ELT – MICADO

SpecCADO:
User Manual

Issue: 0.1.3

Date: 19 February 2019

Prepared:	O. Czoske	2019-02-19	
	Name	Date	Signature
Approved:			
	Name	Date	Signature
Released:			
	Name	Date	Signature

Change record

Issue/Rev.	Date	Section/Parag. affected	Reason/Initiation/Documents/Remarks
0.1.3	16/2/2019	All	Layout initialised

Contents

1 Scope

This document is a brief manual to get users started with SpecCADO, the spectroscopic simulator for MICADO. SpecCADO is a temporary package, intended to be merged into SimCADO, the general instrument data simulator for MICADO in the near future.

SpecCADO depends on SimCADO. For an in-depth description of SimCADO, we refer to the PDR documentation [1].

2 Installation

2.1 Install SimCADO

SpecCADO relies on a prior installation of SimCADO. Download the SimCADO source package from https://homepage.univie.ac.at/oliver.czoske/SimCADO-0.5dev1.tar.gz. Unpack the tar file and change to the source directory:

```
tar xvf SimCADO-0.5dev1.tar.gz
cd SimCADO-0.5dev1/
```

Install the package by doing

```
pip install .
```

or

```
pip install --user .
```

The first version may require root permissions; use the second form to install into a user directory.

If the `pip` and `python` commands point to a python-2.7 installation, please try the commands `pip3` and `python3` instead.

The subdirectory `data/` of the SimCADO source directory contains MICADO-specific data files that are needed for simulations to be performed with SimCADO or SpecCADO. As these data are not installed, you will need to tell the software the absolute path to this directory via the parameter `SIM_DATA_DIR`.

2.2 Install SpecCADO

The SpecCADO source package can be downloaded from https://homepage.univie.ac.at/oliver.czoske/speccado-0.1.3.tar.gz. Unpack the tar file and change to the source directory:

```
tar xvf speccado-0.1.3.tar.gz
cd speccado-0.1.3/
```

It is also possible to obtain SpecCADO from the github repository at https://github.com/oczoske/SpecCADO. Note that the latest snapshots from the repository may not always be perfectly functional.

Install the package by doing

```
pip install .
```

or

```
pip install --user .
```

To test the installation, start a `python` or `ipython` session and do

In 1:
```
import speccado
```

In 2:
```
speccado.__version__
```

Out2:
```
'0.1.3'
```

In 3:
```
speccado.bug_report()
```

```
Python:
 3.6.0 (default, Feb  6 2017, 15:51:19)
[GCC 4.9.2]

speccado :  0.1.3
simcado :  0.5dev1
astropy :  3.0.5
synphot :  0.1.2
numpy :  1.15.4
scipy :  1.1.0
poppy :  0.7.0
wget :  3.2

Operating system:  Linux
         Release:  4.9.0-0.bpo.8-amd64
         Version:  #1 SMP Debian 4.9.110-3+deb9u5~deb8u1 (2018-10-03)
         Machine:  x86_64
```

The last command checks that the dependencies are fulfilled. Please include the output of this command whenever you report a problem or a possible bug.

The necessary dependencies to run `speccado` are `simcado`, `astropy`, `synphot`, `numpy`, and `scipy`. `wget` is required for the following step, poppy is not needed.

If you want to simulate detector read-out noise, you require the file `FPA_noise.fits`. You can obtain a prepared file by doing

In 4:
```
import simcado as sim
sim.get_extras()
```

Unfortunately, this will download a lot of other stuff as well.

3 Example scripts

The subdirectory `example/` in the SpecCADO source distribution includes two example scripts that demonstrate how SpecCADO works:

- `simulate_example.py` – simulates spectral traces on the MICADO detectors. The slit contains two stars, both of which have the same spectrum, viz. that of GW Ori, dimmed by 9 magnitudes to put it into the usual target range of MICADO.[1] Atmospheric emission is included as a background spectrum that fills the entire slit. The spectrum provided in `atmo_emission.fits` was computed by `skycalc`.[2]

- `rectify_example.py` – takes a simulated detector frame as input and recifies it into two-dimensional $\xi - \lambda$ images (ξ is the coordinate along the slit) from which one-dimensional spectra can be extracted in the usual way (e.g. using `apextract` in Iraf). Note that this is a perfect rectification that reverses the transformation applied by `simulate_example.py`. It does not include the uncertainties associated to tracing the spectra in a proper data reduction recipe.

The `example/` directory includes three configuration files set up for MICADO spectroscopy using the IzJ, J and HK order sorting filters, respectively. If you run your own simulations, start from one of these configuration files and modify it according to your needs.

The first parameter in the files is `SIM_DATA_DIR`. This is the directory where SimCADO and SpecCADO look for data files and needs to be set to where this directory is located on your system. Typically, this will be the data directory in the SimCADO source directory (see Sect. 2.1), as in

```
SIM_DATA_DIR      /path/to/simcado-0.5dev1/data
```

When running the example scripts, the data directory can optionally also be specified on the command line.

To simulate an HK spectrum of the sources as described above, type the following at the shell prompt:

```
python simulate_example.py spectro_HK.config <SIM_DATA_DIR>
```

It may take about 20 minutes to simulate the nine detectors of the MICADO focal-plane array. SpecCADO writes a stream of diagnostics to the screen – if all goes well this should not be of interest to anyone except the developer.

The output file is a multi-extension FITS file (one extension per detector), named `detector-<dateTtime>.fits` with the time stamp of the end of the simulation, for example

```
detector-2019-02-14T17-57-20.fits
```

To rectify the spectra run

```
python rectify_example.py detector-2019-02-14T17-57-20.fits
```

[1] No claim of scientific realism is made for this example...
[2] https://www.eso.org/observing/etc/skycalc/

This currently creates a separate FITS file for each order that is defined over the entire MICADO wavelength range. Due to the order sorting filter employed in the simulation, this means that most of the output files will not contain any signal. The file names are `ORDER-xx_yy.fits` with the numbers identifying the orders and cross-orders. The output of `rectify_example.py` will be improved.

4 Simulation of 2D spectra

In this section, the simulation of detector images will be described in more detail. This should enable the user to specify sources of their own.

Any simulation with SpecCADO or SimCADO starts by loading the configuration file:

In 5:
```
import speccado as sc
import simcado as sim

sim_data_dir = "/path/to/simcado-0.5dev1/data/"
cmds = sim.UserCommands("spectro_HK.config", sim_data_dir)
```

The parameter `sim_data_dir` can be ignored if `SIM_DATA_DIR` is set correctly within the configuration file.

It is possible to set individual parameters separately, such as the exposure time:

In 6:
```
cmds['OBS_EXPTIME'] = 60
```

For the simulation of science cases, it is often helpful to turn off detector saturation:

In 7:
```
cmds['FPA_LINEARITY_CURVE'] = 'none'
```

as this makes it possible to simulate long effective exposure times without the need to break the integration up into several sub-exposures to avoid saturation.

Currently, SpecCADO only simulates point sources (represented by the PSF) and background sources (that fill the slit homogeneously).

For point sources, two lists need to be defined, one giving the files holding the 1D spectra and one giving the positions of the sources in the slit. The following example assumes two stars are positioned on the centre line of the slit, both have the same spectrum:

In 8:
```
specfiles = ['GW_Ori+9mag.fits', 'GW_Ori+9mag.fits']
sourcepos = [[-1, 0], [1, 0]]
```

The stars are positioned 1 arcsec on either side of the centre of the 3 arcsec slit ($\xi = \pm 1$, $\eta = 0$, as defined in Sect. 6.3).

Another list gives the background spectra. Here, only one spectrum for atmospheric emission is provided:

In 9:
```
bgfiles = ['atmo_emission.fits']
```

From these lists, a `SpectralSource` object is created by

In 10:
```
srcobj = sc.SpectralSource(cmds, specfiles, sourcepos, bgfiles)
```

The simulation further requires objects for the PSF and the detector array:

In 11:
```
psfobj = sc.prepare_psf(cmds['SCOPE_PSF_FILE'])
detector = sim.Detector(cmds, small_fov=False)
```

For the detector, we use the Detector class from SimCADO. The PSF is a slightly extended form of the SimCADO PSF class, created through a SpecCADO function.

The transmission of the optical system (telescope plus instrument) is extracted from SimCADO's OpticalTrain class and stored as an interpolation object that can be evaluated at any wavelength required for the simulation:

In 12:
```
opttrain = sim.OpticalTrain(cmds)
tc_lam = opttrain.tc_mirror.lam_orig
tc_val = opttrain.tc_mirror.val_orig

from scipy.interpolate import interp1d
transmission = interp1d(tc_lam, tc_val, kind='linear',
                        bounds_error=False, fill_value=0.)
```

The optical layout of the spectral traces in the detector focal plane is described in a FITS file that holds a number of table extensions, each describing one spectral order.[3]. It is loaded by

In 13:
```
tracelist = sc.layout.read_spec_order(cmds['SPEC_ORDER_LAYOUT'])
```

The simulation of the full detector array is then run by

In 14:
```
sc.do_all_chips(detector, srcobj, psfobj, tracelist, cmds,
                transmission)
```

The result is written to disk as the multi-extension FITS file detector-YYYY-MM-DDThh-mm-ss.fits.

It is also possible to simulate just a single detector from the MICADO array:

In 15:
```
sc.do_one_chip(detector.chips[3], srcobj, psfobj, tracelist, cmds,
               transmission)
```

This simulates chip number 4 of the MICADO array (as python arrays are zero-offset). The output file in this case is called chip-YYYY-MM-DDThh-mm-ss.fits.

[3] The most up to date description is specorders-180629.fits, based on data provided by Frank Grupp.

5 Input file format

5.1 Spectra

Input spectra are expected to provided as one-dimensional FITS images (`NAXIS = 1`). For the transformation from pixels to wavelengths, a WCS is required in a format that can be read by the `astropy.wcs` module; this includes most formats defined by Greisen et al. (2006) [2]. To keep things simple, spectra sampled on a linear wavelength grid (`CTYPE1='WAVE'`) are recommended. However, frequency or wave number grids, linear or non-linear, should be possible, if the WCS is constructed correctly.

As an example, the WCS of the file `GW_Ori+9mag.fits` used in Sect. 4 is

```
NAXIS   =                    1 / number of array dimensions
NAXIS1  =                24750
WCSAXES =                    1 / Number of coordinate axes
CRPIX1  =                  1.0 / Pixel coordinate of reference point
CDELT1  =                6E-11 / [m] Coordinate increment at reference point
CUNIT1  = 'm'                  / Units of coordinate increment and value
CTYPE1  = 'WAVE'               / Vacuum wavelength (linear)
CRVAL1  =            9.9402E-07 / [m] Coordinate value at reference point
```

The pixel values have to be in physical units and the units have to be provided explicitely in the `BUNIT` keyword. SpecCADO uses the very flexible `astropy.units` module to convert units:

Input spectra for point sources can be given as photon or energy fluxes; internally, the units are converted to photons/s/m^2/µm. Permitted values for `BUNIT` include

- `'erg / (Angstrom cm2 s)'`
- `'ph / (um m2 s)'`
- `'1 / (nm cm2 s)'`

Input spectra for background sources can be given as photon or energy flux densities; internally, the units are converted to photons/s/m^2/µm/arcsec^2. Permitted values for `BUNIT` include

- `'ph / (s m2 micron arcsec2)'`
- `'erg / (s cm2 Angstrom arcmin2)'`

5.2 PSFs

The PSF used to place point sources on the slit is provided as a FITS image in a format that can be read by the SimCADO `psf` module.

The FITS header needs to specify the (effective) wavelength at which the PSF applies in keyword `WAVE0` or `WAVELENG`. SimCADO requires the units to be µm.

In addition, the size of the pixels in the PSF image is required. This can be given by the keyword `PIXSCALE` or the WCS keywords `CDELT1` or `CD1_1` (for the PSF a WCS with `CTYPEi='LINEAR'` is adequate). The units are arcsec.

6 Coordinate systems

6.1 Detector coordinates

Detector pixels are characterised by coordinates (c, i, j), where c is the number of the chip in the MICADO focal plane array ($c = 1, \ldots, 9$; see Fig. 1) and i and j are the column and row number of a pixel ($i, j = 1, \ldots, 4096$).

6.2 Focal plane coordinates

A position in the focal plane of MICADO is characterised by the coordinate pair (x, y), given in millimeters. The origin $x = 0$, $y = 0$ is taken to coincide with the centre of detector 5, i.e. pixel coordinates $i = 2048.5$, $j = 2048.5$. The x coordinate increases along rows of the chips (increasing i), the y coordinate increases along columns of the chips (increasing j). Small rotations of the detectors with respect to the focal plane coordinates are allowed.

The locations of the detector chips within the focal plane are specified in a focal-plane array definition file that specifies for each chip the identification number `id`; the position of the chip centre `x_cen`, `y_cen`; its half width `xhw`, `yhw`; the number of pixels in each direction `x_len`, `y_len`; the physical pixel size `pixsize`; a rotation angle of the detector rows with respect to the x-axis, `angle`; and the detector `gain`. Listing 1 shows the focal-plane array definition file for MICADO.

The transformation between pixel coordinates and focal-plane coordinates is a linear transformation. SimCADO

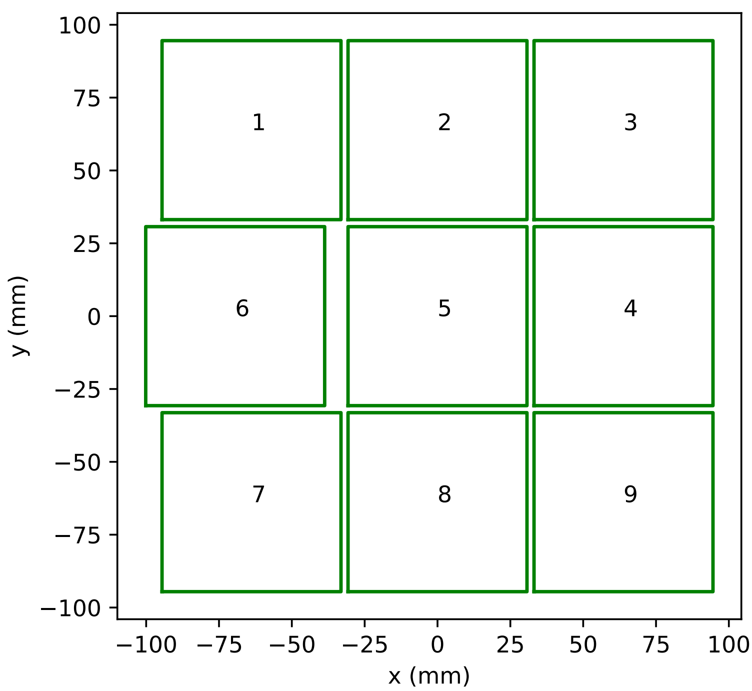

Figure 1: Numbering convention of the detectors in the MICADO focal plane.

```
## MICADO H4RG-15 FPA
   id   x_cen  y_cen    xhw    yhw  x_len  y_len  pixsize  angle    gain
#          mm     mm     mm     mm    pix    pix       mm    deg  e-/ADU
    1  -63.84  63.84  30.72  30.72   4096   4096    0.015    0.0     1.0
    2    0.00  63.84  30.72  30.72   4096   4096    0.015    0.0     1.0
    3   63.84  63.84  30.72  30.72   4096   4096    0.015    0.0     1.0
    4  +63.84   0.00  30.72  30.72   4096   4096    0.015    0.0     1.0
    5    0.00   0.00  30.72  30.72   4096   4096    0.015    0.0     1.0
    6  -79.50   0.00  30.72  30.72   4096   4096    0.015    0.0     1.0
    7  -63.84 -63.84  30.72  30.72   4096   4096    0.015    0.0     1.0
    8    0.00 -63.84  30.72  30.72   4096   4096    0.015    0.0     1.0
    9   63.84 -63.84  30.72  30.72   4096   4096    0.015    0.0     1.0
```

Listing 1: Focal plane array definition file for MICADO with nine HAWAII4RG chips with 15 µm pixels

constructs a world coordinate system (`CTYPEiA = 'LINEAR'`) from the FPA definition file and uses this internally to perform the transformations. The WCS is written to the SpecCADO output files with alternative axis descriptor `A` and name `WCSNAMEA = 'PIX2FP'`.

6.3 Spectral cube coordinates

The spectral source is described as a spectroscopic cube with spatial coordinates ξ and η along and across the slit, respectively (Fig. 2), and wavelength λ. Both ξ and η are measured in arcsec, based on the fixed imaging scale of MICADO. For the short slit of MICADO, ξ runs from -1.5 arcsec to 1.5 arcsec, whereas for the long slit, ξ runs from -1.5 arcsec to 13.5 arcsec. The slit length to be used can be specified by the keyword `SPEC_SLIT_LENGTH`, the slit width by `SPEC_SLIT_WIDTH`.

The mapping between the centre line of the slit $(\xi, \eta = 0, \lambda)$ and focal-plane coordinates (x, y) for a given spectral order s is characterised by order-definition files provided by Frank Grupp. These files were produced using ray-tracing with Zemax and list matching cube and focal-plane coordinates for a number of points. SpecCADO models the transformations as fourth-order polynomials fitted to the order definition files:

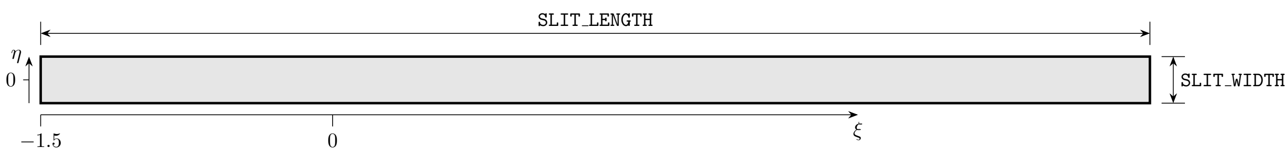

Figure 2: Slit coordinates ξ, η along and across the slit, respectively, along with the slit dimensions `SPEC_SLIT_LENGTH` and `SPEC_SLIT_WIDTH`.

$$x(\xi,\lambda) = \sum_{i,j=0}^{4} A_{s,ij}\,\xi^i\lambda^j \tag{1}$$

$$y(\xi,\lambda) = \sum_{i,j=0}^{4} B_{s,ij}\,\xi^i\lambda^j \tag{2}$$

$$\xi(x,y) = \sum_{i,j=0}^{4} C_{s,ij}\,x^i y^j \tag{3}$$

$$\lambda(x,y) = \sum_{i,j=0}^{4} D_{s,ij}\,x^i y^j \tag{4}$$

The across-slit coordinate η is integrated out before the spectral cube is mapped to the focal plane as illustrated in Fig. 3. A monochromatic slice $\lambda = \lambda_0$ of the spectral cube is mapped to a rectangle in the focal plane. Given a unique mapping of focal-plane coordinates to wavelength (the wavelength calibration) this means that, say, the top of the slit will be at an offset wavelength, $\lambda_0 + \Delta\lambda$. If the slit width (given by the SimCADO keyword `SPEC_SLIT_WIDTH`) is b and the plate scale is p (in mas μm^{-1}), then the wavelength shift is

$$\Delta\lambda = \frac{\partial\lambda}{\partial y}\Delta y = \frac{b}{2}\frac{1}{p}\left[\frac{\partial y}{\partial\lambda}(\xi,\lambda)\right]^{-1} \tag{5}$$

Each plane (η = const.) of the spectral cube is shifted by the appropriate amount in the wavelength direction. The resulting sheared cube is then summed in the η direction to give a two-dimensional spectrum with the

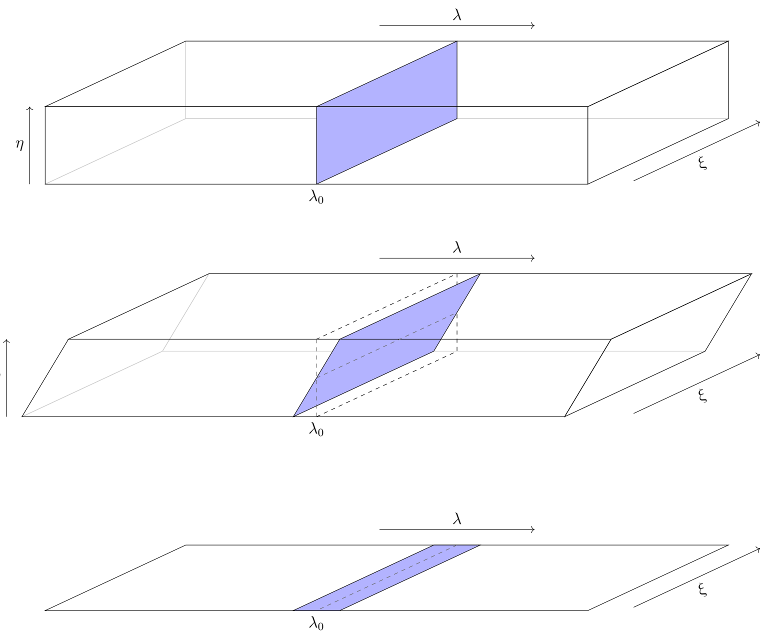

Figure 3: The spectral cube (ξ,η,λ), top, is sheared in the wavelength direction to take into account the slit profile, middle, and then summed in the η direction to form the rectified two-dimensional spectrum (ξ,λ), bottom, which is subsequently mapped onto the focal plane.

spectral lines automatically broadened by the slit profile. The transformations (1) and (2) are then applied to map the 2D spectrum into the focal plane and onto the detectors.

7 Spectral layout

The panels in Fig. 4 show the spectral traces for the 3 arcsec slit and the three order sorting filters, *IzJ*, *J* and *HK*. Fig. 5 identifies the orders as a function of wavelength.

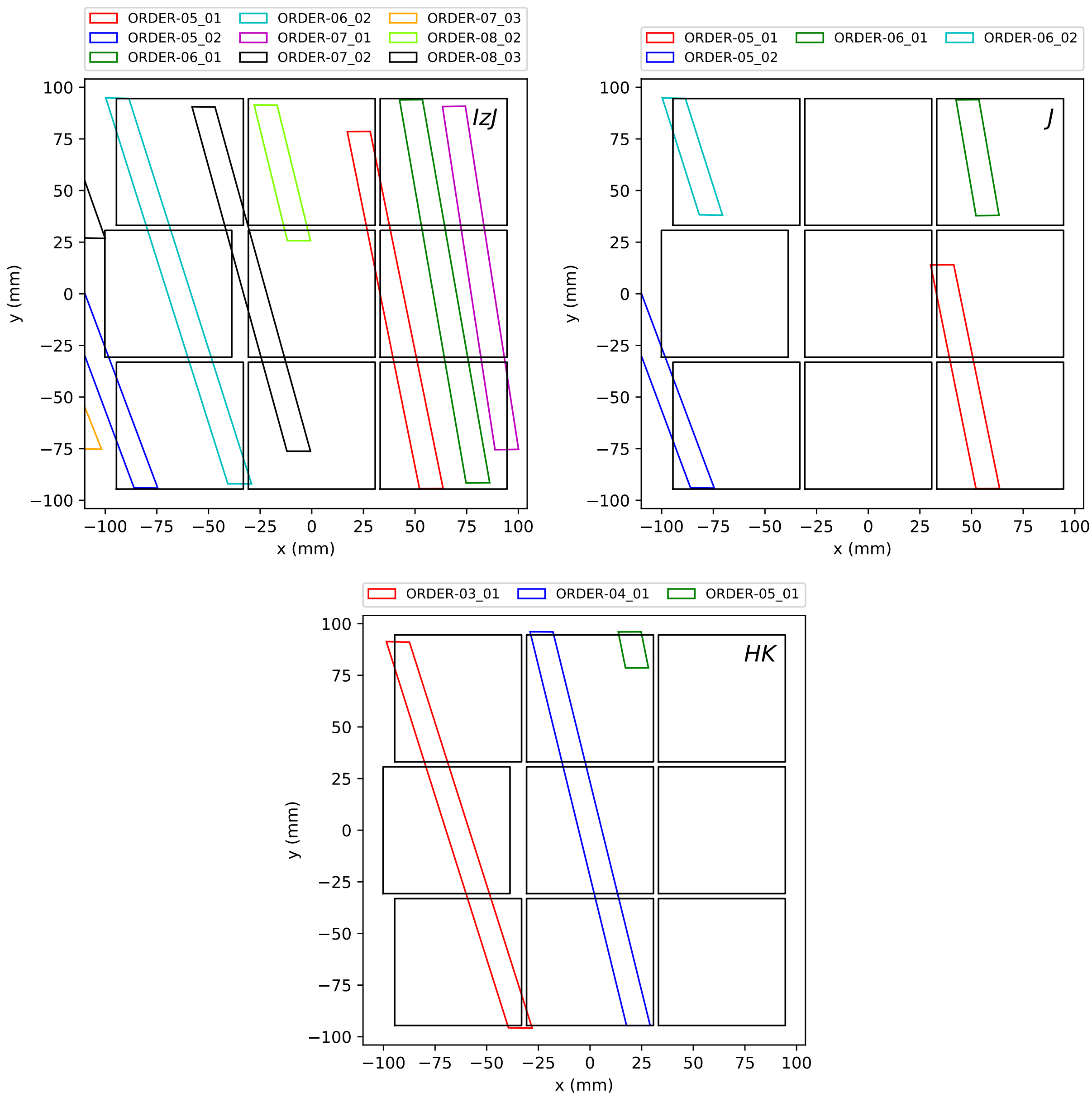

Figure 4: Spectral layout based on the PDR design. The slit has a width of 3 arcsec on the sky.

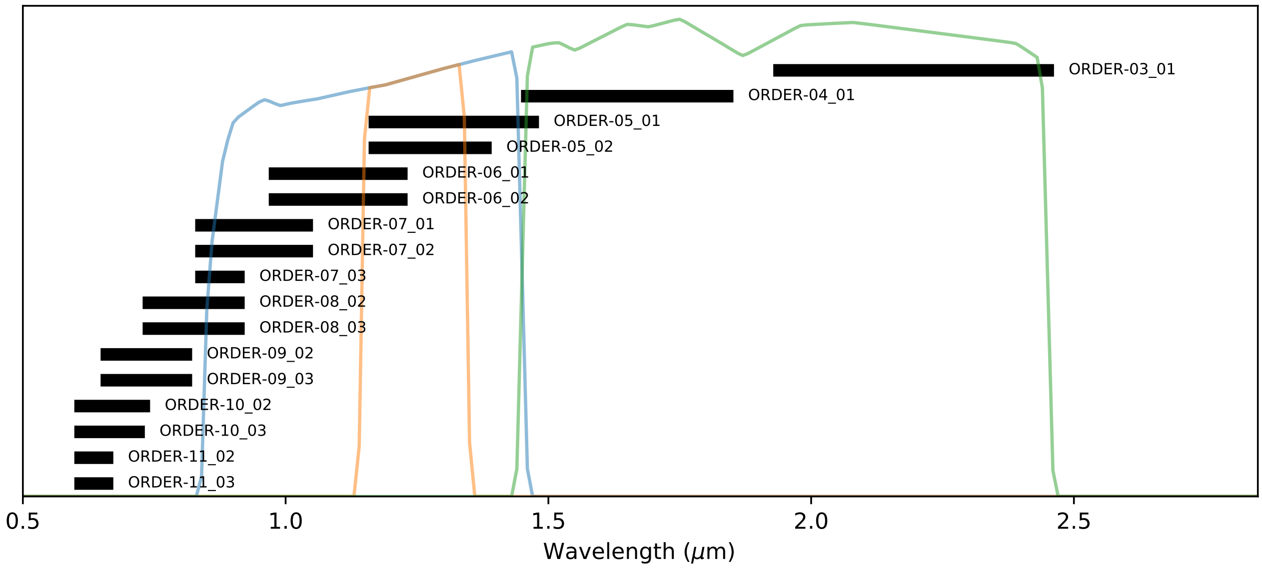

Figure 5: Wavelength ranges covered by the spectral orders. The blue, orange and green curves give the relative system transmissivities for the IzJ, J and HK order sorting filters, respectively.

References

[1] Leschinski, K. and Czoske, O., *SimCADO: the Data Simulator for MICADO* (2018). ELT-ICD-MCD-56306-0050, v1.0. 5

[2] Greisen, E. W., Calabretta, M. R., Valdes, F. G., and Allen, S. L., "Representations of spectral coordinates in fits," *Astronomy & Astrophysics* **446**, 747–771 (2006). 12