a diagnostic tool (see Section 4.1). Only the central part of the trace is used, in the interval [1000, 1200] in x-pixel coordinates. If the trace is too faint in this central region, it is possible to define a different range for the extraction:

```
input.xrange_star = [100, 500]
```

5.4 Skip slits

It is possible to specify which slits to skip during the data reduction by setting the `skip` flag in the `fuel.slits` structures. When this flag is set, the slit is ignored by the pipeline.

Table 2. Settings in the `fuel.util` structure

Field	Default Value	Description
star__x__range	[1000, 1200]	range of star trace (in pixel x-coordinate) that will be extracted
rough__wavecal__R	500	target spectral resolution for the rough wavecal calibration
trace__slit__with__skylines	1	if set, the slit detection is based on the sky emission lines instead of the continu

Setting the `input.reduce_only_oneslit` parameter is equivalent to setting the `skip` flag in each slit except one.

The `skip` flag is automatically set whenever an error is encountered, so that the subsequent steps can be regularly carried on for the remaining slits.

5.5 Line identification issues

When fitting the wavelength solution row by row, there is a minimum number of sky lines that must be identified, otherwise that pixel row will be skipped. In some cases, it is possible that there are very few lines available in the observed wavelength range (for example in part of the K band), and all or most of the pixel rows will be skipped. In such cases it is possible to lower the minimum number of required line identification by changing the parameter `fuel.util.identify_lines_nmin_lines`.

Explain also `fuel.util.identify_lines_poly_degree` and `fuel.util.identify_lines_linefit_window`.

6 Other instruments

6.1 Keck LRIS

Credits

`flame` was written by Sirio Belli with substantial help from Alessandra Contursi and Tom Fletcher for development and testing. Many features were inspired by code written by Eva Wuyts and Ric Davies for the LUCAS pipeline, and by Nick Konidaris' MOSFIRE DRP. Dave Thompson provided useful information particularly for the LUCI-specific parts.