MIRISim Spreadsheet and Quick examples

Christophe COSSOU, on behalf of the MIRISim team

March 20, 2018
v1.8

Contents

1 Introduction

This document is here to list and detail all the options and parameters available in MIRISim. Once you know how to run a basic simulation, this guide will help you build more complex simulations exploiting all the possibilities of MIRISim.

For knowledge on how to run MIRISim, description of the simulator and all the input files, please refer to the official documentation: (http://miri.ster.kuleuven.be/pub/Internal/Software/MIRISim_Public/MIRISim.pdf)

2 How to define config objects?

All the following parameters are optional. Omitting them will result in a default value attributed. Omitting all of them will create the official default file by MIRISim.

Generic import needed:

```
# JWST imports
from mirisim import MiriSimulation

from mirisim.skysim import *
from mirisim.config_parser import *
```

You may not want to define all files by hand. You can define default for each of them, or read one or two from an .ini file but define the third in Python.

Default values for all three files:

```
scene_config = SceneConfig.from_default()
sim_config = SimConfig.from_default() # IMA
simulator_config = SimulatorConfig.from_default()
```

3 Simulation Config

Default simulation config:

```
sim_config = SimConfig.ima_default() # IMA
sim_config = SimConfig.mrs_default() # MRS
sim_config = SimConfig.lrs_default() # LRS
```

```
sim_config = SimConfig.makeSim(
             name="Default Simulation", rel_obsdate=0.0,
             scene="scene.ini", POP='IMA', ConfigPath='IMA_FULL',
             Dither=True, StartInd=1, NDither=2,
             DitherPat="ima_recommended_dither.dat",
             filter="F1130W", readDetect='FULL', ima_mode='FAST',
             ima_exposures=5, ima_integrations=4, ima_frames=10,
             disperser='SHORT', detector='SW', mrs_mode='SLOW',
             mrs_exposures=5, mrs_integrations=4, mrs_frames=10)
```

All the parameters here are mandatory. I advise you to use their dummy name for clarity purpose.

Possible values:

- **POP**: IMA, MRS
- **Mode**: FAST, SLOW
- **ConfigPath**: IMA_FULL, IMA_SUB256, IMA_SUB128, IMA_SUB64, IMA_BRIGHTSKY, IMA_CORO1065, IMA_CORO1140, IMA_CORO1550, IMA_COROLYOT, LRS_SLIT, LRS_SLITLESS, MRS_1SHORT, MRS_1MEDIUM, MRS_1LONG, MRS_2SHORT, MRS_2MEDIUM, MRS_2LONG, MRS_3SHORT, MRS_3MEDIUM, MRS_3LONG, MRS_4SHORT, MRS_4MEDIUM, MRS_4LONG
- IMA/LRS **filters**: F560W, F770W, F1000W, F1130W, F1280W, F1500W, F1800W, F2100W, F2550W, F1065C, F1140C, F1550C, F2300C, P750L, P750L_SLITLESSPRISM

- **ReadDetect**: FULL, BRIGHTSKY, SUB128, SUB256, SUB64, SLITLESSPRISM, MASK1065, MASK1140, MASK1550, MASKLYOT
- MRS **disperser**: SHORT, MEDIUM, LONG
- MRS **detector**: SW, LW, BOTH

4 Simulator Config (advanced)

Default simulation config:

```
simulator_config = SimulatorConfig.from_default()
```

```
simulator_config = SimulatorConfig.makeSimulator(
    take_webbPsf=False,
    add_extended=False,
    include_refpix=True,
    include_poisson=True,
    include_readnoise=True,
    include_badpix=True,
    include_dark=True,
    include_flat=True,
    include_gain=True,
    include_nonlinearity=True,
    include_drifts=True,
    include_latency=True,
    cosmic_ray_mode='SOLAR_MIN')
```

Options for `cosmic_ray_mode`:

- `SOLAR_MIN`
- `SOLAR_MAX`
- `SOLAR_FLARE`

`add_extended` will force the LRS to treat extended sources outside of the slit. They can have an impact on the slit itself but are not included by default, essentially because of the time it takes to model them for possibly no impact at all.

Optional advanced parameters include the possibility to tell the exact CDP version you want for certain CDPs. Note that each require a string of the following format "00.00.00":

- `ima_distortion_version`
- `lrs_distortion_version`
- `mrs_distortion_version`
- `ima_pixarea_version`
- `ima_pce_version`
- `ima_pixelflat_version`
- `ima_skyflat_version`
- `ima_psf_version`
- `ima_psf_F2550WR_version`
- `ima_psfoof_version`
- `mrs_psf_version`
- `mrs_pce_version`
- `mrs_fringe_version`

5 Scene Config

Default scene:

```
scene_config = SceneConfig.from_default()
```

```
background = Background(level='low', gradient=5., pa=15.0, centreFOV=(0., 0.))

SED1 = BBSed(Temp=300., wref=10., flux=1.6e4)

Point1 = Point(Cen=(0.,0.), vel=0.0)
Point1.set_SED(SED1)


targets = [Point1]

scene_config = SceneConfig.makeScene(loglevel=0, background=background, targets=
    targets)
```

See Section 6 for more details on what classes are available to define targets, SEDs,...

By default, there is no background and no targets, allowing you to define targets without background or vice-versa if you want to.

6 Build your own sky

Creating your own SceneConfig in Python require a bit of skills since there's a lot of different classes available for SED, target types or velocity maps. This section aim to present all of them in an extensive way.

A quick and easy way to import everything you need for a Scene is to import via:

```
from mirisim.skysim import *
```

If you prefer, you can:

```
import mirisim.skysim as skysim
```

but you will also get other unwanted classes in the process.

6.1 Targets

All needed imports

```
from mirisim.skysim import *
```

6.1.1 Background

```
background = Background(level='low', gradient=5., pa=15.0, centreFOV=(0., 0.))
```

Parameter	Comment	Unit
level	Background with the G-component of the model included 'high' or missing 'low'	None
gradient	% over 1 arcmin (JWST component only)	%
pa	position angle of gradient (background increasing towards PA)	degree
centreFOV	centre of FOV	?

6.1.2 Point

Example of Point source target:

```
Point1 = Point(Cen=(0.,0.), vel=0.0)
Point1.set_SED(SED1)
```

where `SED1` can be whatever SED class you can think of (and defined in Section 6.2).

Parameter	Comment	Unit
Cen	Where to place the target (ra,dec) offsets from centreFOV)	(arcsec, arsec)
vel	velocity	km/s

6.1.3 Galaxy

Example of Galaxy target:

```
Galaxy1 = Galaxy(Cen=(0., 0.), n=2., re=0.5, q=0.5, pa=35.)
Galaxy1.set_velomap(velomap1)
Galaxy1.set_SED(SED1)
Galaxy1.set_LOSVD(losvd1)
```

where `velomap1` (See Section 6.3), `SED1` (See Section 6.2), `losvd1` (See Section 6.4), are generic object that must obviously be defined before.

Parameter	Comment	Unit
Cen	Where to place the target (ra,dec) offsets from centreFOV)	(arcsec, arsec)
n	Sersic index of the Galaxy (n=1: exponential profile, n=4: de Vaoucouleurs)	None
re	Effective radius (half of flux is within r <= re)	arcsec
q	Axial ratio	None
pa	position angle	degree

6.1.4 SkyCube (input .fits file)

This correspond to a SkyScene defined as a Fits file. More details on how to generate one .fits file compatible with MIRISim (i.e what metadata you need) in [§ A on page 11].

```
target = Skycube('TWhya_input_to_simulation.fits')
```

The first 2 dimensions are in arcsec, the 3rd dimension is wavelength in microns. The flux is in micro Jansky.

Parameter	Comment	Unit
cubefits	filename of the cube .fits	None
center	If yes, the input file is centered anyway, and astrometry ignored	"yes" or "no" (default yes)
method	Interpolation method selected (see below)	0, 1 or 2 (default 1)
conserve	how the flux is conserved (see below)	full or wave (default: wave)

center: (default: yes)

- `"no"`: By definition the center of the detector subarray we selected is set to be on $(ra, dec) = (0, 0)$. Then CRVALi refer to the position of the reference pixel of the input .fits in (ra,dec) (By definition, $(ra, dec) = (0, 0)$ is set to be the center of the selected subarray). On the other hand, this same reference is defined by CRPIXi in the input pixel coordinates (Note that $(\text{CRPIX1}, \text{CRPIX2}) = (1, 1)$ correspond to the center of the bottom left pixel of the input .fits file).
- `"yes"`: The center of the input .fits file will be at the center of the subarray detector

method: specifying grid interpolation method (default: 1)

- 0: interpolation with no flux conservation rebinning (fast, inaccurate)
- 1: cumulative trapezoid rule rebinning with interpolation over entire 3D cube (default)
- 2: cumulative trapezoid rule rebinning with interpolation per each column (scales with the number of pixels - and can be very slow)

conserve: (default: wave)

- `'wave'`: interpolate wavelength grid only to conserve flux (you do not conserve flux over the spatial grid)
- `'full'`: full flux conserving interpolation over the full grid of ra, dec, wav

.fits file metadata must contain:

- `CRVAL1`, `CRVAL2`, `CRVAL3`: position of the reference pixel on the sky
- `CRPIX1`, `CRPIX2`, `CRPIX3`: position of the reference pixel in the skycube coordinate system
- `CDELT1`, `CDELT2`, `CDELT3`: Size of a pixel
- `CTYPE1`, `CTYPE2`, `CTYPE3`: Unit for each dimension

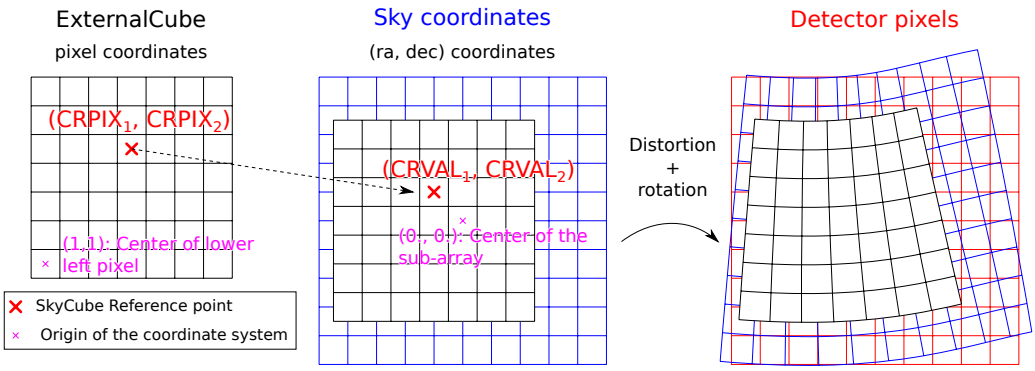

Figure 1 – How a SkyCube is positioned on the detector. Note that a pixel in the Skycube will never exactly be a pixel on the detector, even with a size of 0.11"

6.1.5 ExpDisk

$$I = I_0 e^{-r/h} \tag{6.1}$$

```
target = ExpDisk(h=2., Cen=(0.,0.))
```

Parameter	Comment	Unit
h	scale length of exponential disk	arcsec
Cen	Where to place the target (ra,dec) offsets from centreFOV)	(arcsec, arsec)
q	Axial ratio	None
pa	position angle	degree
rtunk	inner truncated radius	arcsec
c	Boxiness (0: ellipse, >0: disky ellipse, <0: boxy ellipse)	None

6.1.6 SersicDisk

$$I = Iee^{-k*((r/re)**(1./n)-1} \tag{6.2}$$

```
target = SersicDisk(n=1, re=1.5, Cen=(0.,0.))
target.set_velomap(velomap1)
target.set_SED(SED1)
target.set_LOSVD(losvd1)
```

Parameter	Comment	Unit
Cen	Where to place the target (ra,dec) offsets from centreFOV)	(arcsec, arsec)
q	Axial ratio	None
pa	position angle	degree
n	Sersic index (n=1: exponential profile, n=4: de Vaoucouleurs)	None
re	effective radius (half of flux is within r <= re)	arcsec
rtunk	inner truncated radius	arcsec
c	Boxiness (0: ellipse, >0: disky ellipse, <0: boxy ellipse)	None

6.1.7 Others

Other classes not yet implemented:

- PlanetDisk

- Planet
- Star
- SersicAnnulus

6.2 SED

All needed imports

```
from mirisim.skysim import *
```

6.2.1 Black Body

In scene.ini use:

```
Type = BB
```

```
SED = BBSed(Temp=300., wref=10., flux=3.e5)
```

Parameter	Comment	Unit
flux	Reference flux	μJy
wref	Reference wavelength of the reference flux	μm
Temp	Temperature	K

6.2.2 Lines

Sed which is just a set of Gaussian lines.

In scene.ini use:

```
Type = Lines
```

```
wavels = [5.7, 6.6, 8.7, 10.1, 11.3, 13.4, 15.5, 20.6, 24.0]
fluxes = [1.5E4, 1.5E4, 1.5E4, 1.5E4, 1.5E4, 1.5E4, 1.5E4, 1.5E4, 1.5E4]
fwhms = [1.E-3, 1.E-3, 1.E-3, 1.E-3, 1.E-3, 1.E-3, 1.E-3, 1.E-3, 1.E-3]
SED = LinesSed(wavels=wavels, fluxes=fluxes, fwhms=fwhms)
```

Parameter	Comment	Unit
wavels	wavelengths of lines	μm
fluxes	integrated fluxes of lines (areas under lines)	μJy $*$ μm
fwhms	fwhms of lines. Can be zero for delta profiles	μm

6.2.3 PySynphot

See the official website for Pysynphot (https://pysynphot.readthedocs.io/en/latest/).

The best way to find the available families and sedname for Pysynphot is to look into you home, at the following location `$HOME/MIRICLE/cdbs/grid/`.

In scene.ini use:

```
Type = pysynphot
```

The available families are:

- agn
- bc95
- bkmodels
- bpgs
- bz77
- ck04models
- etc_models

- galactic
- gunnstryker
- jacobi
- k93models
- kc96
- phoenix
- pickles

The `sedname` however, are far too numerous to list them here for each and every families. However, the `sedname` is just the name of a .fits file without the .fits extension. For a given `family`, the .fits files related to `sednames` are just in the corresponding folder.

In the following example:

```
SED = PYSPSed(family='bc95', sedname='bc95_b_10E6', wref=10., flux=6.e3)
```

This just means that the following file exists:
`$HOME/MIRICLE/cdbs/grid/bc95/bc95_b_10E6.fits`

Parameter	Comment	Unit
flux	Reference flux	μJy
wref	Reference wavelength of the reference flux	μm

6.2.4 External Sed (input ASCII table)

In scene.ini use:

```
Type = External
sedfile = manual_sed.dat
```

```
SED1 = ExternalSed(sedfile='manual_sed.dat')
```

where `manual_sed.dat` is an ASCII file with two columns, the first being the wavelength in microns, and the second being the flux in micro Jansky.

6.2.5 Power Law Sed

The flux is defined by:

$$F(\lambda) = \text{flux} * \left(\frac{\lambda}{\text{wref}}\right)^{\text{alpha}} \tag{6.3}$$

In scene.ini use:

```
Type = PL
```

```
SED1 = PLSed(alpha=2.0, wref=10., flux=6.e3, b=100.)
```

Parameter	Comment	Unit
flux	Reference flux	μJy
wref	Reference wavelength of the reference flux	μm
alpha	power-law index	None
b	additional flux constant	μJy

6.2.6 Others

Other classes not yet implemented:

- StarSed

6.3 Velomap

All needed imports:

```
from mirisim.skysim import *
```

6.3.1 Flat Disk

In scene.ini use:

```
Type = FlatDisk
```

Flat Rotation Curve Disk:

```
velomap1 = FlatDisk(Cen=(0., 0.), vrot=200., pa=35., q=0.5, c=0.)
```

Parameter	Comment	Unit
vrot	constant velocity value	km/s
Cen	(RA, DEC) relative coordinate offset of the centre	arcsec
q	Axis ratio	None
pa	positional angle	degrees
c	boxiness of the ellipse	None

6.3.2 Kepler Disk

Keplerian Disk with the following velocity distribution:

$$v(r) = v0 \left(\frac{r0}{r}\right)^{0.5} \tag{6.4}$$

In scene.ini use:

```
Type = KeplerDisk
```

```
velomap1 = KeplerDisk(Cen=(0., 0.), v0=200., r0=1., pa=35., q=0.5, c=0.)
```

Parameter	Comment	Unit
v0	velocity value at r0	km/s
r0	Reference radial distance	arcsec
Cen	(RA, DEC) relative coordinate offset of the centre	arcsec
q	Axis ratio	None
pa	positional angle	degrees
c	boxiness of the ellipse	None

6.4 Line Of Sight Velocity Distribution

All needed imports:

```
from mirisim.skysim import *
```

6.4.1 Losvd

The equation for line of sight velocity distribution is:

$$\text{losvd} = \frac{1}{\sqrt{(2\pi)}\sigma} * e^{-y^2/2} * (1 + h3 * H3(y) + h4 * H4(y)) \tag{6.5}$$

```
losvd1 = Losvd(sigma=200., h3=0., h4=0.)
```

Parameter	Comment	Unit
sigma	velocity dispersion	km/s
h3	Gaussian Hermite polynomial coefficient h3	None
h4	Gaussian Hermite polynomial coefficient h4	None

7 How to run a simulation within Python

At last, now that you know how to define each config file (SimConfig, SceneConfig and SimulatorConfig), you can launch a simulation. Of course, this example show the case of one simple simulation but you can run this in a loop and change one or more parameters in the process:

```
import os
import shutil
from mirisim import MiriSimulation
from mirisim.skysim import *
from mirisim.config_parser import * # SimConfig, SimulatorConfig, SceneConfig

# We define the Simulation parameters for MIRISIM
sim_config = SimConfig.from_default()
scene_config = SceneConfig.from_default()
simulator_config = SimulatorConfig.from_default()

# Define the simulation
ms = MiriSimulation(sim_config=sim_config, scene_config=scene_config,
    simulator_config=simulator_config, loglevel='ERROR')

# Run the simulation
ms.run()

# Change simulation folder name
new_dir = "My_simulation_name"

# Delete if the folder already exists
if os.path.isdir(new_dir):
    shutil.rmtree(new_dir)

os.rename(ms.path_out, new_dir)
```

A Generate an input .fits file for SkyCube

This code shows the minimal example to create an input .fits file. Of course, populate the **data** array with the desired data.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

"""
Make a minimal SkyCube that works
"""

import os
from astropy.io import fits
import numpy as np
import astropy.units as u

naxis = (10, 10, 15)

# Index of the reference pixel
crpix = (1, 1, 1)  # Starts at 1

# Value of the reference pixel
# Axis 1 & 2 are in ra,dec ; Axis 3 is in micron
crval = (0., 0., 5.6)

# Size of a pixel in each dimension
cdelt = (0.11, 0.11, 0.1)

data = np.zeros(naxis)

hdu = fits.PrimaryHDU(data)

hdu.header['CRVAL1'] = crval[0]
hdu.header['CRPIX1'] = crpix[0]
hdu.header['CDELT1'] = cdelt[0]
hdu.header['CUNIT1'] = u.arcsec.to_string()

```

```
34 hdu.header['CRVAL2'] = crval[1]
35 hdu.header['CRPIX2'] = crpix[1]
36 hdu.header['CDELT2'] = cdelt[1]
37 hdu.header['CUNIT2'] = u.arcsec.to_string()
38
39 hdu.header['CRVAL3'] = crval[2]
40 hdu.header['CRPIX3'] = crpix[2]
41 hdu.header['CDELT3'] = cdelt[2]
42 hdu.header['CUNIT3'] = u.micron.to_string()
43
44 hdu.header['UNITS'] = u.microjansky.to_string()
45
46 outFileName = 'skycube.fits'
47 if os.path.isfile(outFileName):
48     os.remove(outFileName)
49 hdu.writeto(outFileName)
```