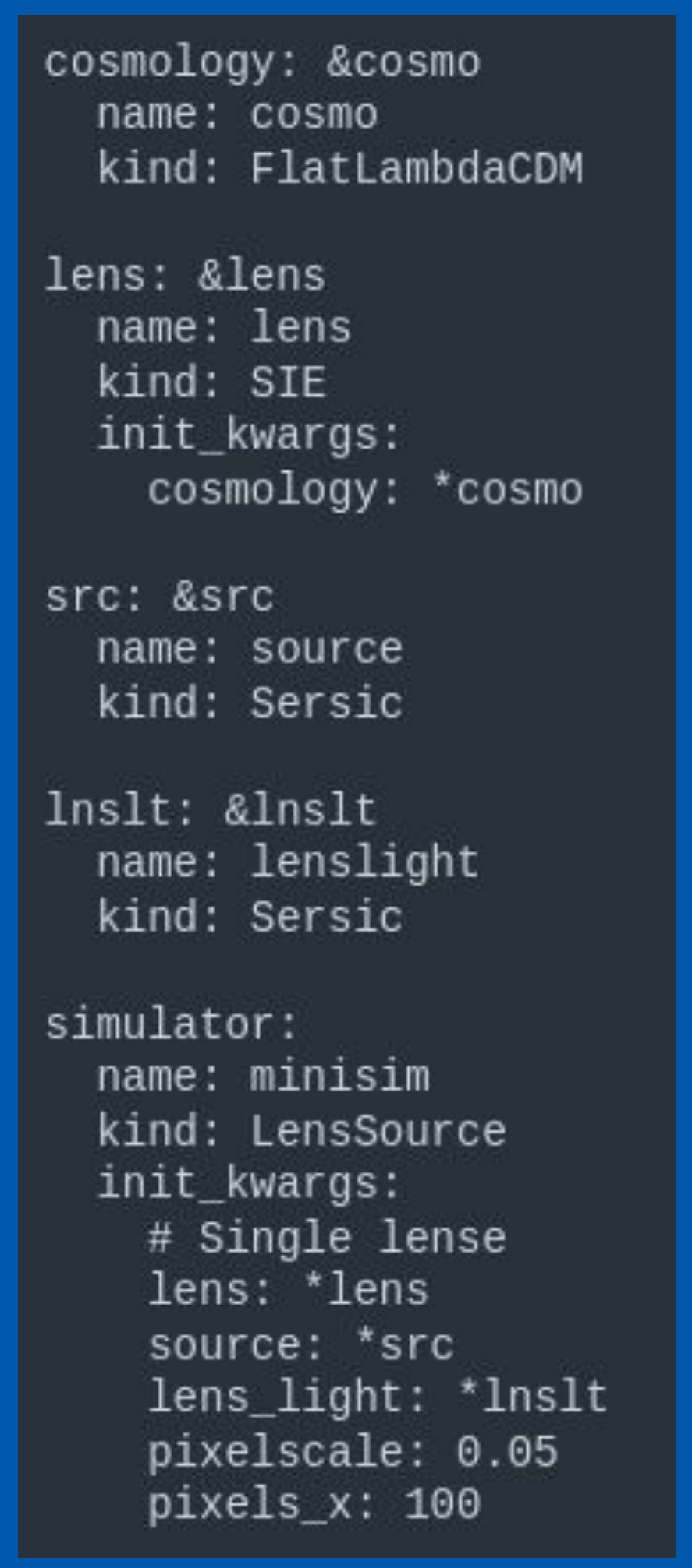
gravitational lensing simulations made **user friendly** with **Caustics'** three interface levels

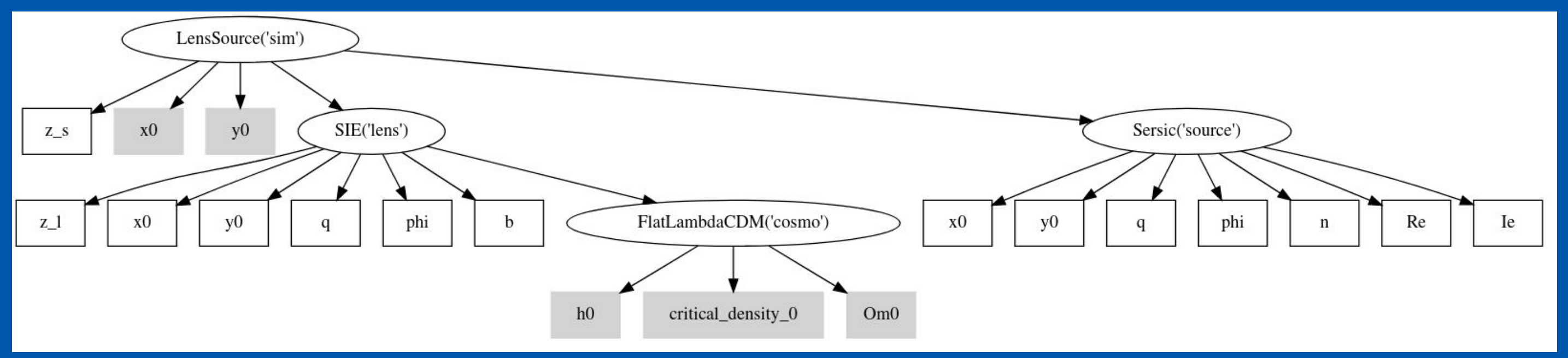
YAML interface



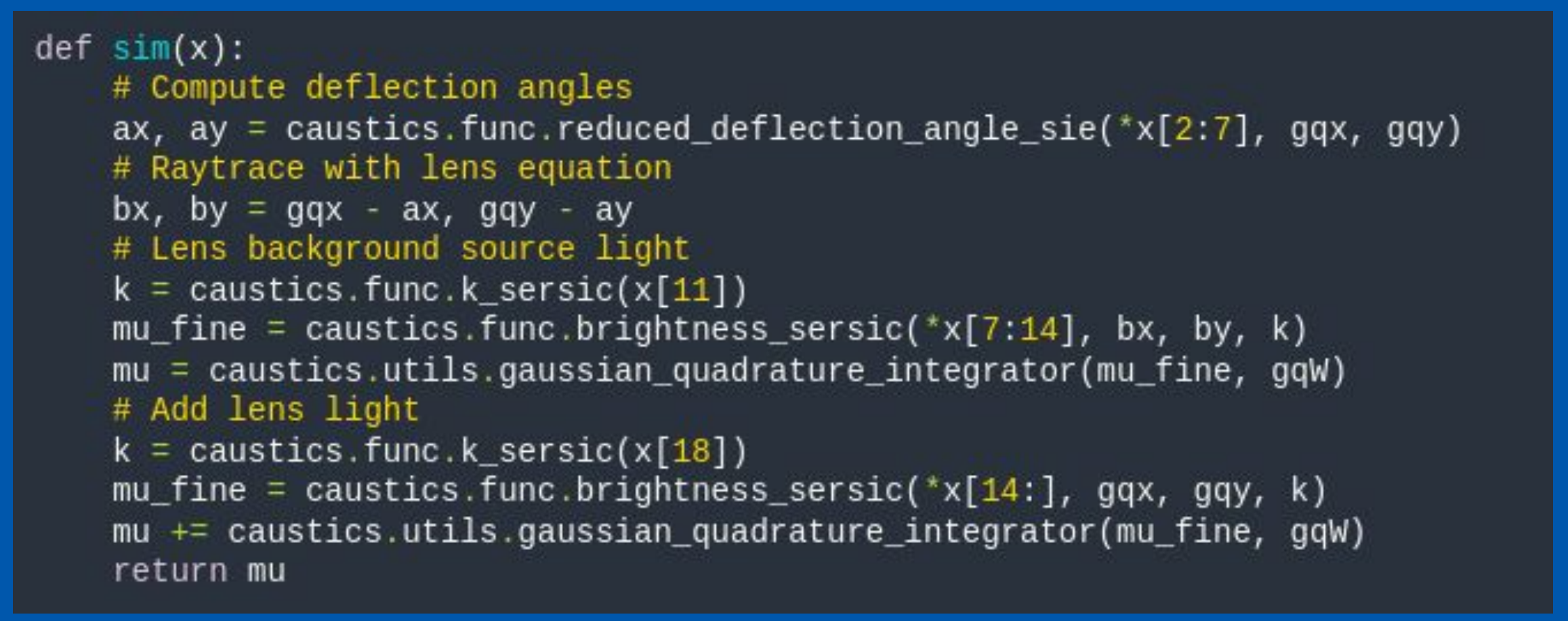
```yaml
cosmology: &cosmo
  name: cosmo
  kind: FlatLambdaCDM

lens: &lens
  name: lens
  kind: SIE
  init_kwargs:
    cosmology: *cosmo

src: &src
  name: source
  kind: Sersic

lnslt: &lnslt
  name: lenslight
  kind: Sersic

simulator:
  name: minisim
  kind: LensSource
  init_kwargs:
    # Single lense
    lens: *lens
    source: *src
    lens_light: *lnslt
    pixelscale: 0.05
    pixels_x: 100
```

The YAML interface allows accessing pre-built simulators in a single line. The OOP interface gives flexibility to build any simulator with the available modules. The functional interface gives total freedom with extensively tested code. This allows all users to approach the code at their skill level, and provides a clear pipeline to increasing knowledge of the package.

Object Oriented interface

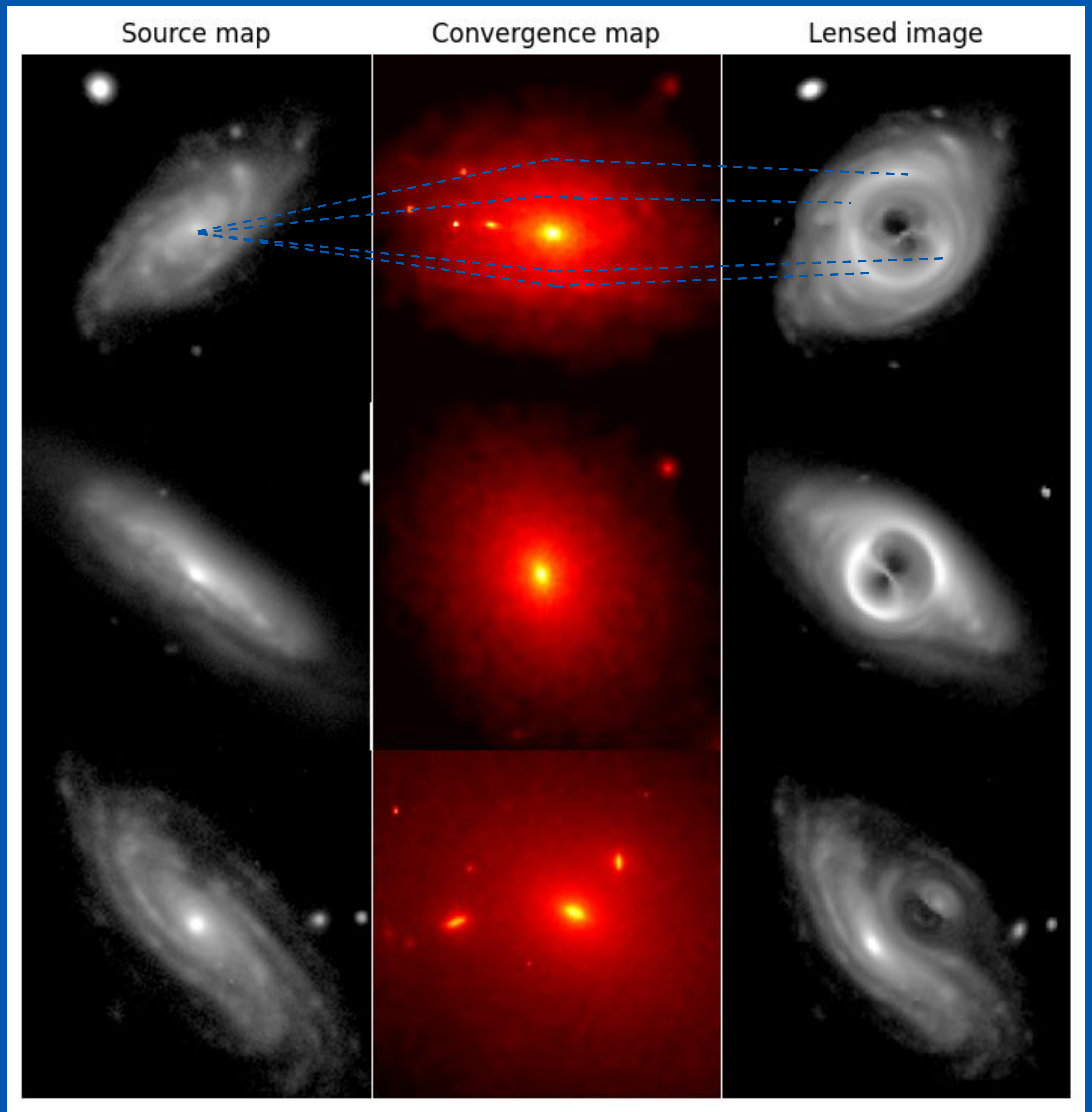


Functional interface

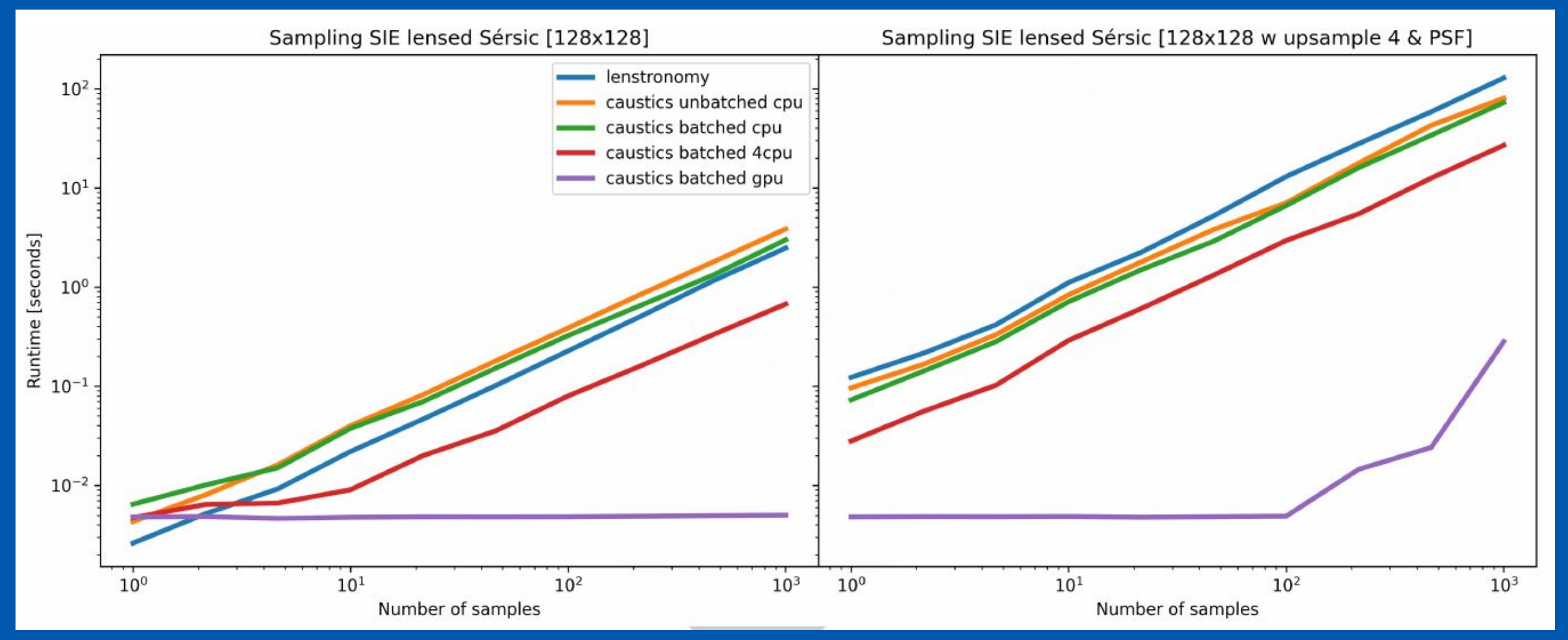


```python
def sim(x):
    # Compute deflection angles
    ax, ay = caustics.func.reduced_deflection_angle_sie(*x[2:7], gqx, gqy)
    # Raytrace with lens equation
    bx, by = gqx - ax, gqy - ay
    # Lens background source light
    k = caustics.func.k_sersic(x[11])
    mu_fine = caustics.func.brightness_sersic(*x[7:14], bx, by, k)
    mu = caustics.utils.gaussian_quadrature_integrator(mu_fine, gqW)
    # Add lens light
    k = caustics.func.k_sersic(x[18])
    mu_fine = caustics.func.brightness_sersic(*x[14:], gqx, gqy, k)
    mu += caustics.utils.gaussian_quadrature_integrator(mu_fine, gqW)
    return mu
```

Lensing is a raytracing problem

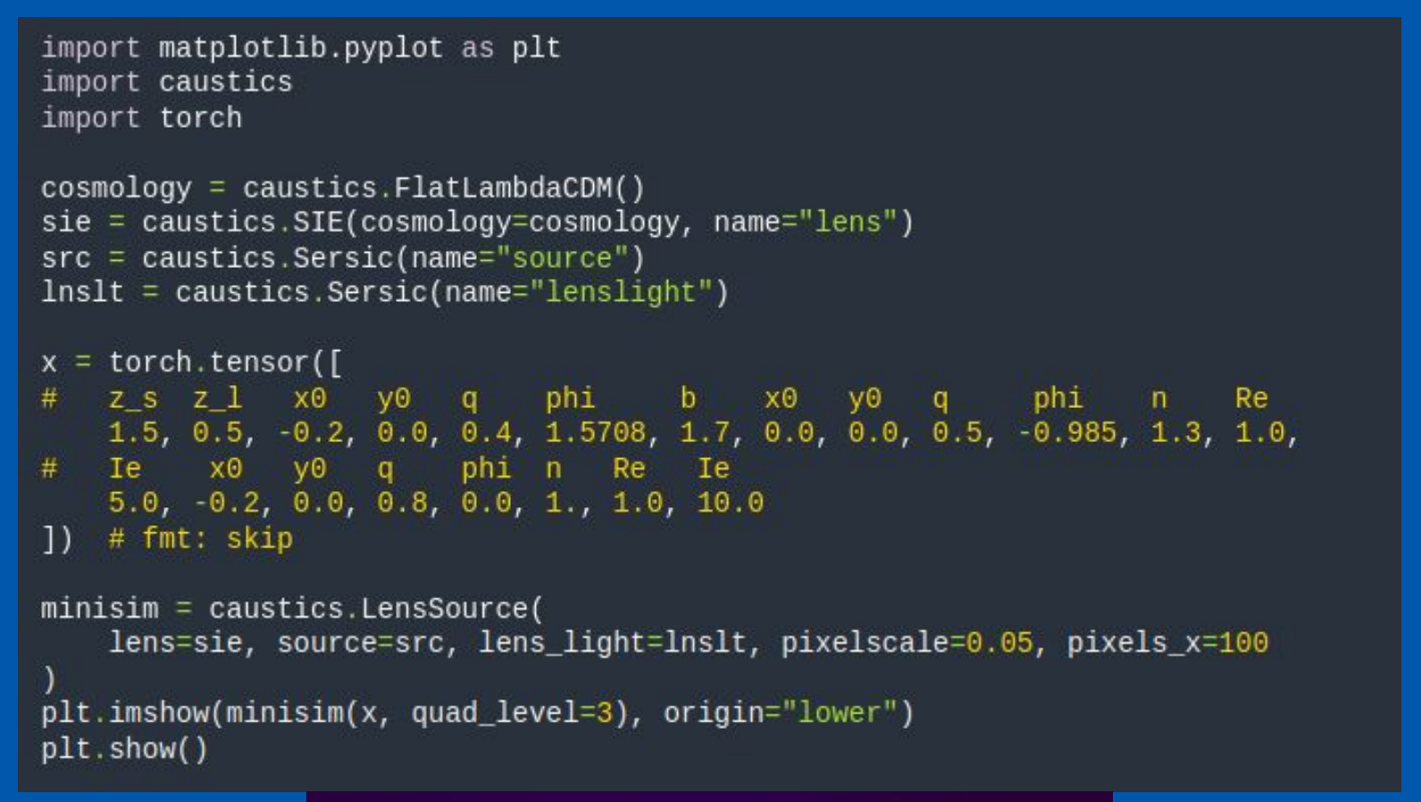


GPUs can give >100x speedup

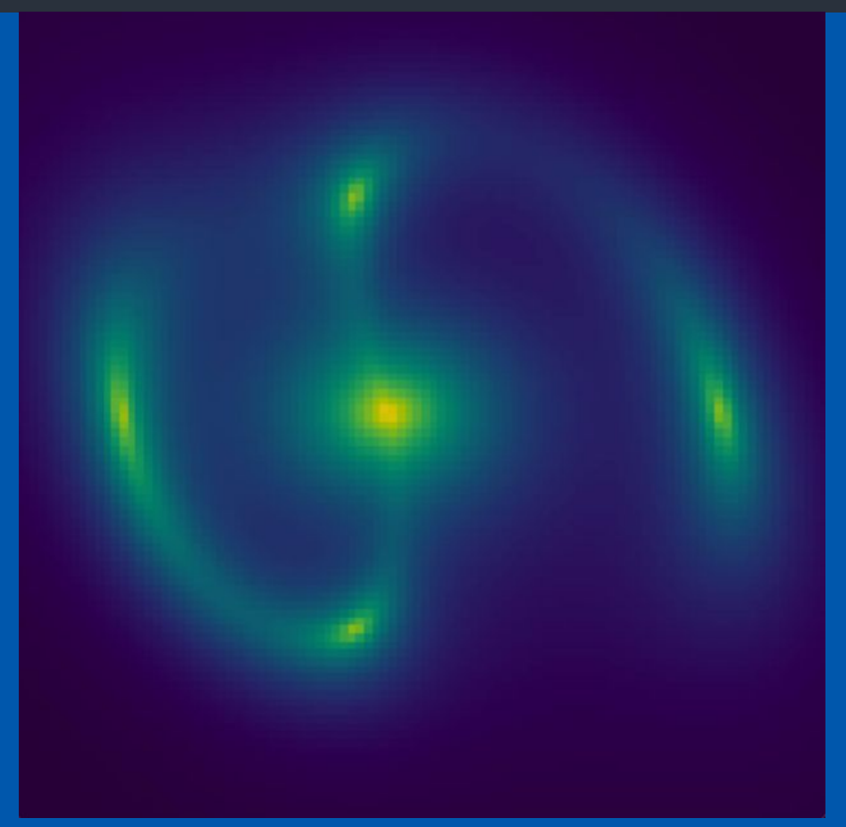


Caustics paper

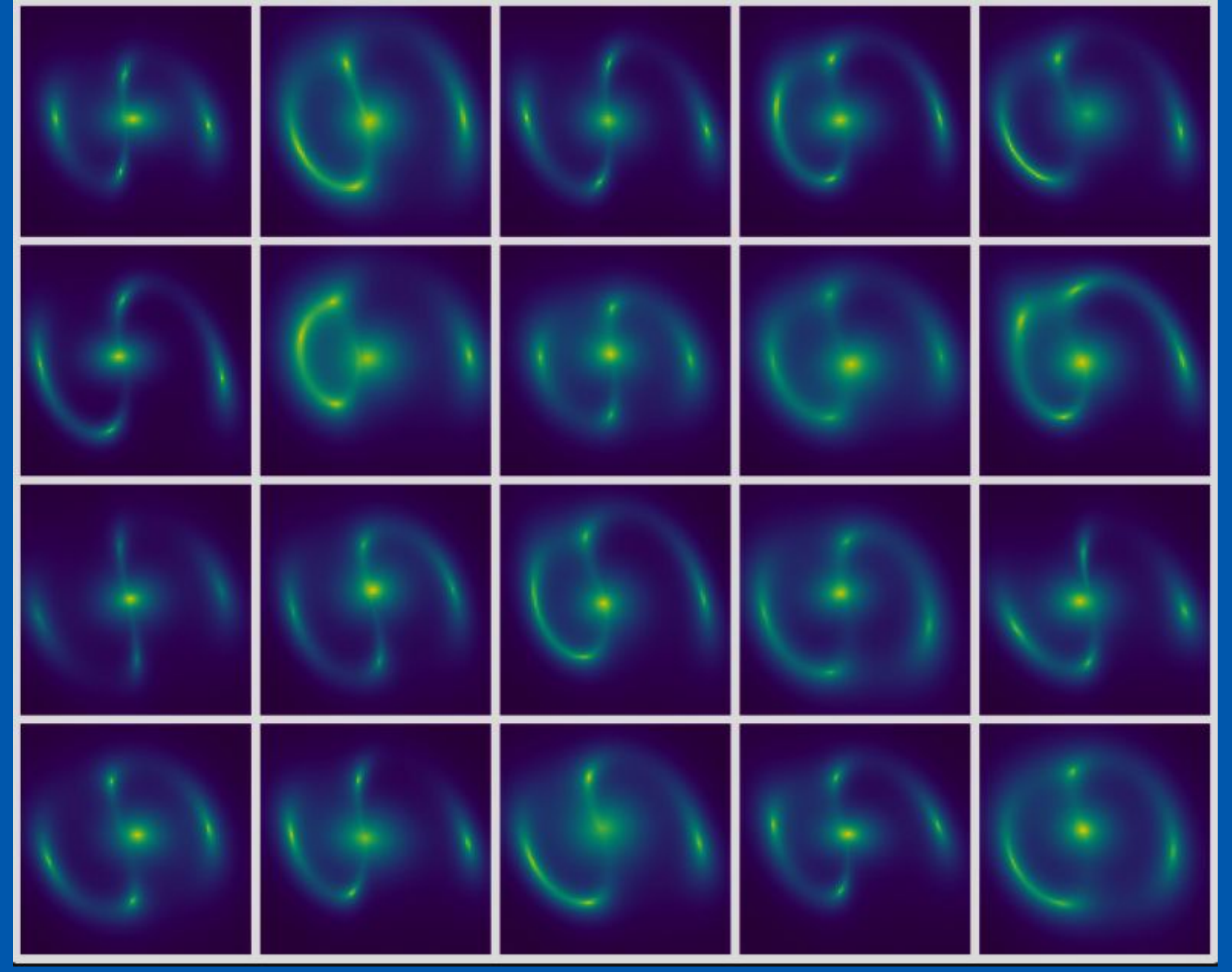
Benefits of PyTorch



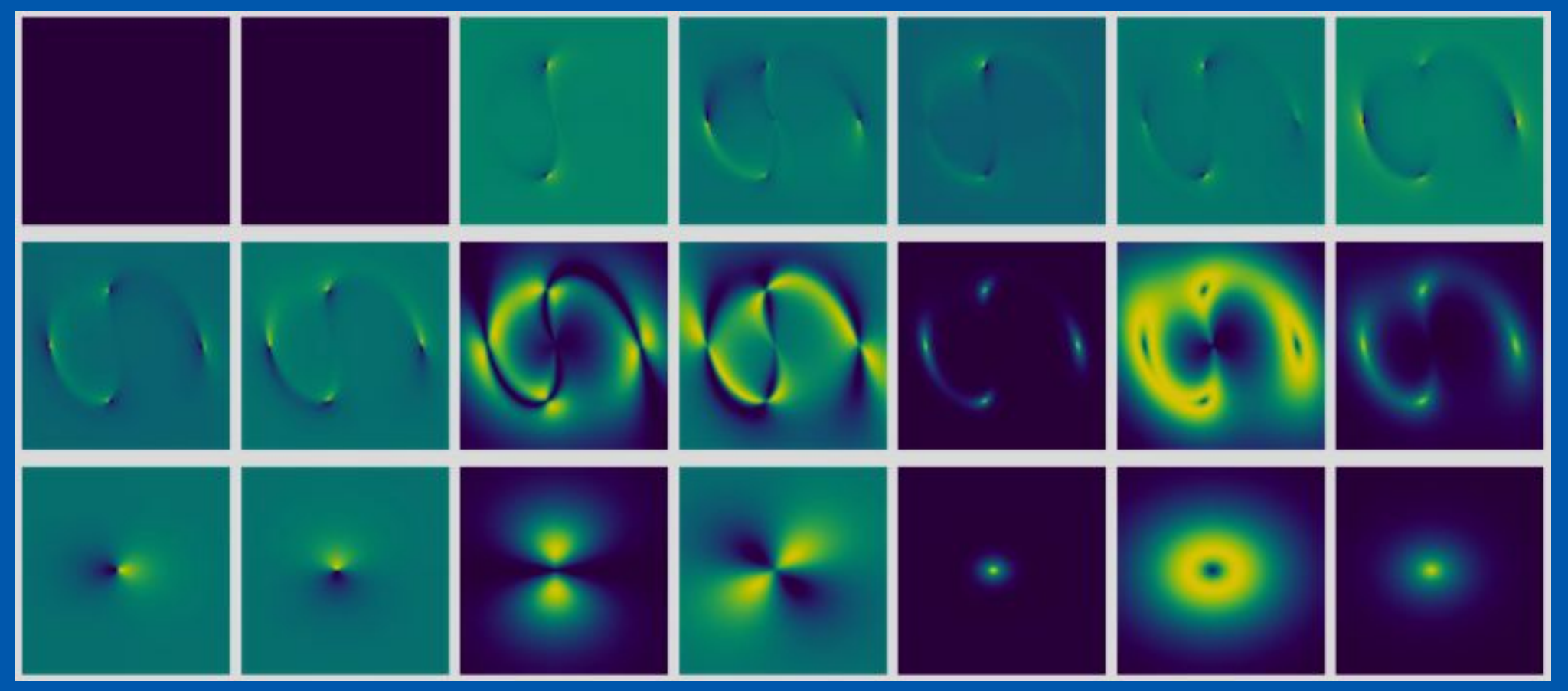
```python
import matplotlib.pyplot as plt
import caustics
import torch

cosmology = caustics.FlatLambdaCDM()
sie = caustics.SIE(cosmology=cosmology, name="lens")
src = caustics.Sersic(name="source")
lnslt = caustics.Sersic(name="lenslight")

x = torch.tensor([
#   z_s  z_l   x0   y0   q    phi     b    x0   y0   q     phi    n    Re
    1.5, 0.5, -0.2, 0.0, 0.4, 1.5708, 1.7, 0.0, 0.0, 0.5, -0.985, 1.3, 1.0,
#   Ie    x0   y0   q    phi  n   Re   Ie
    5.0, -0.2, 0.0, 0.8, 0.0, 1., 1.0, 10.0
])  # fmt: skip

minisim = caustics.LensSource(
    lens=sie, source=src, lens_light=lnslt, pixelscale=0.05, pixels_x=100
)
plt.imshow(minisim(x, quad_level=3), origin="lower")
plt.show()
```



Batches on GPU



Automatic derivatives



Connor Stone, Postdoctoral Fellow
connor.stone@umontreal.ca