PHY688 - Modelling Project

Author: Paras Sharma (ID#: 115415559)

(Original notebook by Dr. Anowar J. Shajib available here)

Modelling a Real Hubble Space Telescope Image (DESIJ0201-2739)

In [39]:
```
# make sure lenstronomy is installed, otherwise install the latest pip version
try:
    import lenstronomy
except ModuleNotFoundError:
    !pip install lenstronomy

from lenstronomy.Workflow.fitting_sequence import FittingSequence
from lenstronomy.Plots.model_plot import ModelPlot
```

In [40]:
```
# import of standard python libraries
import numpy as np
import corner
import matplotlib.pyplot as plt
import h5py

%matplotlib inline
```

Load imaging data

The data and the PSF needs to be provided to `lenstronomy` using the dictionaries `kwargs_data` and `kwargs_psf`.

In `kwargs_data`, we also need to proivde information on the noise level. Either the pixel-wise noise map can be provided using the `noise_map` keyword, or simply the `exposure_time` and `background_rms` can be provided for `lenstronomy` to create the noise map by itself.

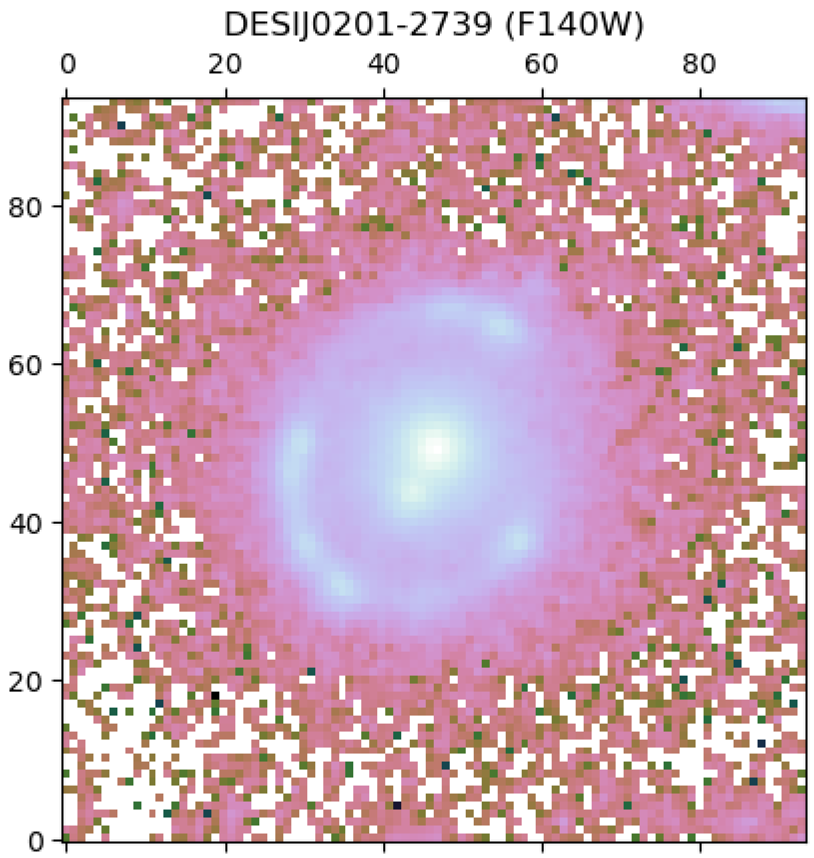
The keywords `ra_at_xy_0` and `dec_at_xy_0` are the RA and Declination in arcsecond units at the (0, 0) pixel. The keyword `transform_pix2angle` is the transformation matrix from pixel number coordinates to (RA, Decl.). These keywords are used to convert pixel coordinates to RA and Decl. coordinates. If you want to convert one 2D coordinate system to another, you will need to the offset between the zeropoints of two coordinate systems and transformation matrix that specifies the scaling and rotation of the axes. So, the keywords `ra_at_xy_0` and `dec_at_xy_0` specify the zeropoint offsets, and `transform_pix2angle` is the tansformation matrix specifying scaling and rotation.

In [41]:
```
with h5py.File("DESIJ0201-2739_F140W.h5", "r") as f:
    kwargs_data = {}
    for key in f:
        kwargs_data[key] = f[key][()]

with h5py.File("psf_F140W.h5", "r") as f:
    kwargs_psf = {}
    for key in f:
        kwargs_psf[key] = f[key][()]

kwargs_data['noise_map'] = None
kwargs_psf["psf_type"] = "PIXEL"
```

In [42]:
```
plt.figure()
plt.matshow(np.log10(kwargs_data['image_data']), origin="lower", cmap="cubehelix");
plt.title("DESIJ0201-2739 (F140W)")
```

```
/var/folders/4r/jq9w4fy92y9_bqzvdfk2d5h40000gn/T/ipykernel_1432/652338674.py:2: RuntimeWarning: invalid value encountered in log10
  plt.matshow(np.log10(kwargs_data['image_data']), origin="lower", cmap="cubehelix");
```

Out[42]:
```
Text(0.5, 1.0, 'DESIJ0201-2739 (F140W)')
<Figure size 640x480 with 0 Axes>
```

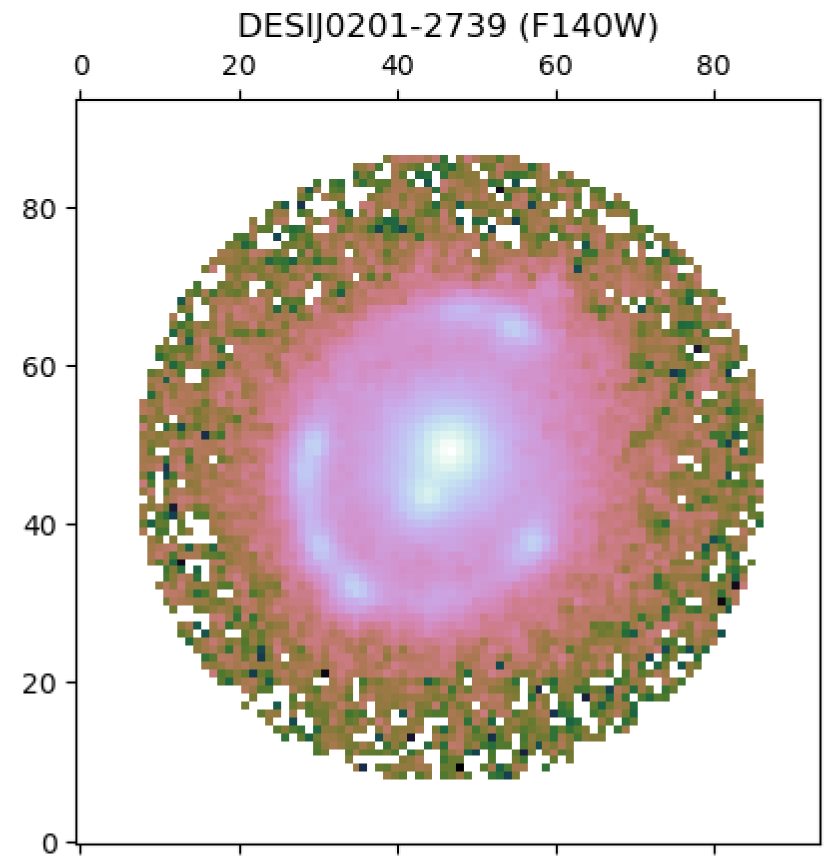


Masking the data

In [43]:
```
circle_mask = np.zeros_like(kwargs_data['image_data'])
y, x = np.mgrid[:circle_mask.shape[0], :circle_mask.shape[1]]
r = np.sqrt((x-kwargs_data['image_data'].shape[1]/2)**2 + (y-kwargs_data['image_data'].shape[0]/2)**2)
circle_mask[r < 40] = 1
kwargs_data['image_data'] = kwargs_data['image_data'] * circle_mask

plt.figure()
plt.matshow(np.log10(kwargs_data['image_data']), origin="lower", cmap="cubehelix");
plt.title("DESIJ0201-2739 (F140W)")
# plt.scatter([47], [47], color="red", marker="x")
```

```
/var/folders/4r/jq9w4fy92y9_bqzvdfk2d5h40000gn/T/ipykernel_1432/3674790897.py:8: RuntimeWarning: divide by zero encountered in log10
  plt.matshow(np.log10(kwargs_data['image_data']), origin="lower", cmap="cubehelix");
/var/folders/4r/jq9w4fy92y9_bqzvdfk2d5h40000gn/T/ipykernel_1432/3674790897.py:8: RuntimeWarning: invalid value encountered in log10
  plt.matshow(np.log10(kwargs_data['image_data']), origin="lower", cmap="cubehelix");
```

Out[43]:
```
Text(0.5, 1.0, 'DESIJ0201-2739 (F140W)')
<Figure size 640x480 with 0 Axes>
```



Building a lens model

Here, we build a lens model. The lens model can be thought of three components: the lens galaxy's mass model, the lens galaxy's light model, and the source galaxy's light model. We have to give a list of profiles for each component as shown in the next codecell.

The `'EPL'` lens mass profile stands for Elliptical Power Law. The form of this convergence profile is given by:

$$\kappa(x, y) = \frac{3-\gamma}{2}\left[\frac{\theta_{\rm E}}{\sqrt{qx^2 + y^2/q}}\right]^{\gamma-1}.$$

The position angle ϕ adjusts the orientation of the mass profile's major axis. The ellipticity parameters q and ϕ can be reformulated as

$$e_1 = \frac{1-q}{1+q}\cos 2\phi,$$

$$e_2 = \frac{1-q}{1+q}\sin 2\phi.$$

`lenstronomy` uses e_1 and e_2 instead of q and ϕ, because e_1 and e_2 are easier to handle in numerical optimization, for example, in MCMC. You can check here for more details on parameter definitions and conventions used in `lenstronomy`.

Both the lens galaxy's and the source galaxy's light profiles are modeled with Sersic function, which is given by:

$$I(x, y) = I_{\rm e} \exp\left[-b_n\left\{\left(\frac{\sqrt{qx^2+y^2/q}}{R_{\rm Sersic}}\right)^{1/n_{\rm Sersic}} - 1\right\}\right].$$

In [44]:
```
lens_model_list = ["EPL", "SHEAR"]
source_model_list = ["SERSIC_ELLIPSE", "SHAPELETS"]
lens_light_model_list = ["SERSIC_ELLIPSE"]
```

In this cell below, we have to specify the initial values, upper and lower limits for the model parameters specific to each of the model components specified in the above lists. The `kwargs_*_sigma` dictionaries/lists are used to set the initial size of the search area within particle swarm optimization (PSO) or MCMC.

In [45]:
```
# lens galaxy's mass model
fixed_lens = []
kwargs_lens_init = []
kwargs_lens_sigma = []
kwargs_lower_lens = []
kwargs_upper_lens = []

fixed_lens.append({})
kwargs_lens_init.append(
    {
        "theta_E": 1.0,
        "gamma": 2.0,
        "e1": 0.0,
        "e2": 0.0,
        "center_x": 0.0,
        "center_y": 0.0,
    }
)
kwargs_lens_sigma.append(
    {
        "theta_E": 0.2,
        "gamma": 0.1,
        "e1": 0.05,
        "e2": 0.05,
        "center_x": 0.5,
        "center_y": 0.5,
    }
)
kwargs_lower_lens.append(
    {
        "theta_E": 0.01,
        "gamma": 1.0,
        "e1": -0.5,
        "e2": -0.5,
        "center_x": -10,
        "center_y": -10,
    }
)
kwargs_upper_lens.append(
    {
        "theta_E": 10.0,
        "gamma": 3.0,
        "e1": 0.5,
        "e2": 0.5,
        "center_x": 10,
        "center_y": 10,
    }
)
fixed_lens.append({'ra_0': 0, 'dec_0': 0})
kwargs_lens_init.append({'gamma1': 0., 'gamma2': 0.0})
kwargs_lens_sigma.append({'gamma1': 0.1, 'gamma2': 0.1})
```

```
kwargs_lower_lens.append({'gamma1': -0.3, 'gamma2': -0.3})
kwargs_upper_lens.append({'gamma1': 0.3, 'gamma2': 0.3})


lens_params = [
    kwargs_lens_init,
    kwargs_lens_sigma,
    fixed_lens,
    kwargs_lower_lens,
    kwargs_upper_lens,
]


# lens galaxy's light model
fixed_lens_light = []
kwargs_lens_light_init = []
kwargs_lens_light_sigma = []
kwargs_lower_lens_light = []
kwargs_upper_lens_light = []

fixed_lens_light.append({"n_sersic": 4.0})
kwargs_lens_light_init.append(
    {
        "R_sersic": 0.5,
        "n_sersic": 2,
        "e1": 0,
        "e2": 0,
        "center_x": 0.0,
        "center_y": 0,
        "amp": 16,
    }
)
kwargs_lens_light_sigma.append(
    {
        "n_sersic": 1,
        "R_sersic": 0.3,
        "e1": 0.05,
        "e2": 0.05,
        "center_x": 0.1,
        "center_y": 0.1,
        "amp": 10,
    }
)
kwargs_lower_lens_light.append(
    {
        "e1": -0.5,
        "e2": -0.5,
        "R_sersic": 0.001,
        "n_sersic": 0.5,
        "center_x": -10,
        "center_y": -10,
        "amp": 0,
    }
)
kwargs_upper_lens_light.append(
    {
        "e1": 0.5,
        "e2": 0.5,
        "R_sersic": 10,
        "n_sersic": 5.0,
```

```
        "center_x": 10,
        "center_y": 10,
        "amp": 100,
    }
)

joint_lens_with_light = [[0, 0, ["center_x", "center_y", "e1", "e2"]]]
joint_two_sources = [[0, 1, ["center_x", "center_y"]]]

lens_light_params = [
    kwargs_lens_light_init,
    kwargs_lens_light_sigma,
    fixed_lens_light,
    kwargs_lower_lens_light,
    kwargs_upper_lens_light,
]


# source galaxy's light model
fixed_source = []
kwargs_source_init = []
kwargs_source_sigma = []
kwargs_lower_source = []
kwargs_upper_source = []

fixed_source.append({"n_sersic": 1.0})
kwargs_source_init.append(
    {
        "R_sersic": 0.2,
        "n_sersic": 1,
        "e1": 0,
        "e2": 0,
        "center_x": 0.0,
        "center_y": 0,
        "amp": 16,
    }
)
kwargs_source_sigma.append(
    {
        "n_sersic": 0.5,
        "R_sersic": 0.1,
        "e1": 0.05,
        "e2": 0.05,
        "center_x": 0.2,
        "center_y": 0.2,
        "amp": 10,
    }
)
kwargs_lower_source.append(
    {
        "e1": -0.5,
        "e2": -0.5,
        "R_sersic": 0.001,
        "n_sersic": 0.5,
        "center_x": -10,
        "center_y": -10,
        "amp": 0,
    }
)
kwargs_upper_source.append(
```

```
    {
        "e1": 0.5,
        "e2": 0.5,
        "R_sersic": 10,
        "n_sersic": 5.0,
        "center_x": 10,
        "center_y": 10,
        "amp": 100,
    }
)
fixed_source.append({"n_max": 10})
kwargs_source_init.append(
    {'beta':0.1, 'center_x':0, 'center_y':0})
kwargs_source_sigma.append(
    {'beta':0.1, 'center_x':0.1, 'center_y':0.1})
kwargs_lower_source.append(
    {'beta':0.01, 'center_x':-10, 'center_y':-10})
kwargs_upper_source.append(
    {'beta':0.5, 'center_x':10, 'center_y':10})

source_params = [
    kwargs_source_init,
    kwargs_source_sigma,
    fixed_source,
    kwargs_lower_source,
    kwargs_upper_source,
]


# combining all the above specification in the `kwargs_params` dictionary
kwargs_params = {
    "lens_model": lens_params,
    "source_model": source_params,
    "lens_light_model": lens_light_params,
}

kwargs_constraints = {"joint_lens_with_light": joint_lens_with_light,
                      "joint_source_with_source": joint_two_sources}
```

Numerical settings

No need to change anything here for now. It's also fine to not understand these settings for now.

In [46]:
```
kwargs_likelihood = {"check_bounds": True}

kwargs_numerics = {"supersampling_factor": 1, "supersampling_convolution": False}
```

Combining all the information to be sent to lenstronomy

In [47]:
```
kwargs_model = {
    "lens_model_list": lens_model_list,
    "source_light_model_list": source_model_list,
    "lens_light_model_list": lens_light_model_list,
}

multi_band_list = [[kwargs_data, kwargs_psf, kwargs_numerics]]

kwargs_data_joint = {
```

```
    "multi_band_list": multi_band_list,
    "multi_band_type": "single-band"
}
```

Here the model fiting is done

In [48]:
```
fitting_seq = FittingSequence(
    kwargs_data_joint,
    kwargs_model,
    kwargs_constraints,
    kwargs_likelihood,
    kwargs_params,
)

fitting_kwargs_list = [
    ["PSO", {"sigma_scale": 1.0, "n_particles": 200, "n_iterations": 100}],
    # ['MCMC', {'n_burn': 200, 'n_run': 600, 'n_walkers':
    # 200, 'sigma_scale': .1}]
]

chain_list = fitting_seq.fit_sequence(fitting_kwargs_list)
kwargs_result = fitting_seq.best_fit()
```

```
Computing the PSO ...
100%|██████████| 100/100 [14:24<00:00,  8.64s/it]
Max iteration reached! Stopping.
-1.0540376517771253 reduced X^2 of best position
-4612.995783002589 log likelihood
8753 effective number of data points
[{'theta_E': 0.8645672741042885, 'gamma': 1.401291798530624, 'e1': -0.0661283800221844, 'e2': -0.042873511800708385, 'center_x': 0.06483596889312458, 'center_y': 0.13138646088
227746}, {'gamma1': -0.08470888046754915, 'gamma2': -0.07618361310253743, 'ra_0': 0, 'dec_0': 0}] lens result
[{'amp': 1, 'R_sersic': 0.3411464182718626, 'n_sersic': 1.0, 'e1': 0.26036285381894797, 'e2': -0.044843596462539236, 'center_x': 0.13931968318335172, 'center_y': 0.07802627597
848832}, {'amp': 1, 'n_max': 10, 'beta': 0.07302235061880191, 'center_x': 0.13931968318335172, 'center_y': 0.07802627597848832}] source result
[{'amp': 1, 'R_sersic': 0.46443756685021176, 'n_sersic': 4.0, 'e1': -0.0661283800221844, 'e2': -0.042873511800708385, 'center_x': 0.06483596889312458, 'center_y': 0.1313864608
8227746}] lens light result
[] point source result
[] tracer source result
{} special param result
864.3810060024261 time used for  PSO
===================
```

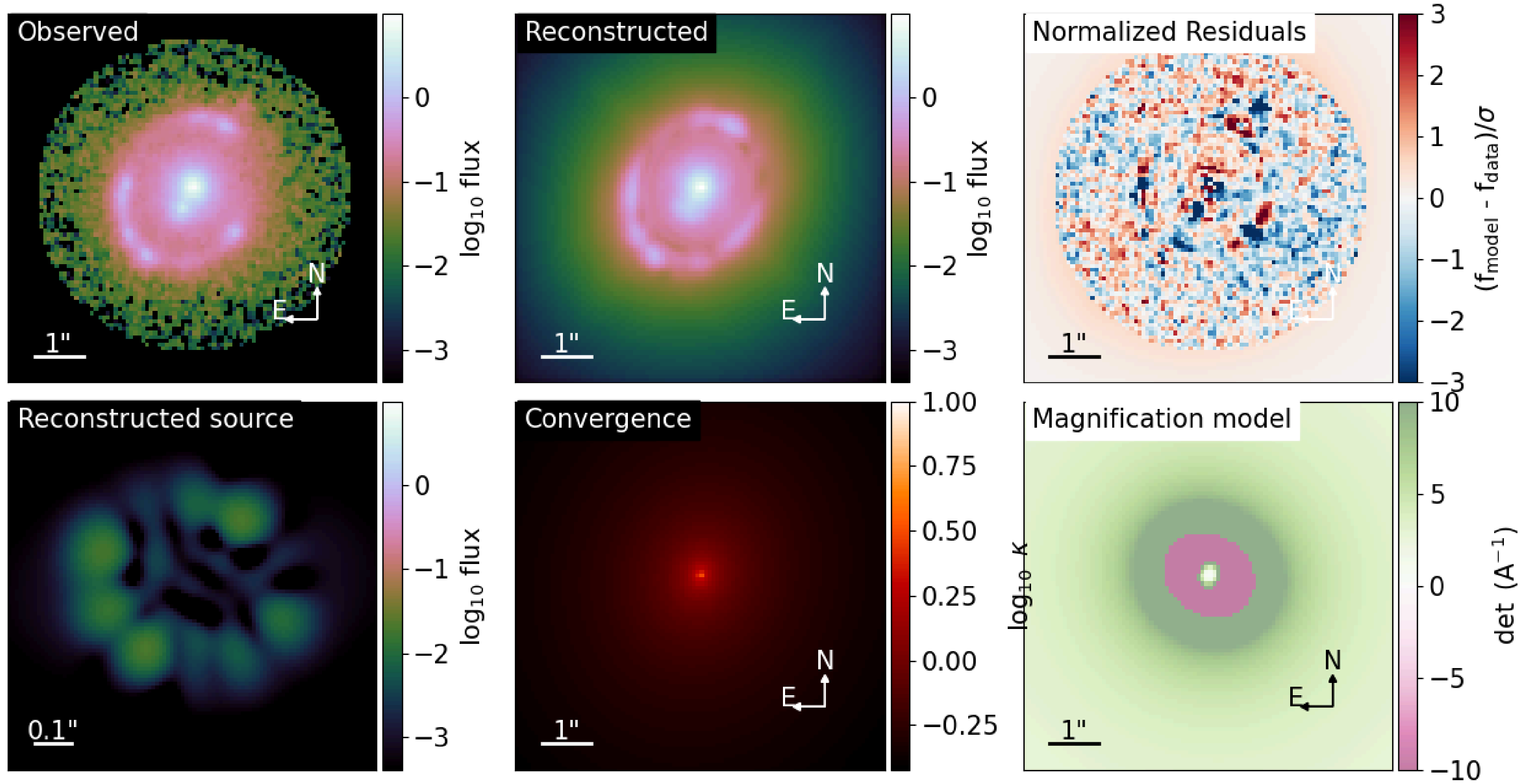
Visualizing the fitted model

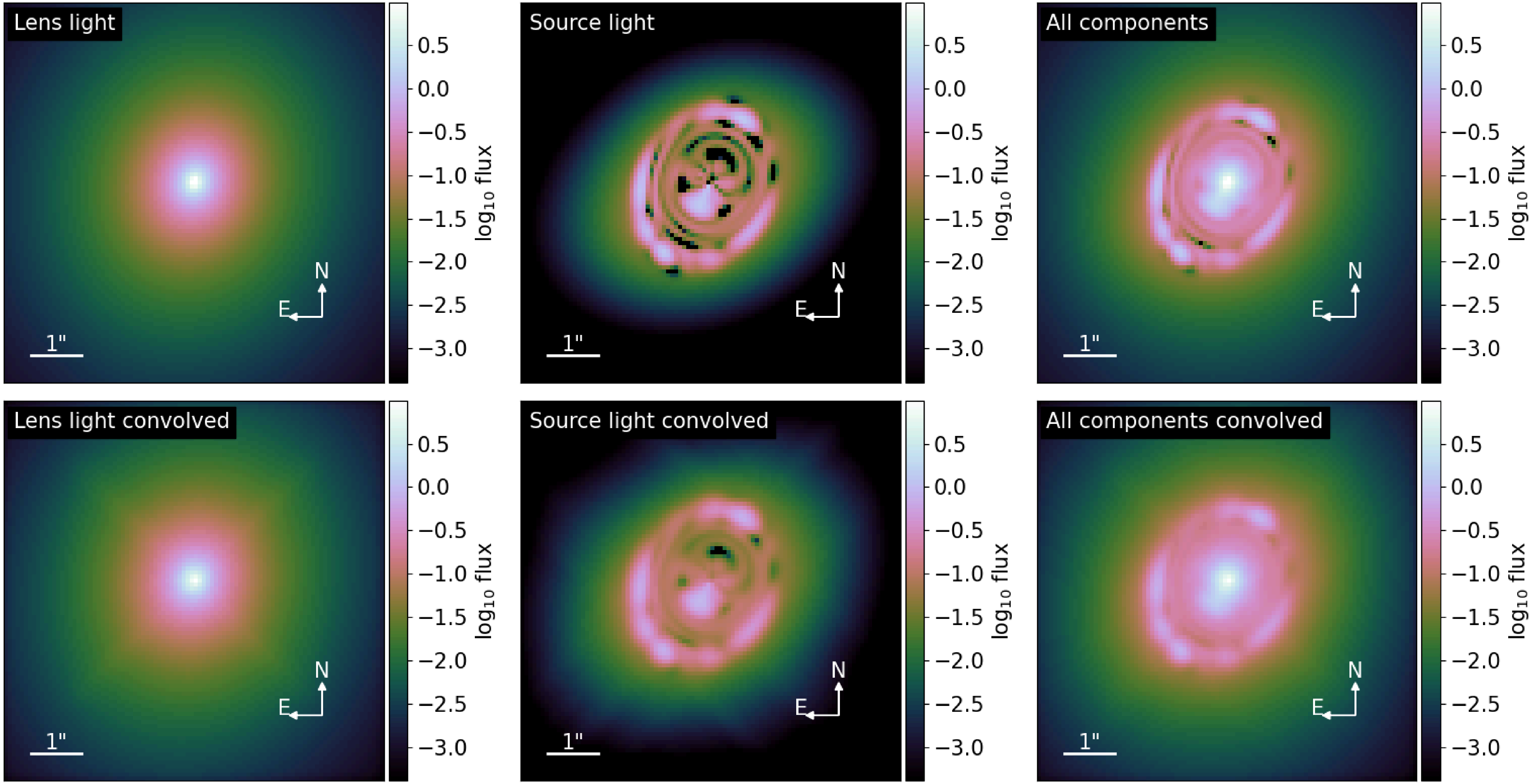
In [49]:
```
model_plot = ModelPlot(
    multi_band_list,
    kwargs_model,
    kwargs_result,
    arrow_size=0.02,
    cmap_string="cubehelix",
)

f, axes = plt.subplots(2, 3, figsize=(16, 8), sharex=False, sharey=False)

model_plot.data_plot(ax=axes[0, 0])
model_plot.model_plot(ax=axes[0, 1])
model_plot.normalized_residual_plot(ax=axes[0, 2], v_min=-3, v_max=3, cmap="RdBu_r")
model_plot.source_plot(ax=axes[1, 0], deltaPix_source=0.01, numPix=100)
```

```python
model_plot.convergence_plot(ax=axes[1, 1], v_max=1, cmap="gist_heat")
model_plot.magnification_plot(ax=axes[1, 2], cmap="PiYG")
f.tight_layout()
f.subplots_adjust(left=None, bottom=None, right=None, top=None, wspace=0.0, hspace=0.05)
plt.show()

f, axes = plt.subplots(2, 3, figsize=(16, 8), sharex=False, sharey=False)

model_plot.decomposition_plot(
    ax=axes[0, 0], text="Lens light", lens_light_add=True, unconvolved=True
)
model_plot.decomposition_plot(
    ax=axes[1, 0], text="Lens light convolved", lens_light_add=True
)
model_plot.decomposition_plot(
    ax=axes[0, 1], text="Source light", source_add=True, unconvolved=True
)
model_plot.decomposition_plot(
    ax=axes[1, 1], text="Source light convolved", source_add=True
)
model_plot.decomposition_plot(
    ax=axes[0, 2],
    text="All components",
    source_add=True,
    lens_light_add=True,
    unconvolved=True,
)
model_plot.decomposition_plot(
    ax=axes[1, 2],
    text="All components convolved",
    source_add=True,
    lens_light_add=True,
    point_source_add=True,
)
f.tight_layout()
f.subplots_adjust(left=None, bottom=None, right=None, top=None, wspace=0.0, hspace=0.05)
plt.show()
print(kwargs_result)
```

```
-1.044136664328336 reduced X^2 of all evaluated imaging data combined (without degrees of freedom subtracted).
reduced chi^2 of data  0 =  1.0441366643283323
```

```
/var/folders/4r/jq9w4fy92y9_bqzvdfk2d5h40000gn/T/ipykernel_1432/565317093.py:17: UserWarning: Tight layout not applied. tight_layout cannot make axes width small enough to acc
ommodate all axes decorations
  f.tight_layout()
```


Observed
Reconstructed
Normalized Residuals
Reconstructed source
Convergence
Magnification model
log10 flux
(f_model - f_data)/σ
log10 κ
det (A^-1)
1"
0.1"
N
E


Lens light
Source light
All components
Lens light convolved
Source light convolved
All components convolved
1"
N
E
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
−3.0
$\log_{10}$ flux

{'kwargs_lens': [{'theta_E': 0.8645672741042885, 'gamma': 1.401291798530624, 'e1': -0.0661283800221844, 'e2': -0.042873511800708385, 'center_x': 0.06483596889312458, 'center_y': 0.13138646088227746}, {'gamma1': -0.08470888046754915, 'gamma2': -0.07618361310253743, 'ra_0': 0, 'dec_0': 0}], 'kwargs_source': [{'amp': 6.495483371242892, 'R_sersic': 0.3411464182718626, 'n_sersic': 1.0, 'e1': 0.26036285381894797, 'e2': -0.044843596462539236, 'center_x': 0.13931968318335172, 'center_y': 0.07802627597848832}, {'amp': array([-29.5331639 , 21.80198105, -5.75901607, 42.08609325,
       -34.76724311, 70.31325862, 36.81348509, -23.09617456,
        12.43648101, -20.89216942, 75.83865583, -24.64334425,
       119.86827566, -90.3331999 , 103.60865057, 70.41187334,
       -25.6865798 , 17.83079243, -15.60604382, -27.49788583,
        25.11124498, 120.59799753, 42.55453333, 79.57210546,
       -72.30739893, 127.39290564, -57.99645646, 52.33338136,
        81.05981987, -10.55162859, 30.0556506 , 24.33389881,
       -19.70514407, -13.52883897, -18.39165764, 29.18927605,
        88.84120237, 68.40418659, 58.91445126, 42.34584363,
        25.61455357, -59.15281959, 34.25523943, 15.26928846,
         2.24708377, 34.82820279,   4.97310703, 26.76317566,
        14.030009 , -18.76333626, -3.36863786,   3.95308615,
        -5.94215697,   5.69319508, -4.32466682,   6.83383544,
        28.0458408 , 73.07235701, 37.17483995, -74.8406443 ,
        -5.66486674, 23.93694966, 31.1411983 , -31.74359987,
        21.69652801, -4.69029153]), 'n_max': 10, 'beta': 0.07302235061880191, 'center_x': 0.13931968318335172, 'center_y': 0.07802627597848832}], 'kwargs_lens_light': [{'amp': 89.61144246369359, 'R_sersic': 0.46443756685021176, 'n_sersic': 4.0, 'e1': -0.0661283800221844, 'e2': -0.042873511800708385, 'center_x': 0.06483596889312458, 'center_y': 0.13138646088227746}], 'kwargs_ps': [], 'kwargs_special': {}, 'kwargs_extinction': [], 'kwargs_tracer_source': []}

visualizing the MCMC chain, if run

In [15]:
```python
if len(chain_list) > 1:
    sampler_type, samples_mcmc, param_mcmc, dist_mcmc = chain_list[1]

    param_class = fitting_seq.param_class

    print("number of non-linear parameters in the MCMC process: ", len(param_mcmc))
    print("parameters in order: ", param_mcmc)
    print("number of evaluations in the MCMC process: ", np.shape(samples_mcmc)[0])
    n_sample = len(samples_mcmc)
    print(n_sample)
    samples_mcmc_cut = samples_mcmc[int(n_sample * 1 / 2.0) :]
    if not samples_mcmc == []:
        n, num_param = np.shape(samples_mcmc_cut)
        plot = corner.corner(
            samples_mcmc_cut[:, :], labels=param_mcmc[:], show_titles=True
        )
```

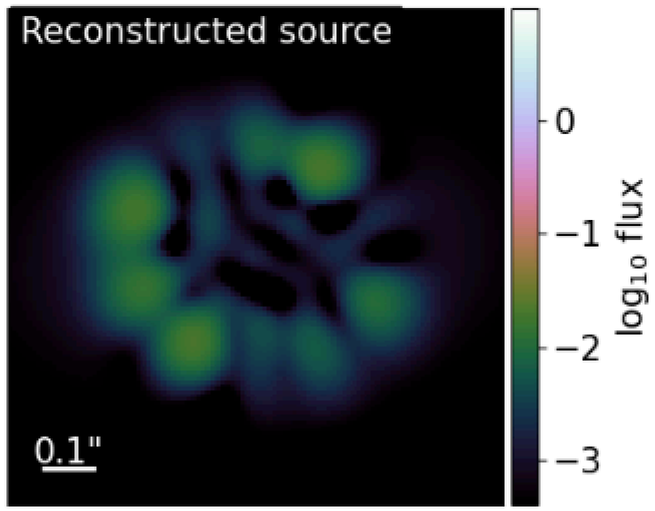
Discussion: My Model compared to Rafee et al. (2024)

Differences, compared to Rafee et al. (J0201-2739)

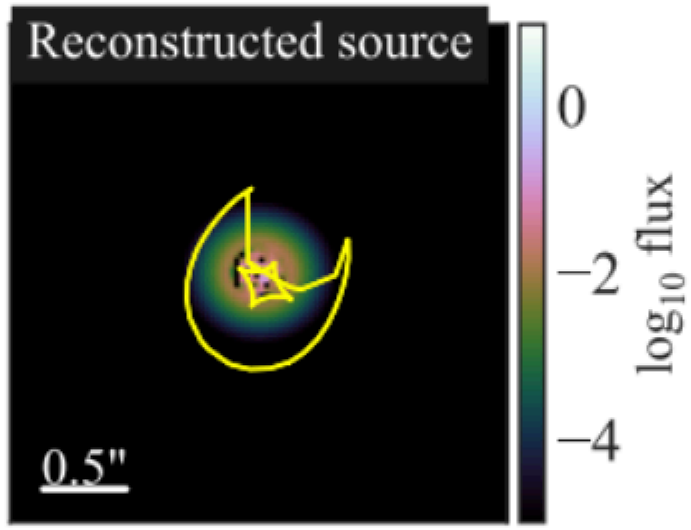
Property	My Model	Rafee et al. 2024
Lens Light	SERSIC ELLIPSE	Double Elliptical Sérsic, Satellite: Elliptical Sérsic
Source Light	SERSIC ELLIPSE + SHAPELETS (n_max = 10)	SERSIC ELLIPSE, SHAPELETS (n_max = 8)
Lens Model	EPL + SHEAR	EPL + SHEAR + SIE
Red. Chi-square	1.044	0.98 (BDLensing NB)

https://github.com/AstroBridge/BDLensing/blob/main/analysis/make_lens_model_figure.ipynb

Differences, compared to Rafee et al. (J0201-2739)

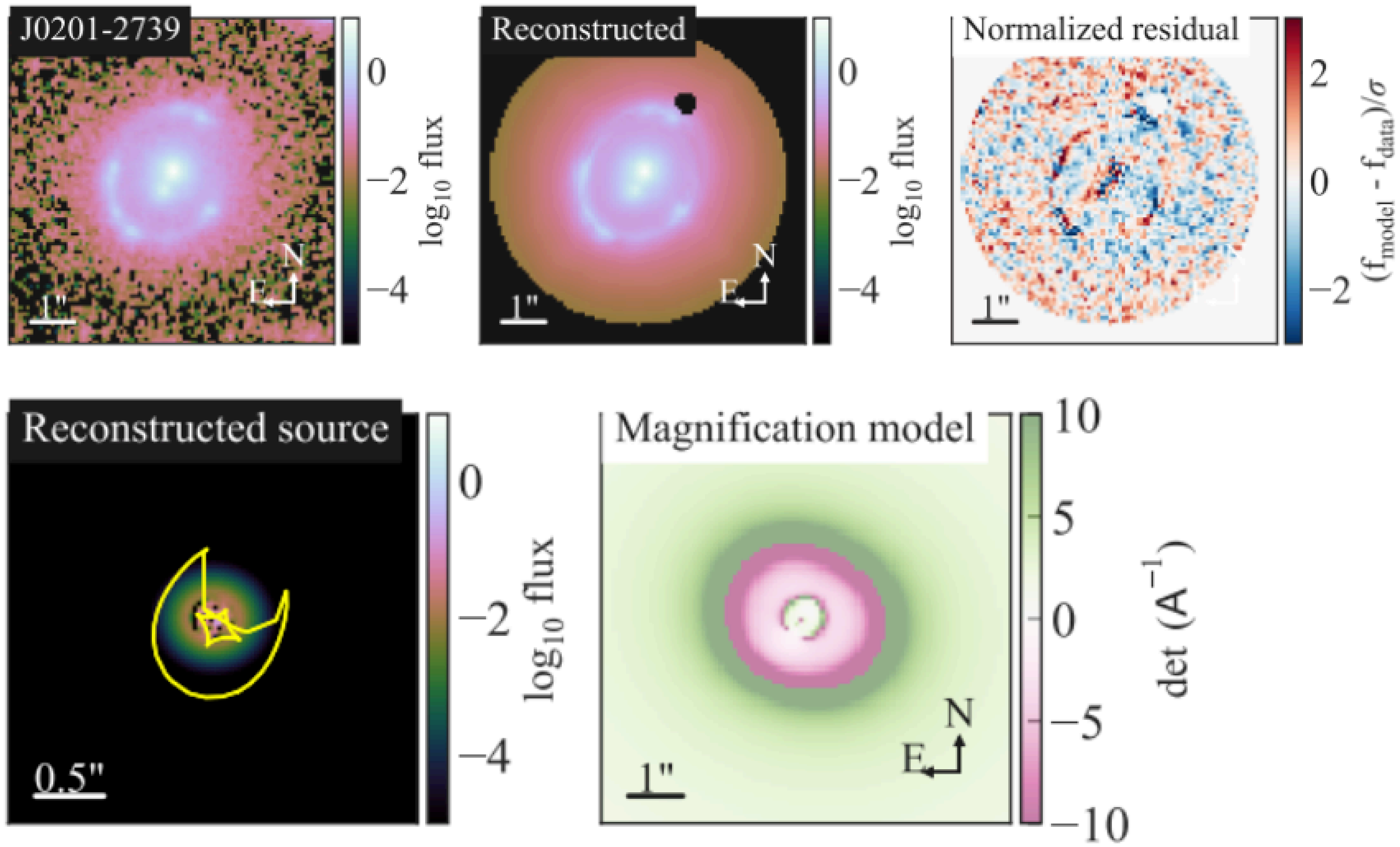


My Model



Rafee et al.

From Rafee et al., Red. Chi-Squared = 0.98



My model certainly uses less number of parameters than Rafee et al. (2024) and so it produces a different source light profile. However the reduced chi-square is almost ~1, which is a good indication that the model is a good fit to the data. But again there can be multiple models that can fit the data well to produce a reduced chi-square of ~1.

My model didn't consider the SIE profile for the lens in addition to the EPL and SHEAR, but Rafee et al. (2024) did. This could be a reason for the difference in the source light profile.