```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Thu Nov 24 11:19:34 2022

@author: dhuga
"""

# Standard Library Imports
from glob import glob
import os
import shutil
import urllib

# Third Party Imports
import pysiaf
#import mirage
from astropy.io import ascii as asc
from astropy.io import fits
from matplotlib import cm
from matplotlib.colors import LogNorm
import matplotlib.pyplot as plt

# Local Imports (from nircam_simulator package)
from mirage import imaging_simulator
from mirage.catalogs import create_catalog
from mirage.utils.utils import ensure_dir_exists
from mirage.yaml import yaml_generator

os.environ["MIRAGE_DATA"]="/Users/dhuga/Desktop/jwst-ulx-jet-prop/
sims/mirage_data"
os.environ["CRDS_PATH"] = "CRDS/crds_cache"
os.environ["CRDS_SERVER_URL"] = "https://jwst-crds.stsci.edu"

# Where to save MIRaGe output
#out_dir = os.path.join(input_dir, 'output')
out_dir = 'output'

xml_file = 'dummy-jet-prop.xml'
pointing_file = 'dummy-jet-prop.pointing'

catalog_out = create_catalog.for_proposal(xml_file, pointing_file,
                                          point_source=True,
                                          extragalactic=True,

catalog_splitting_threshold=0.12,
                                          out_dir=out_dir,
                                          save_catalogs=True)
ptsrc_cat, gal_cat, ptsrc_names, gal_names, pmap, gmap = catalog_out
cat_dict = {'ptsrc': ptsrc_names[0],
            'gal': gal_names[0]}

# Plot all queried sources
for catalog_filename in ptsrc_names:
    target_catalog = asc.read(catalog_filename)
    plt.scatter(target_catalog['x_or_RA'], target_catalog['y_or_Dec'],
s=1, alpha=.7, label='Point Source Catalog')
for catalog_filename in gal_names:
    target_catalog = asc.read(catalog_filename)
    plt.scatter(target_catalog['x_or_RA'], target_catalog['y_or_Dec'],
s=1, alpha=.7, label='Galactic Catalog')
plt.xlabel('Right Ascension [degrees]')
plt.ylabel('Declination [degrees]')
plt.legend()
plt.show()

background = 'medium'
roll_angle = None
dates = '2022-5-25'
cr = None
minimum_signal = 0.08
minimum_signal_units = 'MJy/sr'
ghosts = True
convolve_ghosts = False
datatype = 'linear'

yaml_dir = os.path.join(out_dir, 'yamls')
yam = yaml_generator.SimInput(xml_file, pointing_file, cat_dict,
verbose=True,
                              output_dir=yaml_dir,
                              simdata_output_dir=out_dir,
                              cosmic_rays=cr, background=background,
roll_angle=roll_angle,
                              dates=dates,
dateobs_for_background=False,
                              reffile_defaults='crds_full_name',
                              segmap_flux_limit=minimum_signal,
segmap_flux_limit_units=minimum_signal_units,
                              add_ghosts=ghosts,
convolve_ghosts_with_psf=convolve_ghosts)

yam.use_linearized_darks = True
yam.create_inputs()

# Examine apertures and V2/V3 references for each array/subarray
#nc_siaf = pysiaf.Siaf('NIRCam')
#nc_full = nc_siaf['NRCA1_FULL']

#plt.figure(figsize=(15,10))
#for apername in sorted(nc_siaf.apernames):
```

```
#    a = apername
#    if ('_FULL' in a) and ('OSS' not in a) and ('MASK' not in a) and
(a[-1] != 'P'):
#        nc_siaf[a].plot(frame='tel', label=True, fill_color='white')
#plt.gca().invert_xaxis()

# Compare V2/V3 of targets (from .pointing file)
#all_pointings = set([(v2, v3, filename) for v2, v3, filename in
zip(yam.info['v2'],
#
yam.info['v3'],
#
yam.info['yamlfile'])])
#
#print('Example files for each pointing:')
#print('--------------------------------')
#plotted_points = []
#for i_point, (v2, v3, filename) in enumerate(all_pointings):
#    if (v2, v3) not in plotted_points:
#        plotted_points.append((v2, v3))
#        plt.scatter(v2, v3, marker='*', s=500,
#                    label='Pointing {}/{}'.format(i_point + 1,
len(all_pointings)))
#        print('{}. {}'.format(i_point + 1, filename))
#
#plt.legend()
#plt.show()

# Select one YAML to estimate where the sources will be
#test_yaml_filename = '/Users/dhuga/Desktop/jwst-ulx-jet-prop/sims/
output/yamls/jw00042001001_01101_00001_nrcb1.yaml'
#test_yaml = os.path.join(yaml_dir, test_yaml_filename)
#assert os.path.isfile(test_yaml)
#print(test_yaml)

# Run the image simulator using the input defined in test_yaml
#img_sim = imaging_simulator.ImgSim()
#img_sim.paramfile =test_yaml_filename
#img_sim.create()

#select all yaml files
yaml_files = glob('*.yaml')
for yfile in yaml_files:
    img_sim = imaging_simulator.ImgSim()
    img_sim.paramfile = yfile
    img_sim.create()


# Plot the seed image, dark image, and final exposure simulation
fig, [ax1, ax2, ax3] = plt.subplots(1, 3, figsize=(20, 7))
```

```
plt.tight_layout()

# Define scale limits and colormap
clim=(0.001, 0.1)
cmap = cm.get_cmap('viridis')
cmap.set_bad(cmap(0))

# Plot seed image
fitsplot = ax1.imshow(img_sim.seedimage, clim=clim, cmap=cmap)
ax1.set_title('Seed Image', size=24)
ax1.invert_xaxis()
ax1.invert_yaxis()

# Plot dark current
ax2.imshow(img_sim.linDark.data[0,-1,:,:] -
img_sim.linDark.data[0,0,:,:], clim=clim, cmap=cmap)
ax2.set_title('Dark Current', size=24)
ax2.invert_xaxis()
ax2.invert_yaxis()

# Plot final exposure
#file_root = os.path.basename(test_yaml_filename).split('.yaml')[0]
file_root = os.path.basename(yfile).split('.yaml')[0]
linear_output = os.path.join(out_dir, '{}
_linear.fits'.format(file_root))
with fits.open(linear_output) as h:
    lindata = h[1].data
    header = h[0].header
exptime = header['EFFINTTM']
diffdata = (lindata[0,-1,:,:] - lindata[0,0,:,:]) / exptime

ax3.imshow(diffdata, clim=clim, cmap=cmap)
ax3.set_title('Final Exposure Simulation', size=24)
ax3.invert_xaxis()
ax3.invert_yaxis()

# Define the colorbar
cbar_ax = fig.add_axes([1, 0.09, 0.03, 0.87])
cbar = plt.colorbar(fitsplot, cbar_ax)
cbar.set_label('Count Rate', rotation=270, labelpad=30, size=24)
```