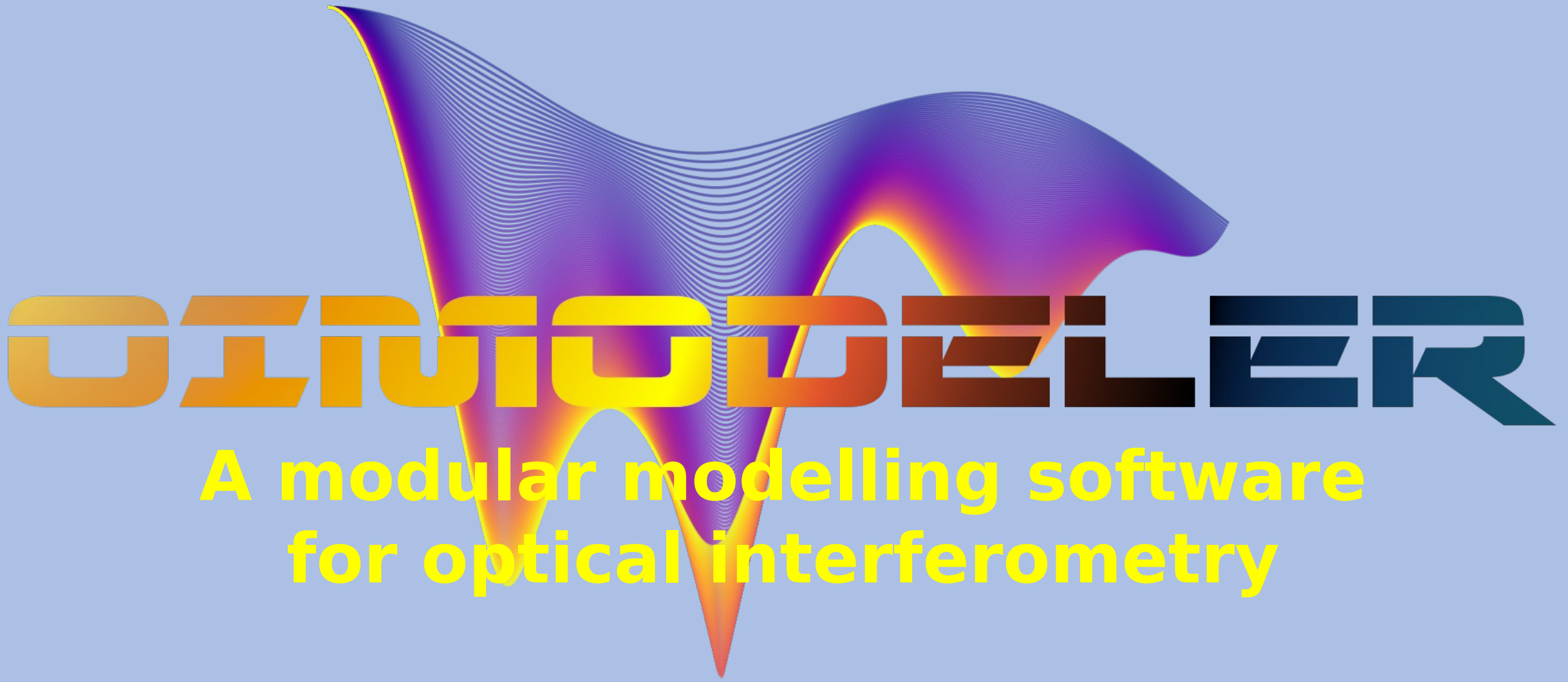
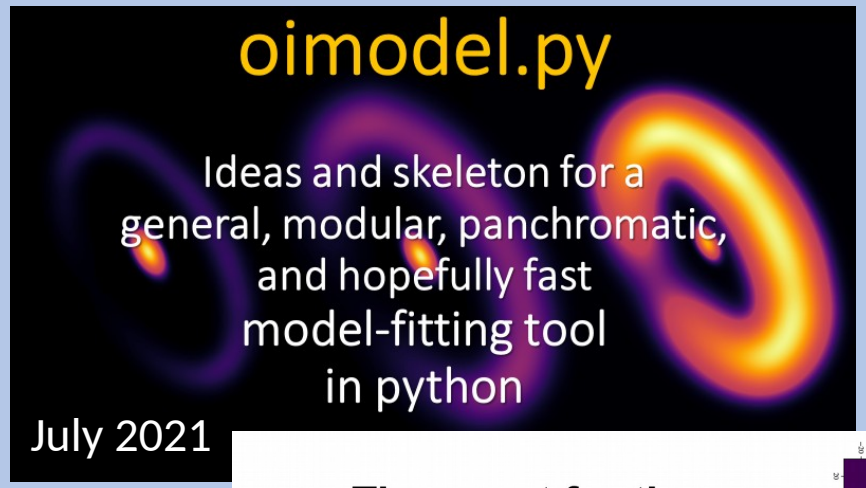
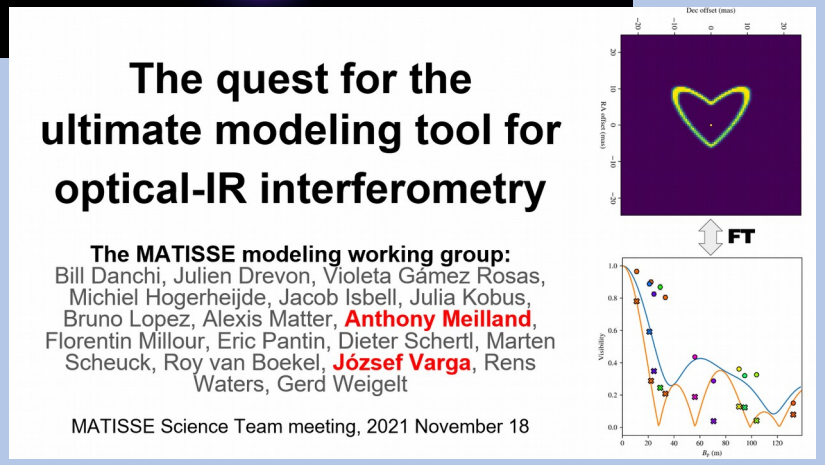


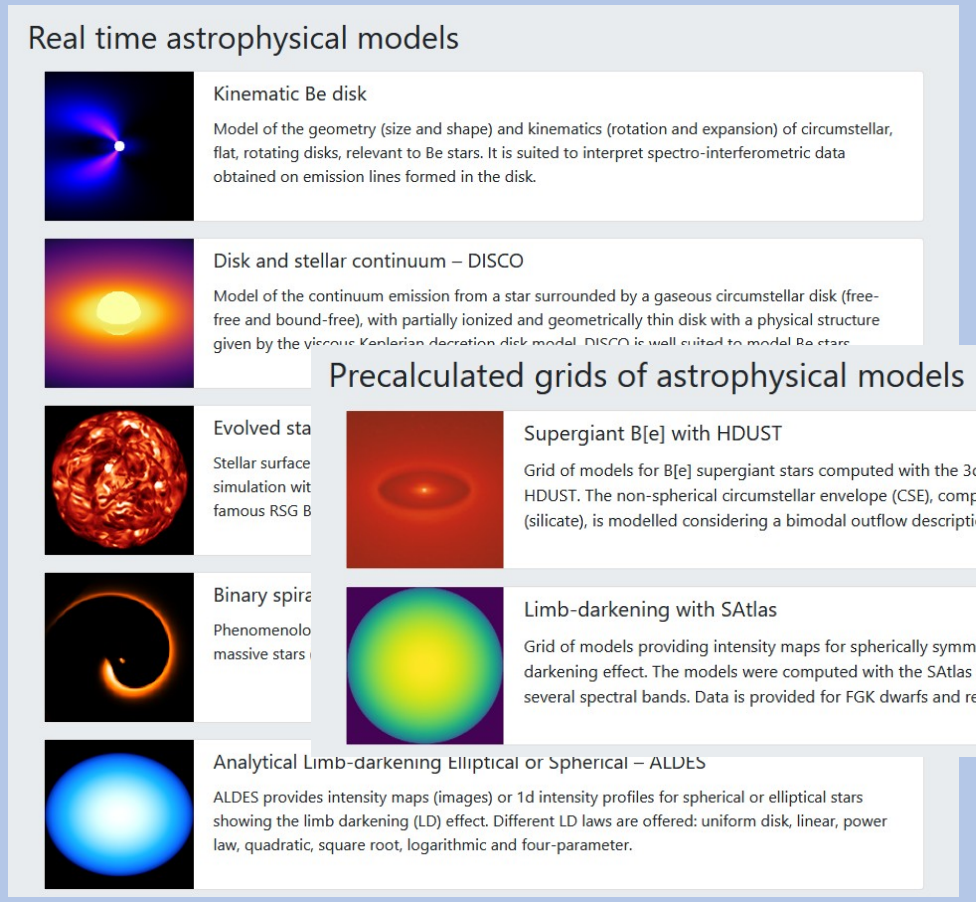
A modular modelling software for optical interferometry

Context of the project



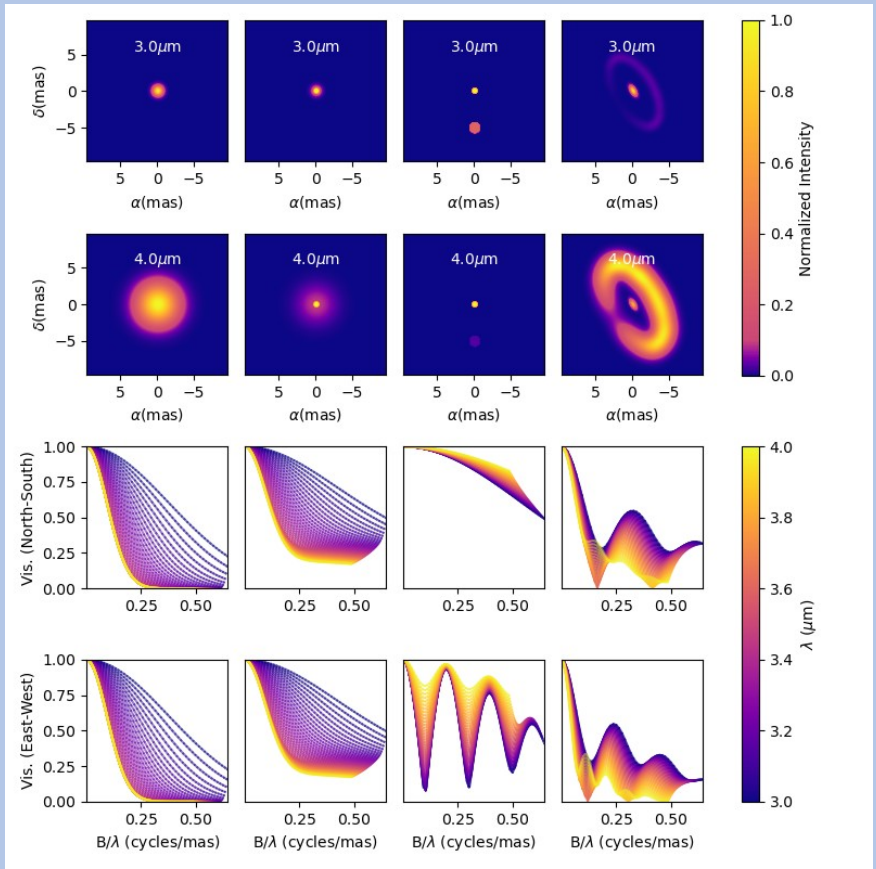


MATISSE group



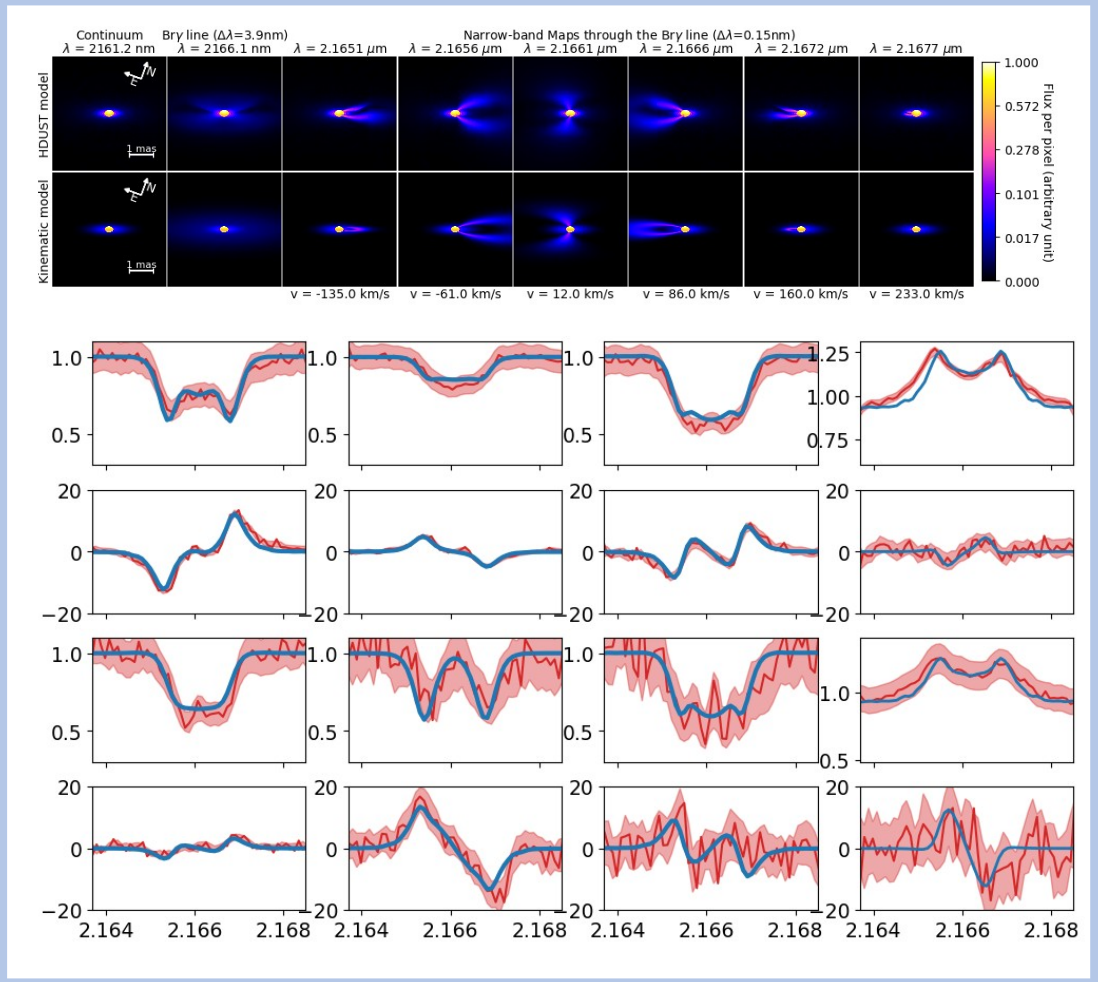
JMMC/AMHRA

Context of the project



Oimodel.py (2021)

A modular & Fourier-based chromatic modelling tool with a emcee (MCMC) fitter

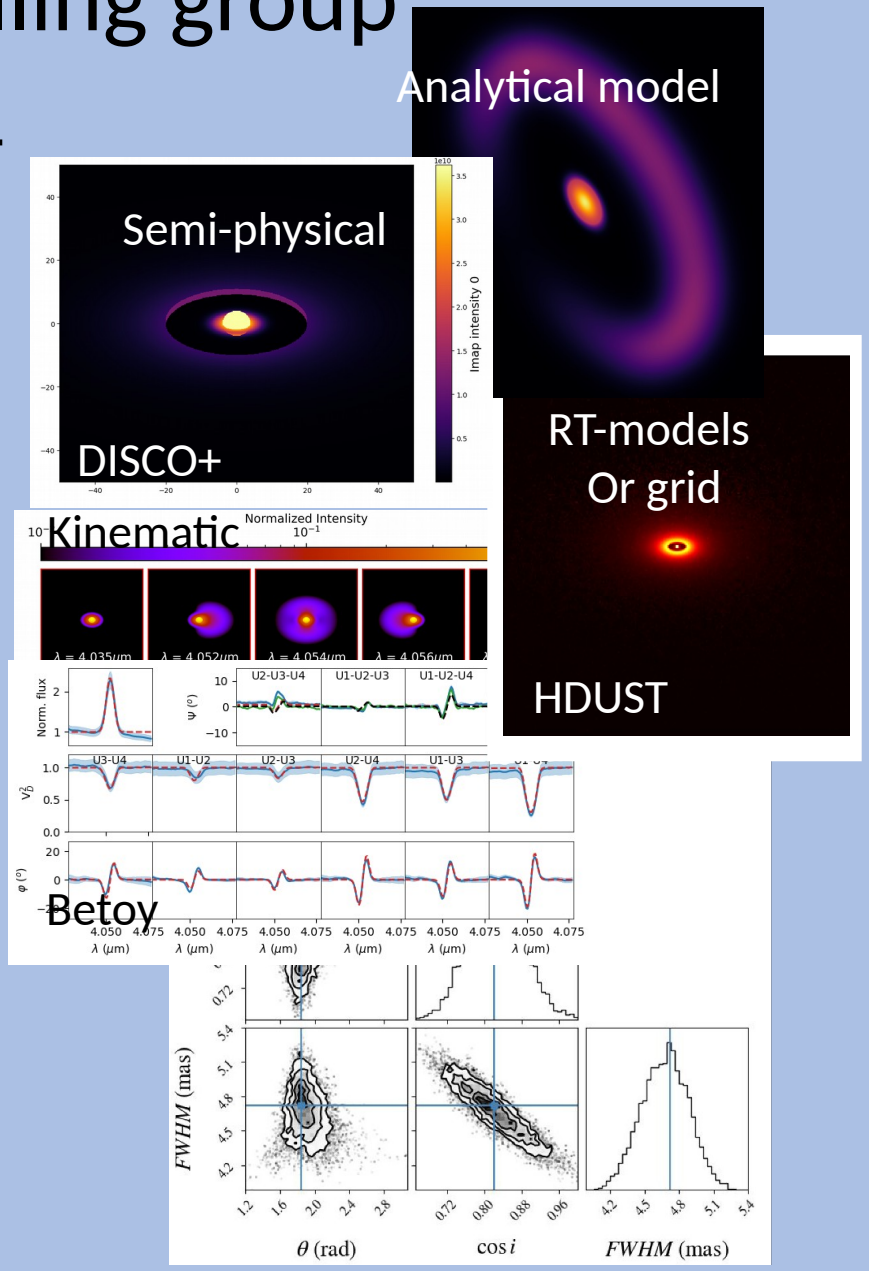


Cubetools.py (2017)

Computing interferometric measurements from chromatic image-cubes

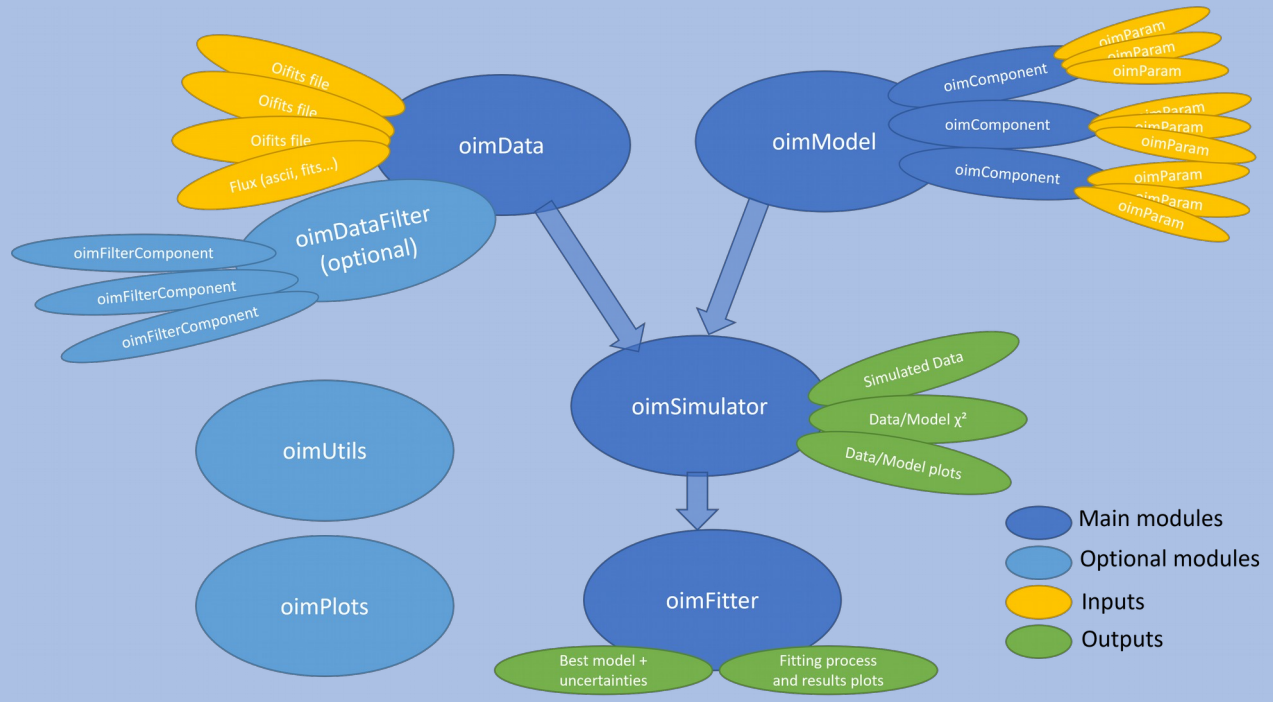
Discussion in MATISSE modelling group
By the end of 2021

- Python3
- Modularity and flexibility
  - Analytical models in Fourier-plan
  - Analytical & numerical models in Image-plan
  - Use outputs from radiative transfer and explore grids of models
  - Build more complex geometries by mixing components
- Chromaticity and time dependence
  - Of the components parameters
  - Chromatic components (such as temperature gradient, binary)
  - Kinematics through line models
- Ability to use interferometric data from all instruments (oifits)
- Produce high-quality publishable outputs
  - Robust estimation of parameters with uncertainties and correlations
  - Nice customizable plots
  - Export simulated data and images to standard format (oifits and fits images)
- Expandability
  - Easily create new components for models (inheritance, wrapping functions)
  - But also other features: type of data, filters, fitters, plots
- Well documented (and with a test suite, examples, tutorials)
- Open source & easily available (Github)



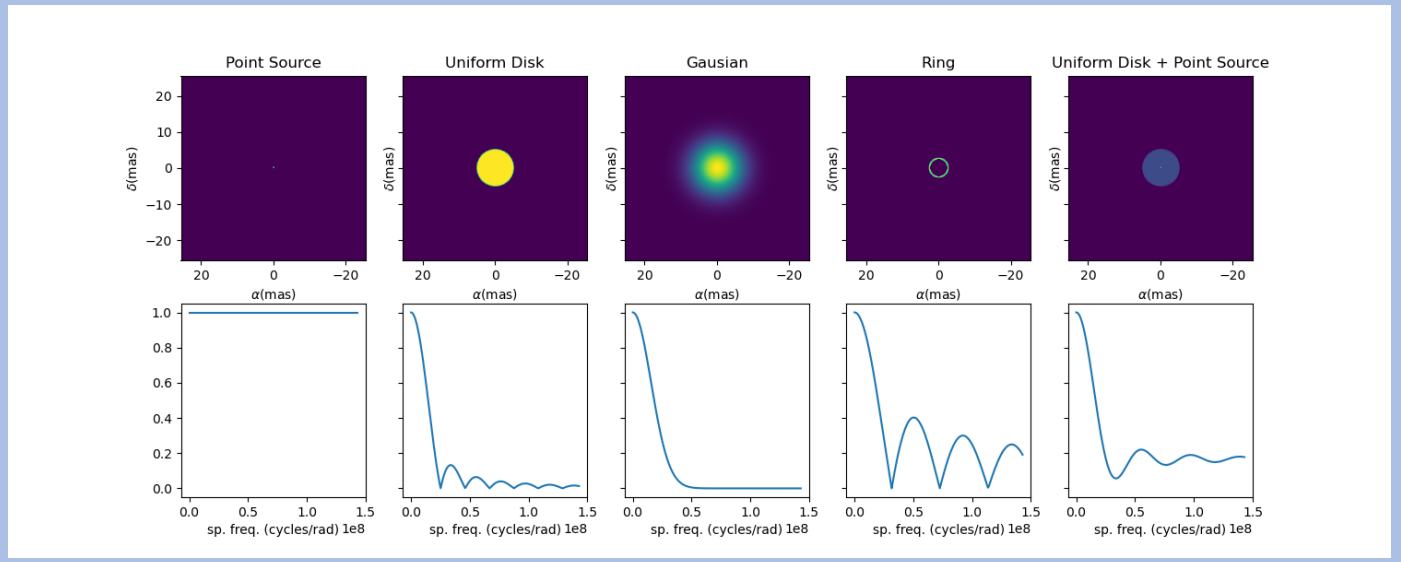
Coding started in 2022

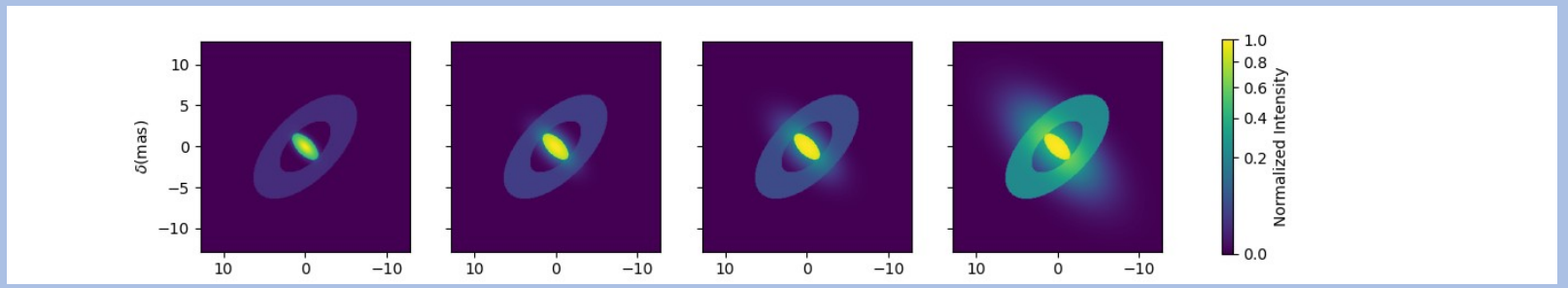
- **January :**
  - Model skeleton : oimParam, oimComponents, oimModel



Coding started in 2022

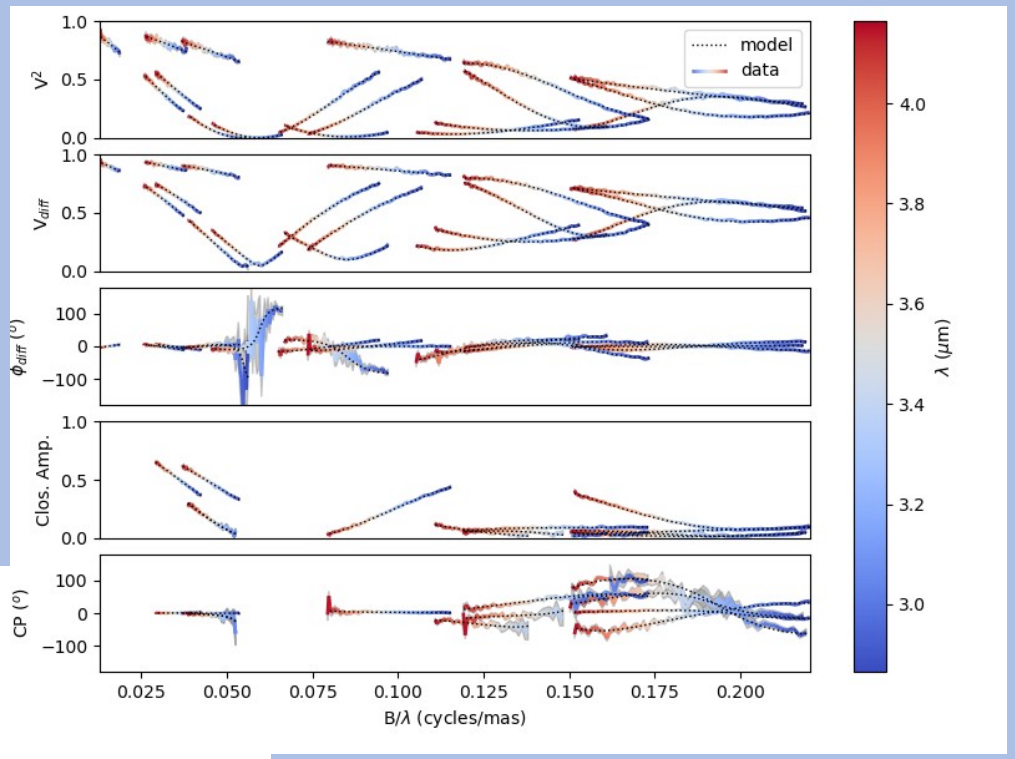
- January :
  - Model skeleton : oimParam, oimComponents, oimModel
- **March-May :**
  - Implementation of Fourier-based components
  - with chromatic parameters (using linear interpolation)
  - Link between parameters

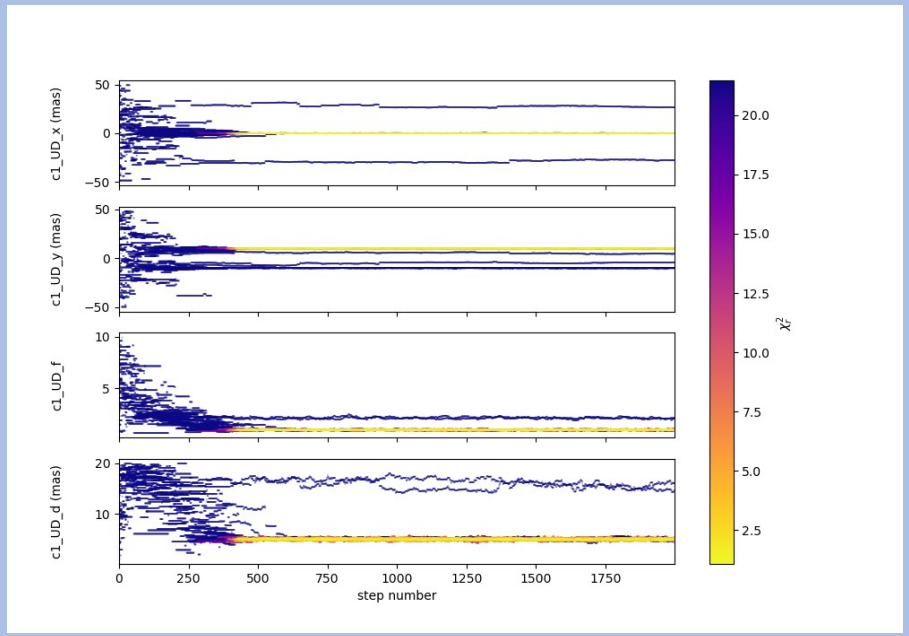


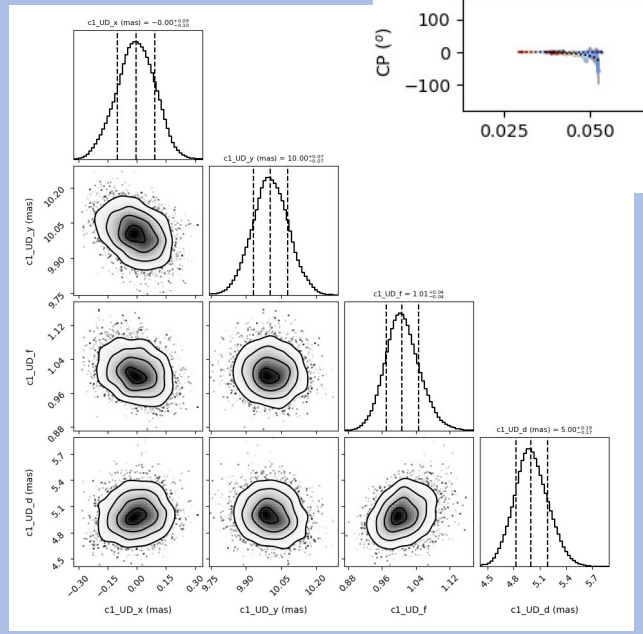


Coding started in 2022

- January :
  - Model skeleton : oimParam, oimComponents, oimModel
- March-May :
  - Implementation of Fourier-based components
  - with chromatic parameters (using linear interpolation)
  - Link between parameters
- **June-August:**
  - oimSimulator class (optimized data, data simulation, chi2)
  - oimFitter class and first emcee Fitter

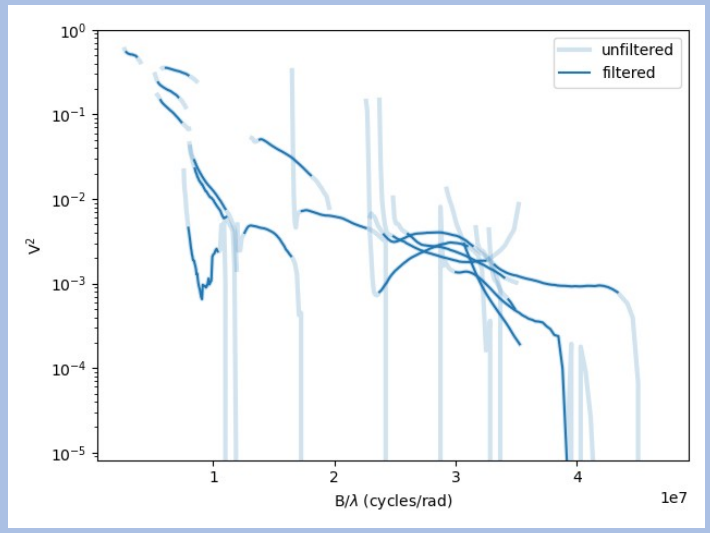


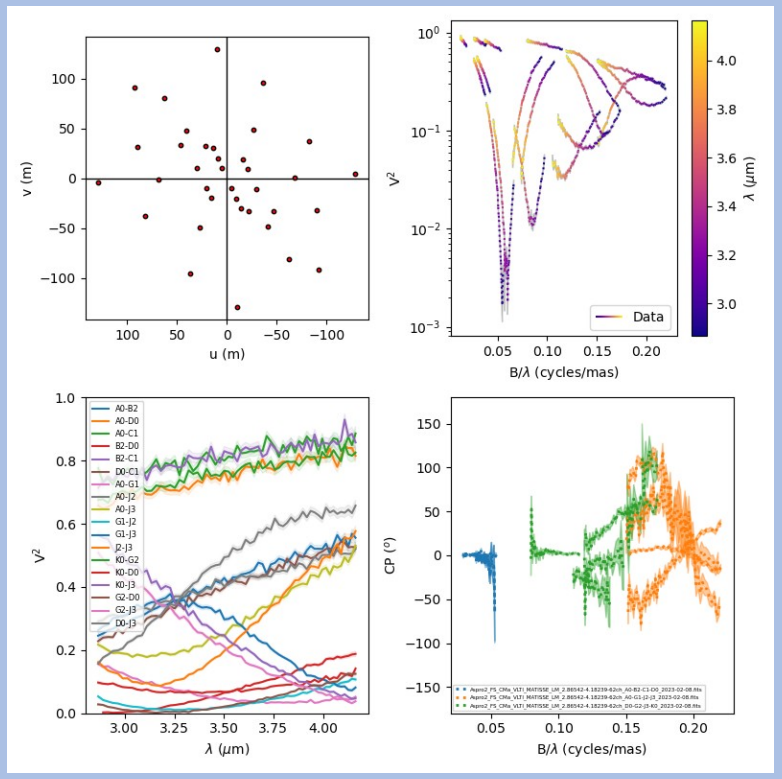




Coding started in 2022

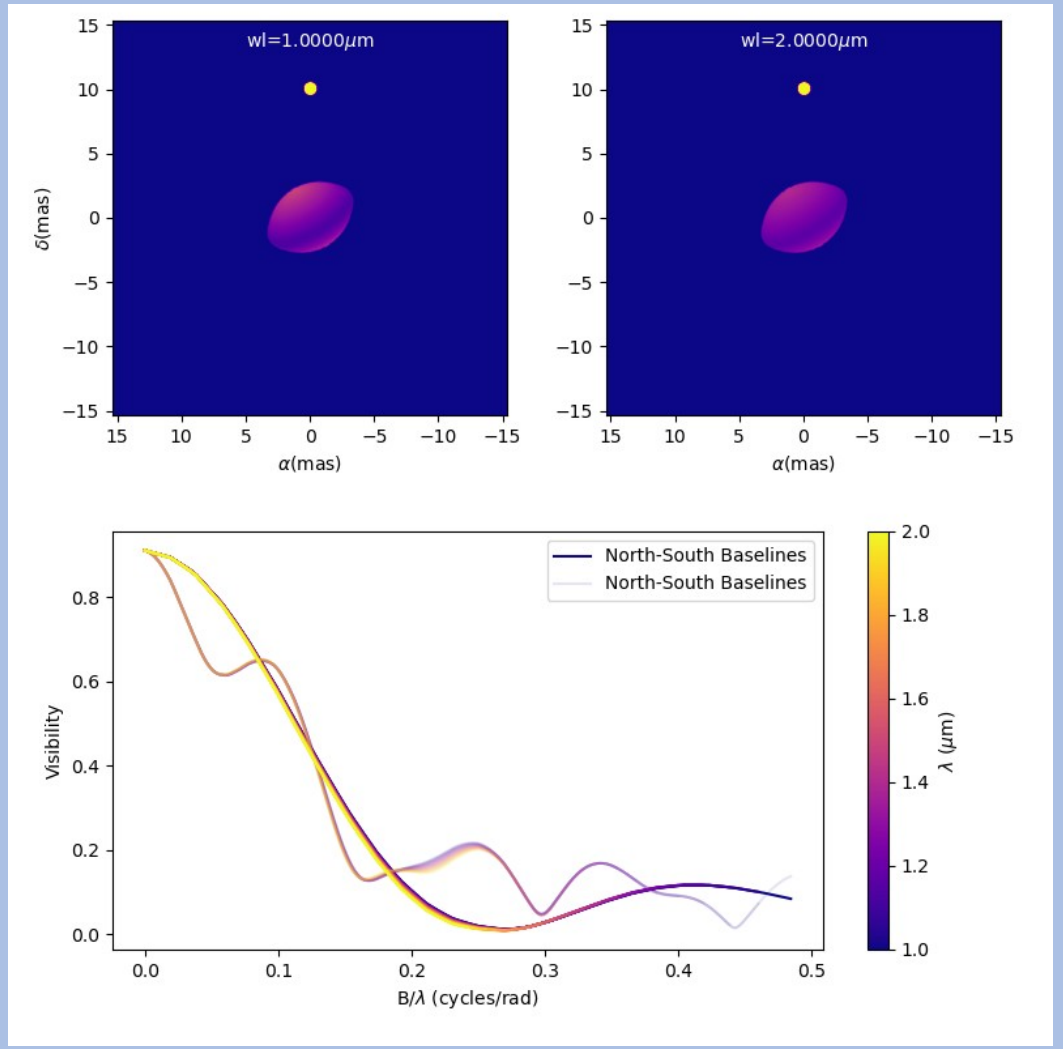
- January :
  - Model skeleton : oimParam, oimComponents, oimModel
- March-May :
  - Implementation of Fourier-based components
  - with chromatic parameters (using linear interpolation)
  - Link between parameters
- June-August:
  - oimSimulator class (optimized data, data simulation, chi2)
  - oimFitter class and first emcee Fitter
- **September:**
  - data filtering & plots
  - documentation (on readthedoc)





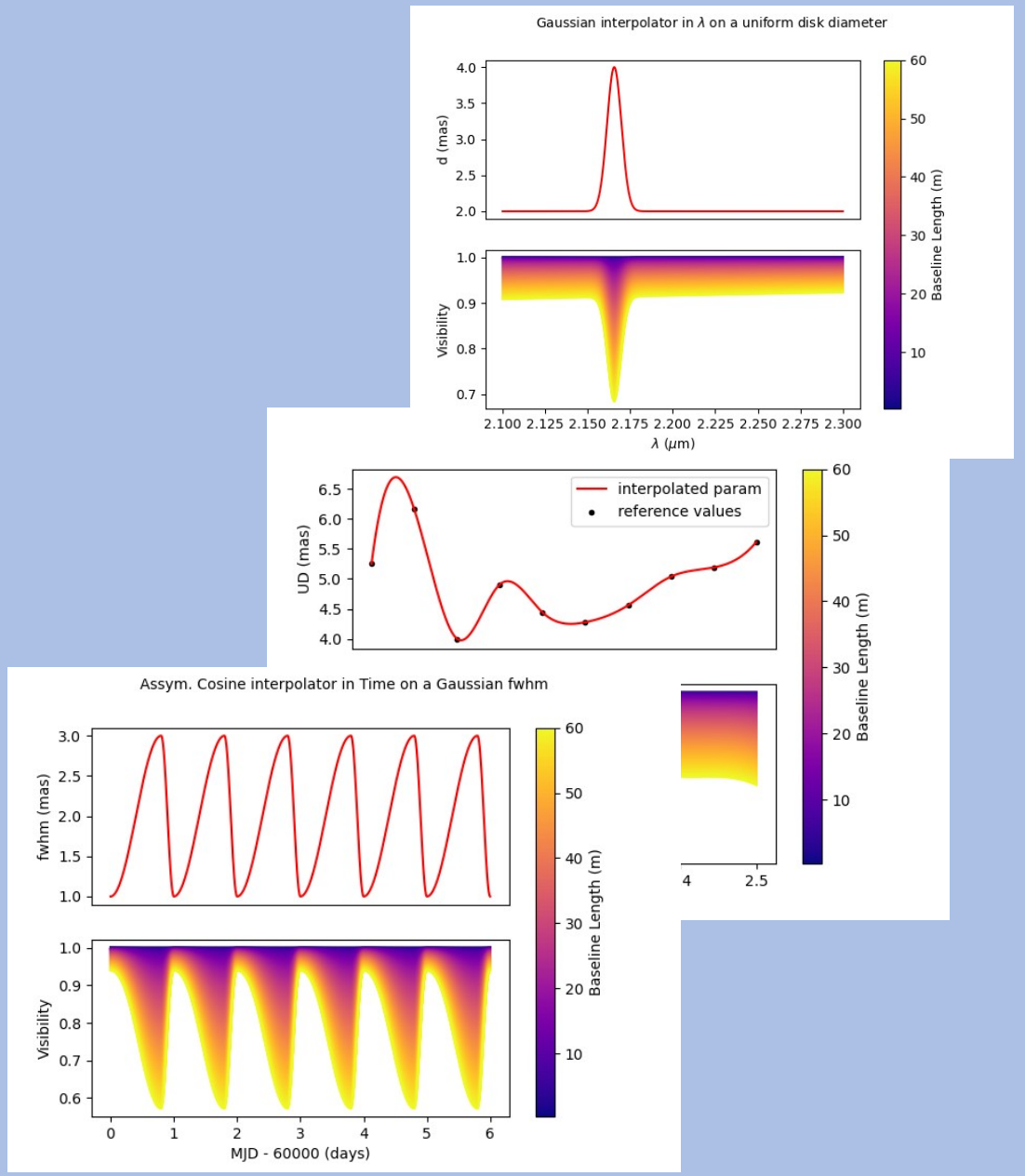
Coding started in 2022

- January :
  - Model skeleton : oimParam, oimComponents, oimModel
- March-May :
  - Implementation of Fourier-based components
  - with chromatic parameters (using linear interpolation)
  - Link between parameters
- June-August:
  - oimSimulator class (optimized data, data simulation, chi2)
  - oimFitter class and first emcee Fitter
- September:
  - data filtering & plots
  - documentation (on readthedoc)
- **October :**
  - image-plan components (FFT, sampling…)



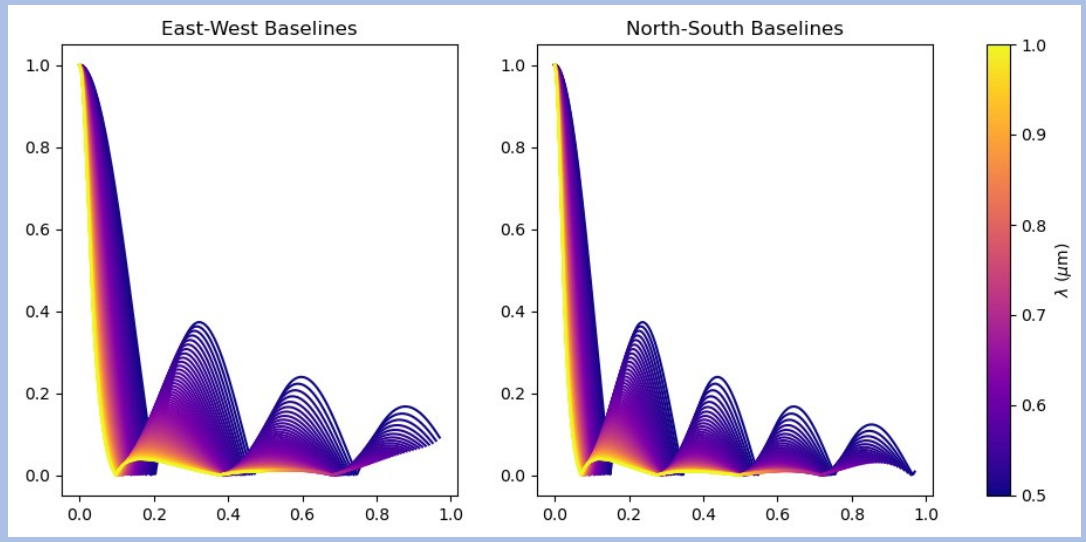
Coding started in 2022

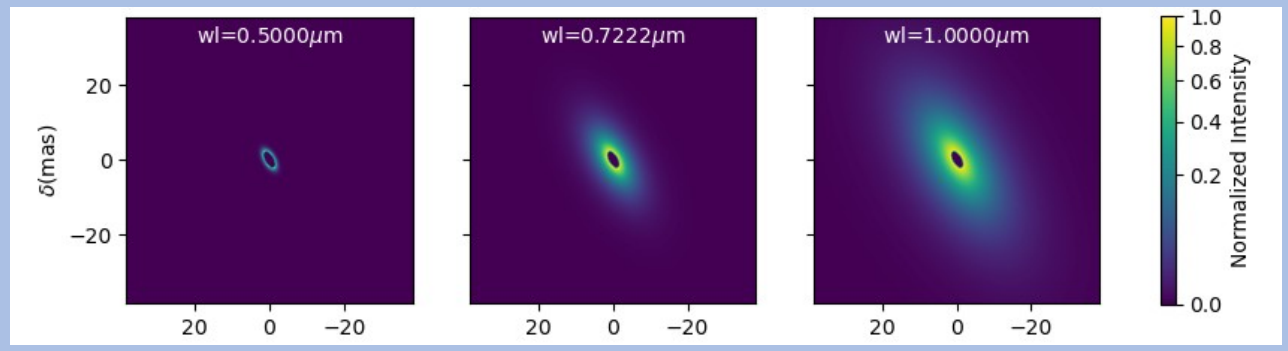
- January
  - Model skeleton : oimParam, oimComponents, oimModel
- March-May
  - Implementation of Fourier-based components
  - with chromatic parameters (using linear interpolation)
  - Link between parameters
- June-August:
  - oimSimulator class (optimized data, data simulation, chi2)
  - oimFitter class and first emcee Fitter
- September:
  - data filtering & plots
  - documentation (on readthedoc)
- **October :**
  - image-plan components (FFT, sampling…)
  - parameter interpolators (time and chromaticity)



Coding started in 2022

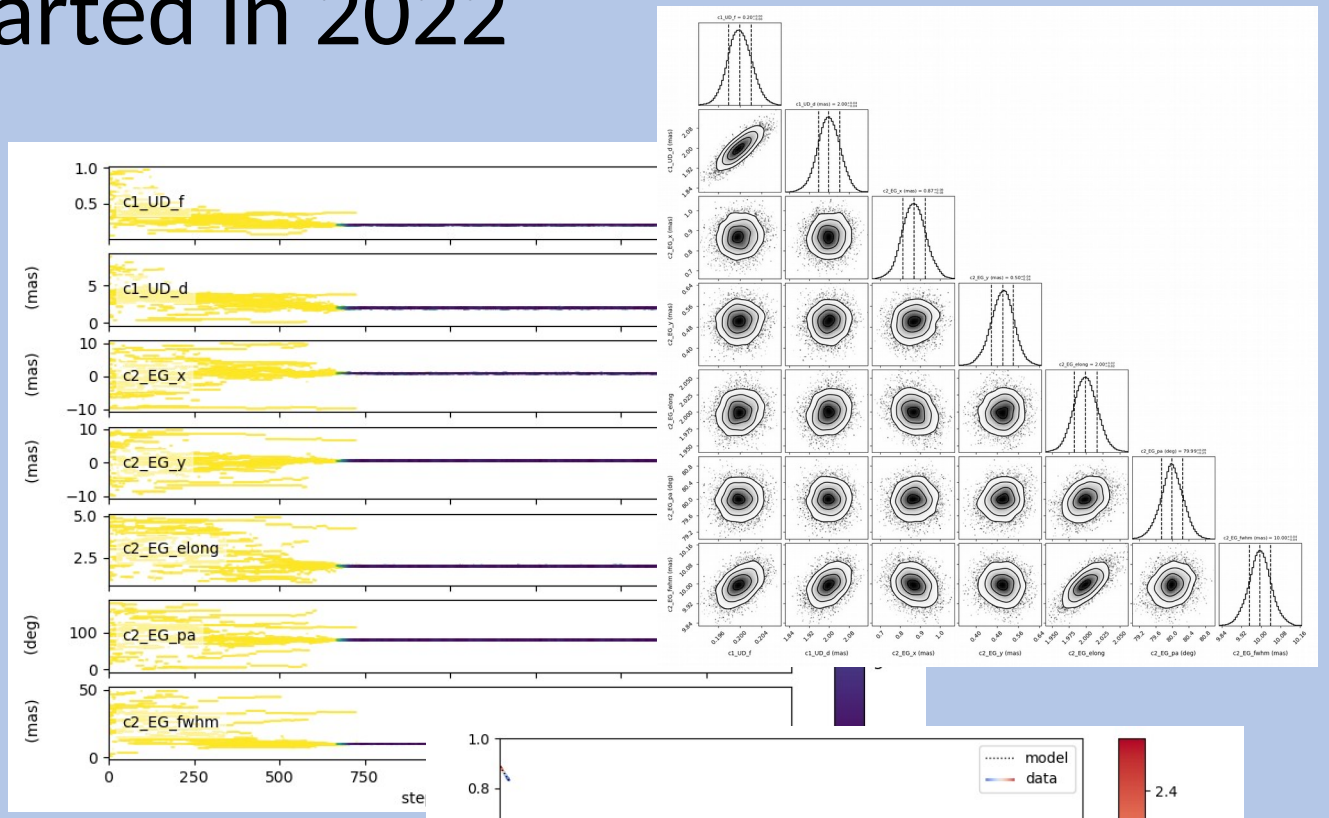
- January
  - Model skeleton : oimParam, oimComponents, oimModel
- March-May
  - Implementation of Fourier-based components
  - with chromatic parameters (using linear interpolation)
  - Link between parameters
- June-August:
  - oimSimulator class (optimized data, data simulation, chi2)
  - oimFitter class and first emcee Fitter
- September:
  - data filtering & plots
  - documentation (on readthedoc)
- October :
  - image-plan components (FFT, sampling…)
  - parameter interpolators (time and chromaticity)
- **November :**
  - radial profile components (Hankel Transform)

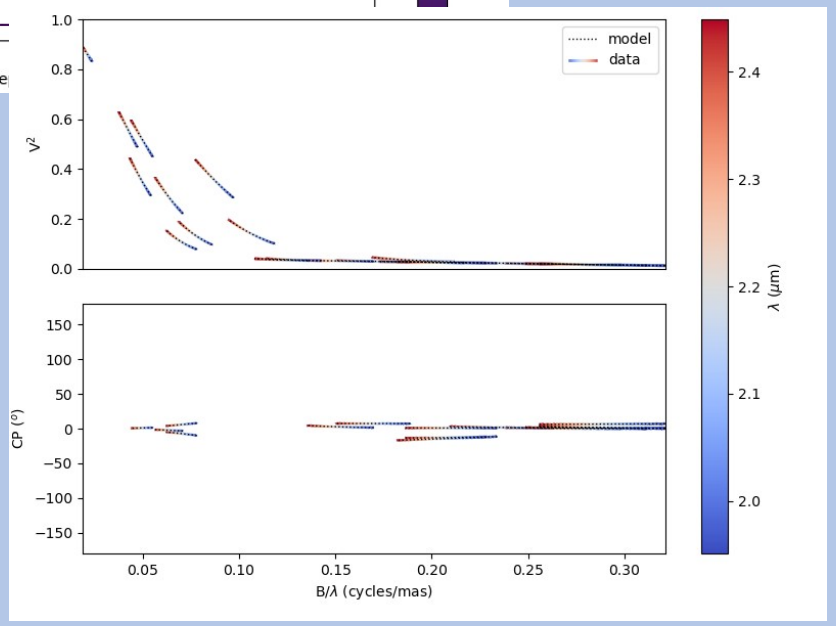




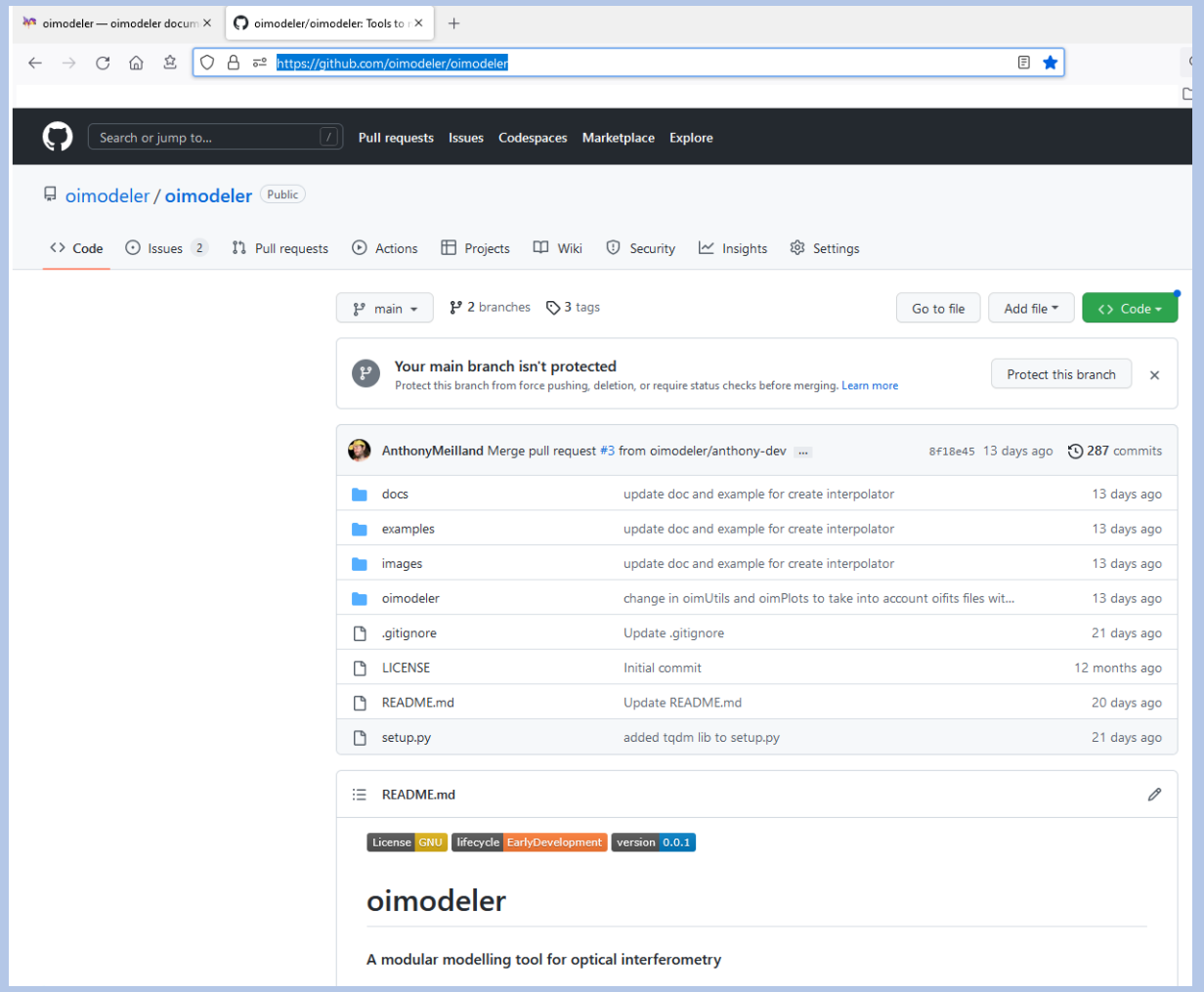
Coding started in 2022

- January
  - Model skeleton : oimParam, oimComponents, oimModel
- March-May
  - Implementation of Fourier-based components
  - with chromatic parameters (using linear interpolation)
  - Link between parameters
- June-August:
  - oimSimulator class (optimized data, data simulation, chi2)
  - oimFitter class and first emcee Fitter
- September:
  - data filtering & plots
  - documentation (on readthedoc)
- October :
  - image-plan components (FFT, sampling…)
  - parameter interpolators (time and chromaticity)
- November :
  - radial profile components (Hankel Transform)
- **December :**
  - Code cleaning + examples
  - Test suite (ASPRO simulated data and real data from various instruments)

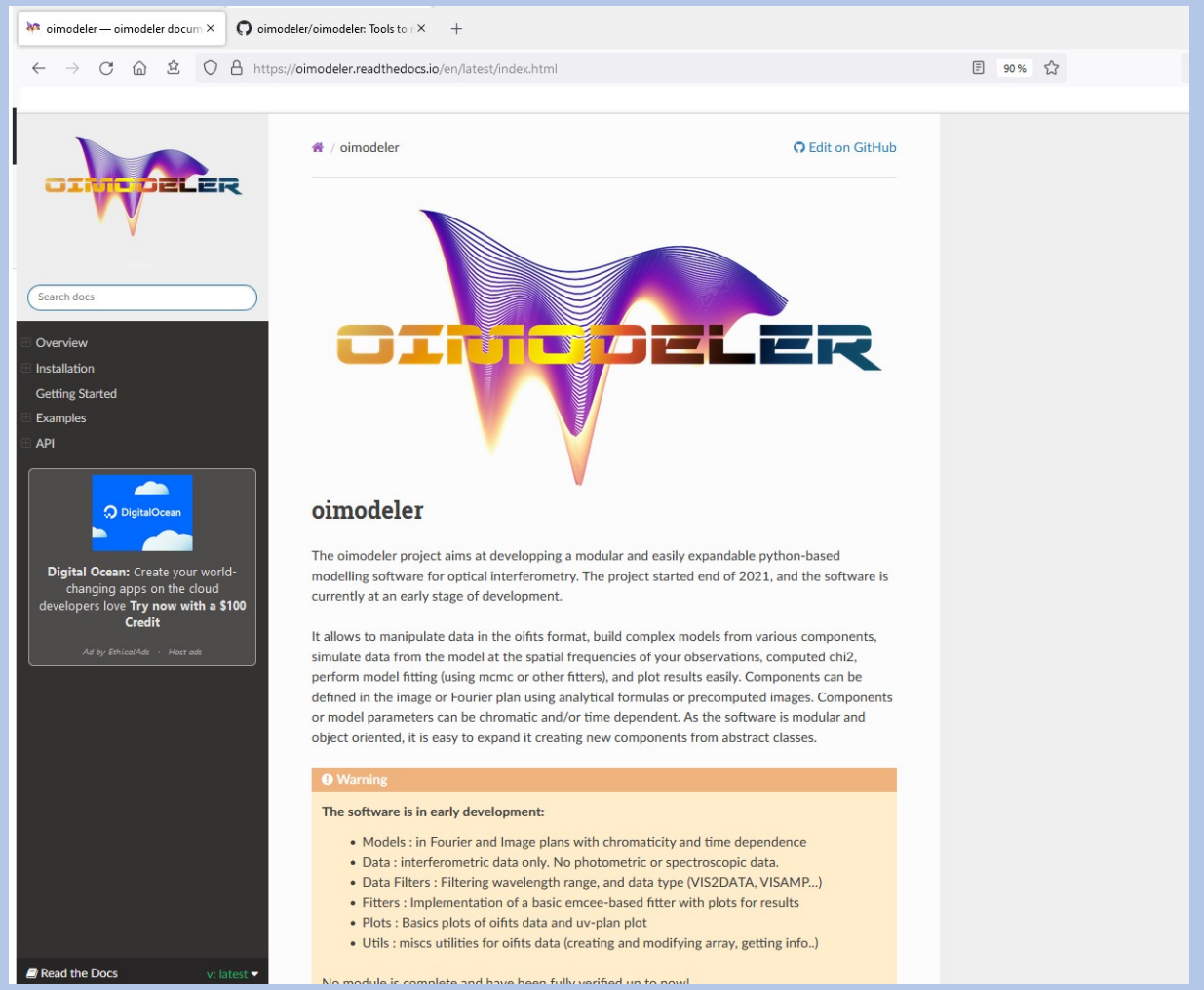




Oimodeler on the web



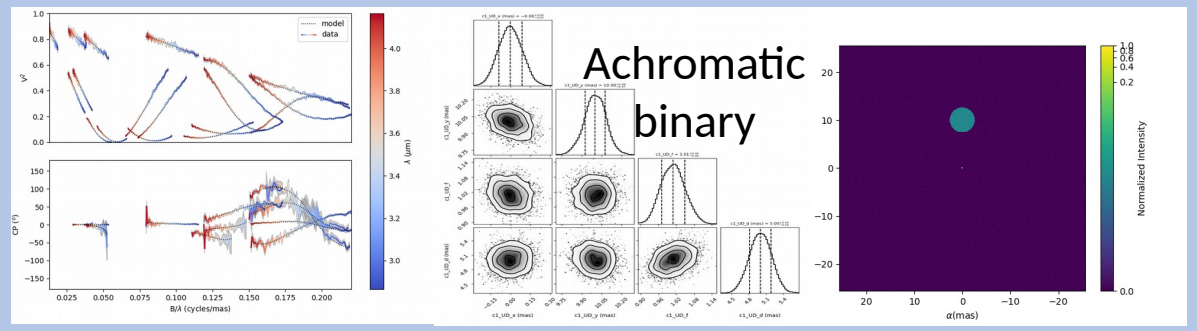
Code available on github
https://github.com/oimodeler



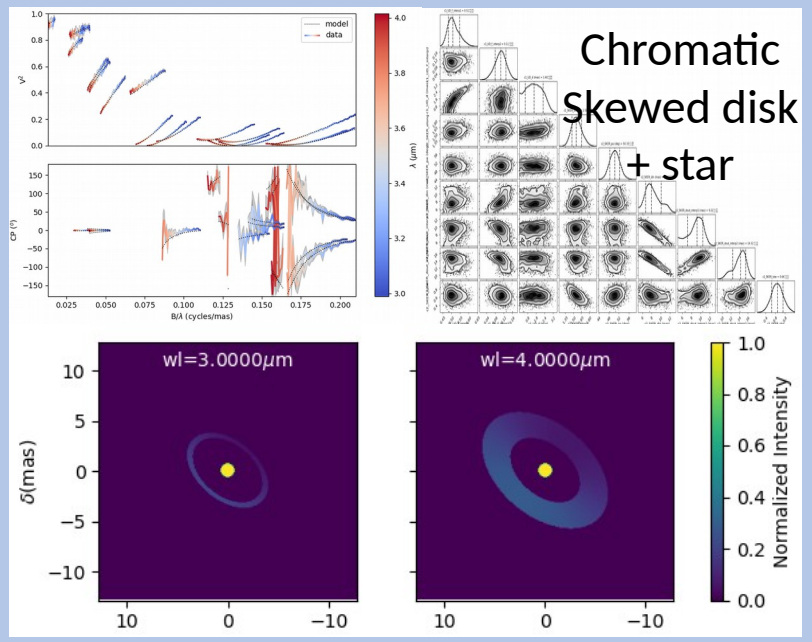
Documentation available on readthedocs
https://oimodeler.readthedocs.io

A Few examples

Model-fitting (on simulated data)

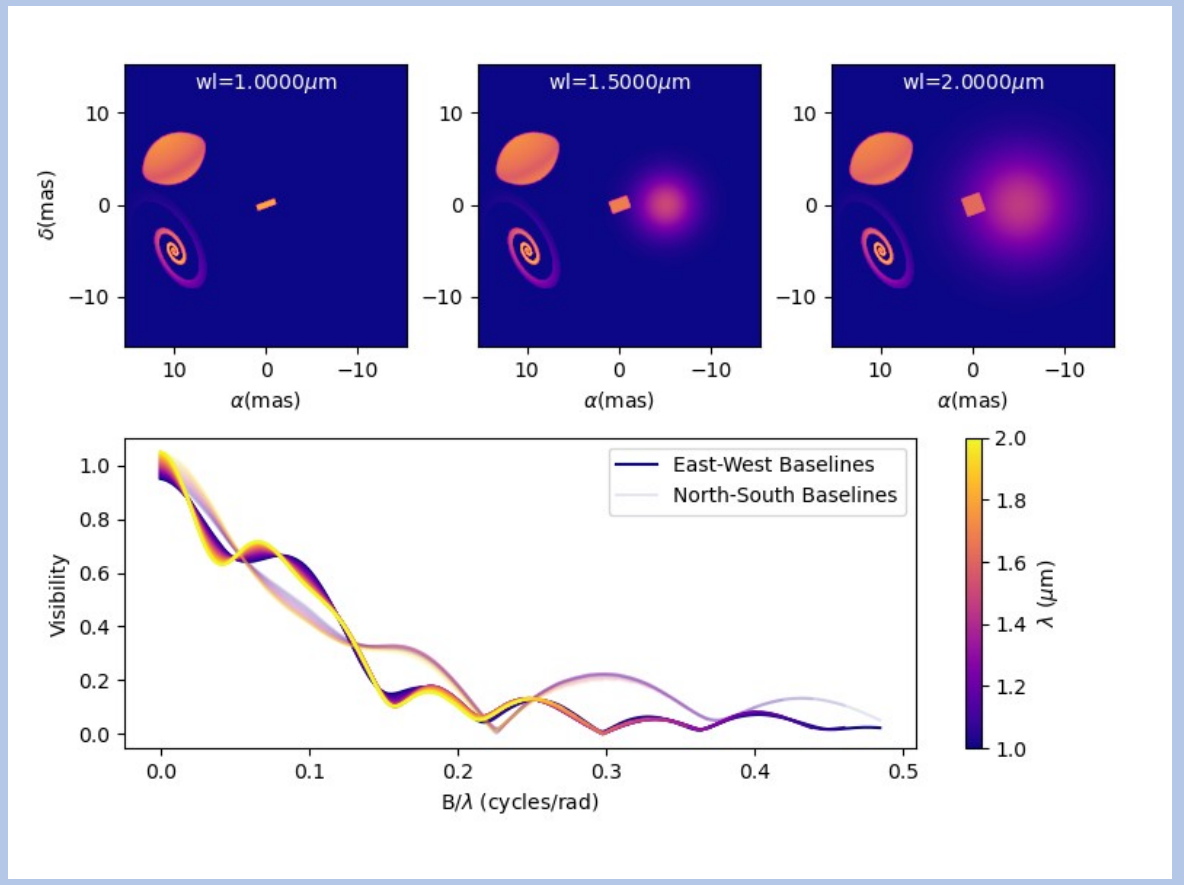


Achromatic binary



Chromatic Skewed disk + star

Adding new components



Example of simple model-fitting

```
import oimodeler as oim
```

Import oimodeler

```
ud = oim.oimUD(d=3,f=0.5,x=5,y=-5)
pt = oim.oimPt(f=1)
model = oim.oimModel(ud,pt)
```

Create a binary model

```
ud.params['d'].set(min=0.01,max=20)
ud.params['x'].set(min=-50,max=50,free=True)
ud.params['y'].set(min=-50,max=50,free=True)
ud.params['f'].set(min=0.,max=10.)
```

Set the parameters space

```
pt.params['f']=oim.oimParamNorm(ud.params['f'])
```

Normalizing flux to 1 (remove one free parameter)

```
fit = oim.oimFitterEmcee(files,model,nwalkers=20)
fit.prepare(init="random")
fit.run(nsteps=2000, progress=True)
```

Create, prepare and run the MCMC fitter

```
median,err_l,err_u,err=fit.getResults(discard=1000)
```

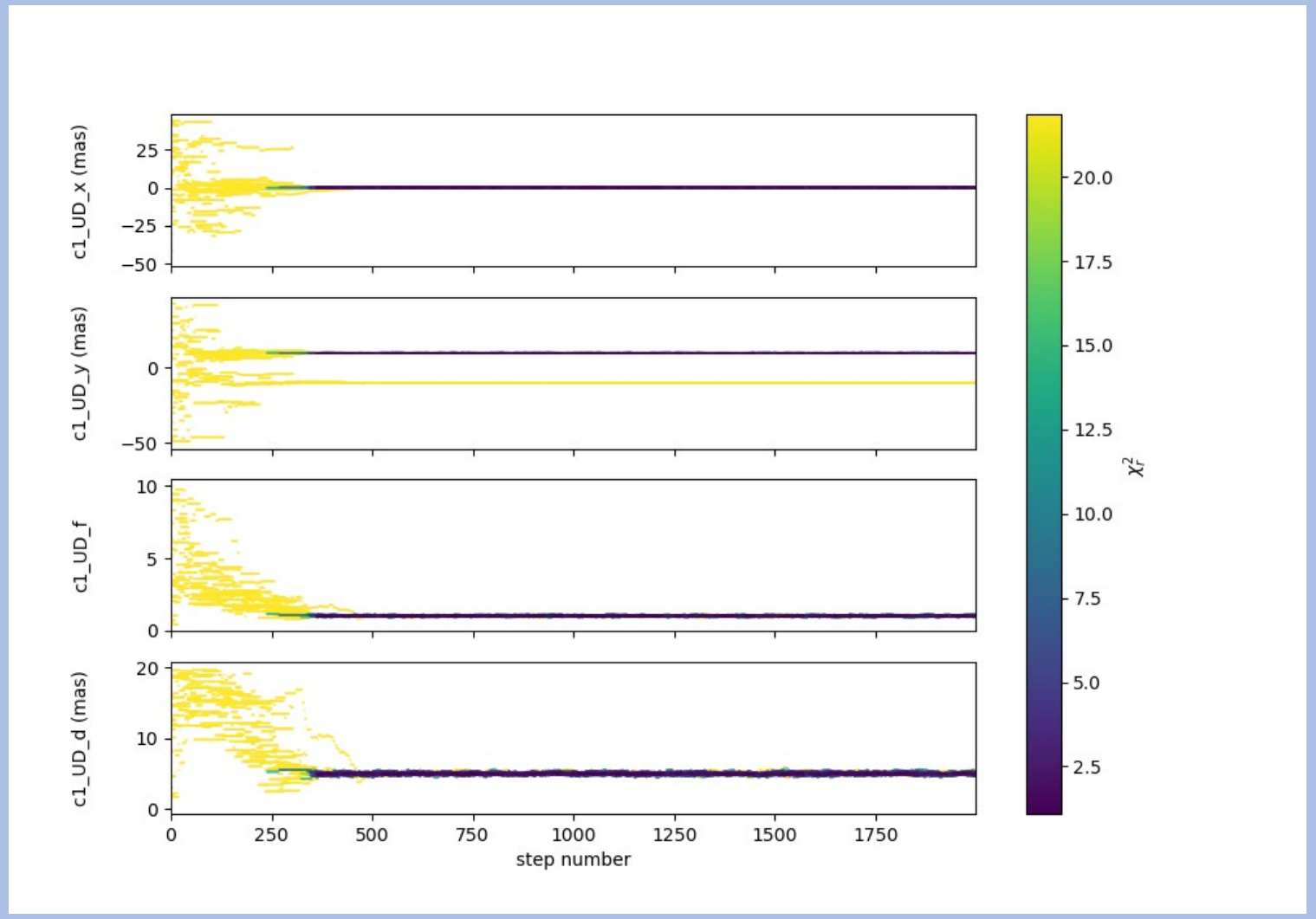
Get the results and plots

```
fit.walkersPlot()
fit.cornerPlot()
fit.simulator.plot(["VIS2DATA","T3PHI"])
model.showModel(512,0.1)
```

Make nice plots

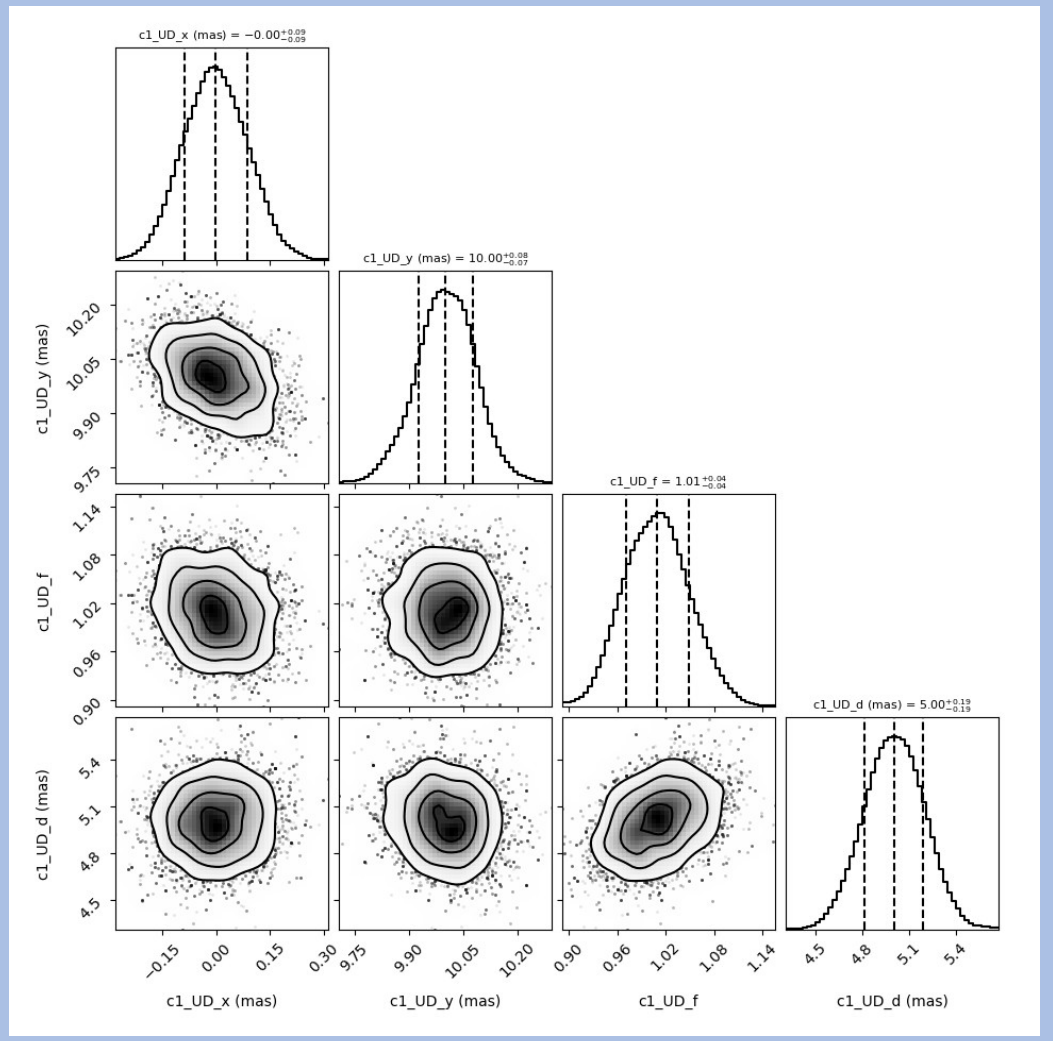
Example of simple model-fitting

```
import oimodeler as oim

ud = oim.oimUD(d=3,f=0.5,x=5,y=-5)
pt = oim.oimPt(f=1)
model = oim.oimModel(ud,pt)

ud.params['d'].set(min=0.01,max=20)
ud.params['x'].set(min=-50,max=50,free=True)
ud.params['y'].set(min=-50,max=50,free=True)
ud.params['f'].set(min=0.,max=10.)

pt.params['f']=oim.oimParamNorm(ud.params['f'])

fit = oim.oimFitterEmcee(files,model,nwalkers=20)
fit.prepare(init="random")
fit.run(nsteps=2000, progress=True)

median,err_l,err_u,err=fit.getResults(discard=1000)

fit.walkersPlot()
fit.cornerPlot()
fit.simulator.plot(["VIS2DATA","T3PHI"])
model.showModel(512,0.1)
```



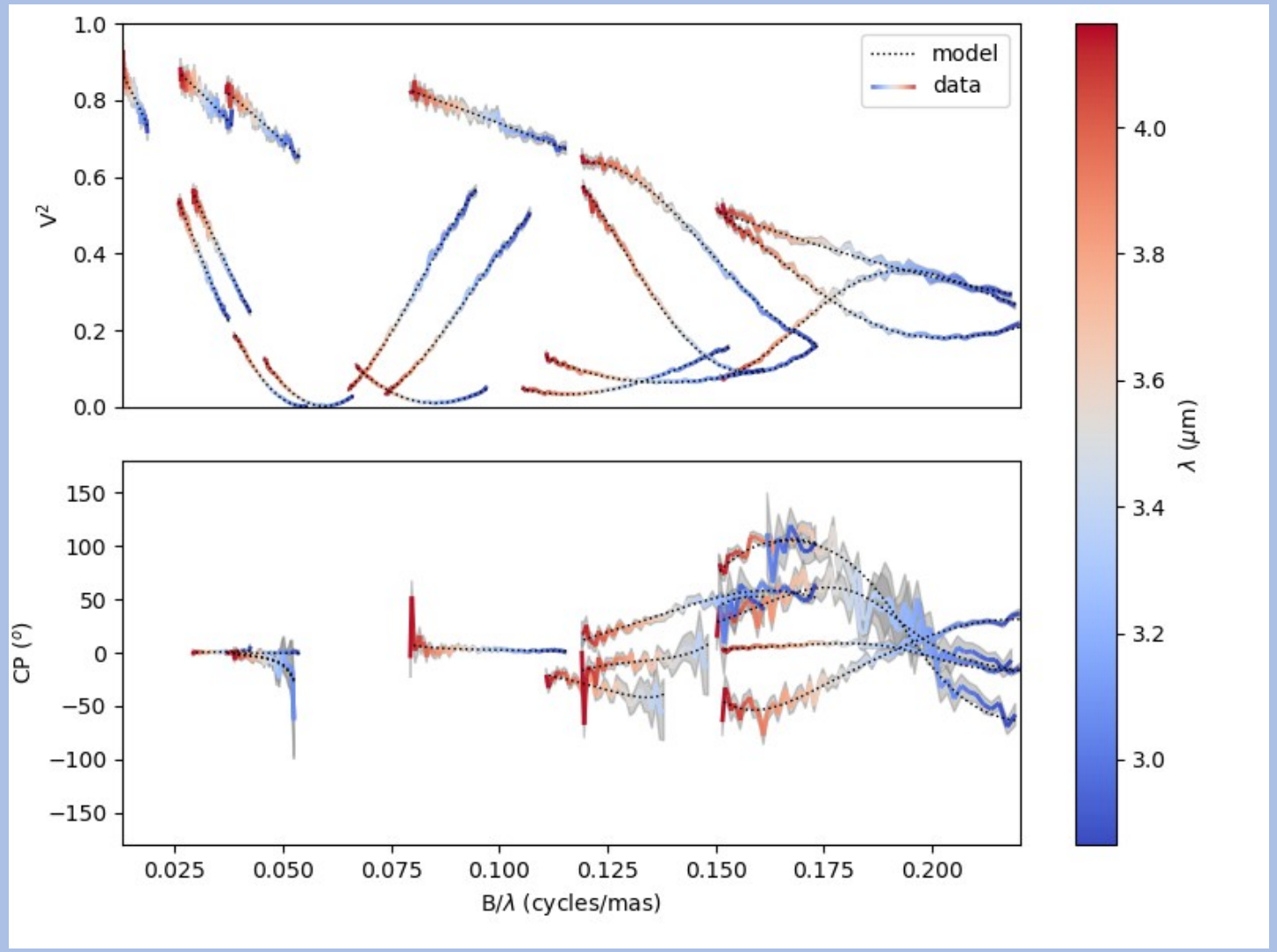
Example of simple model-fitting

```
import oimodeler as oim

ud = oim.oimUD(d=3,f=0.5,x=5,y=-5)
pt = oim.oimPt(f=1)
model = oim.oimModel(ud,pt)

ud.params['d'].set(min=0.01,max=20)
ud.params['x'].set(min=-50,max=50,free=True)
ud.params['y'].set(min=-50,max=50,free=True)
ud.params['f'].set(min=0.,max=10.)

pt.params['f']=oim.oimParamNorm(ud.params['f'])

fit = oim.oimFitterEmcee(files,model,nwalkers=20)
fit.prepare(init="random")
fit.run(nsteps=2000, progress=True)

median,err_l,err_u,err=fit.getResults(discard=1000)

fit.walkersPlot()
fit.cornerPlot()
fit.simulator.plot(["VIS2DATA","T3PHI"])
model.showModel(512,0.1)
```



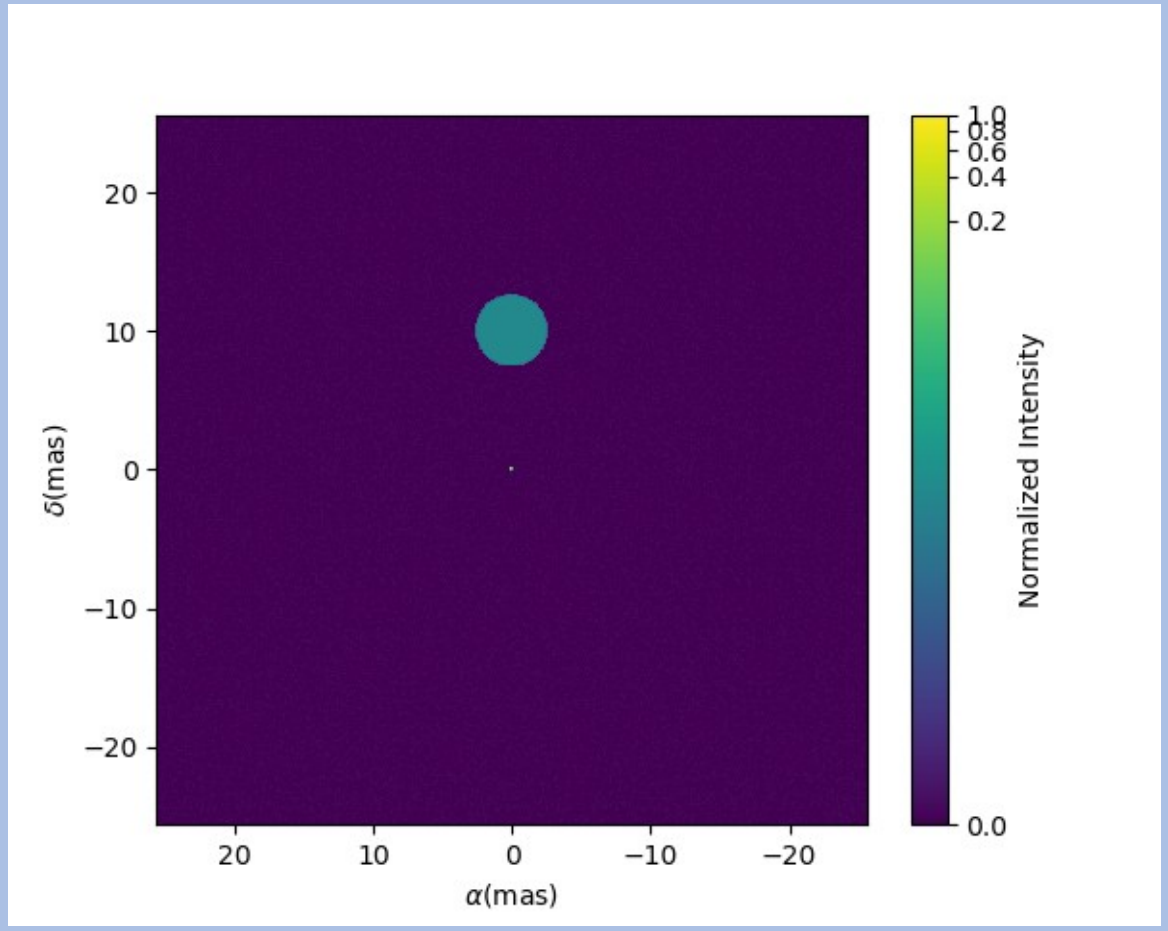
Example of simple model-fitting

```
import oimodeler as oim

ud = oim.oimUD(d=3,f=0.5,x=5,y=-5)
pt = oim.oimPt(f=1)
model = oim.oimModel(ud,pt)

ud.params['d'].set(min=0.01,max=20)
ud.params['x'].set(min=-50,max=50,free=True)
ud.params['y'].set(min=-50,max=50,free=True)
ud.params['f'].set(min=0.,max=10.)

pt.params['f']=oim.oimParamNorm(ud.params['f'])

fit = oim.oimFitterEmcee(files,model,nwalkers=20)
fit.prepare(init="random")
fit.run(nsteps=2000, progress=True)

median,err_l,err_u,err=fit.getResults(discard=1000)

fit.walkersPlot()
fit.cornerPlot()
fit.simulator.plot(["VIS2DATA","T3PHI"])
model.showModel(512,0.1)
```



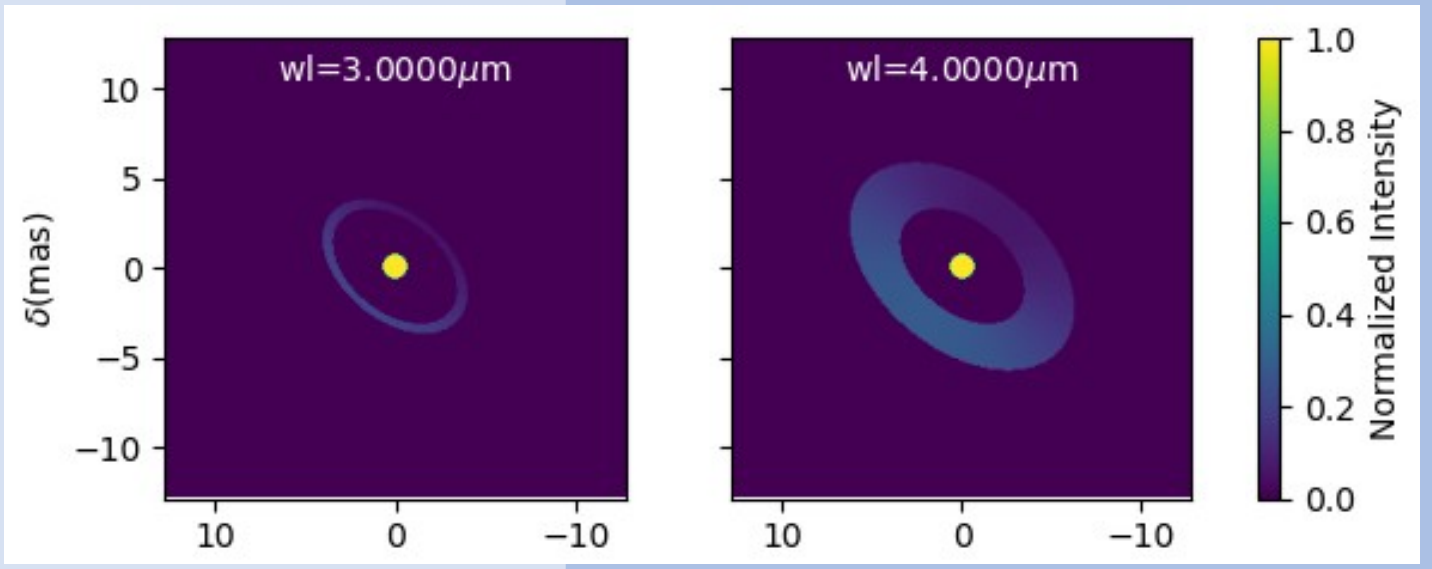
Example of simple model-fitting

```
import oimodeler as oim

ud = oim.oimUD(d=3,f=0.5,x=5,y=-5)
pt = oim.oimPt(f=1)
model = oim.oimModel(ud,pt)

ud.params['d'].set(min=0.01,max=20)
ud.params['x'].set(min=-50,max=50,free=True)
ud.params['y'].set(min=-50,max=50,free=True)
ud.params['f'].set(min=0.,max=10.)

pt.params['f']=oim.oimParamNorm(ud.params['f’])

fit = oim.oimFitterEmcee(files,model,nwalkers=20)
fit.prepare(init="random")
fit.run(nsteps=2000, progress=True)

median,err_l,err_u,err=fit.getResults(discard=1000)

fit.walkersPlot()
fit.cornerPlot()
fit.simulator.plot(["VIS2DATA","T3PHI"])
model.showModel(512,0.1)
```



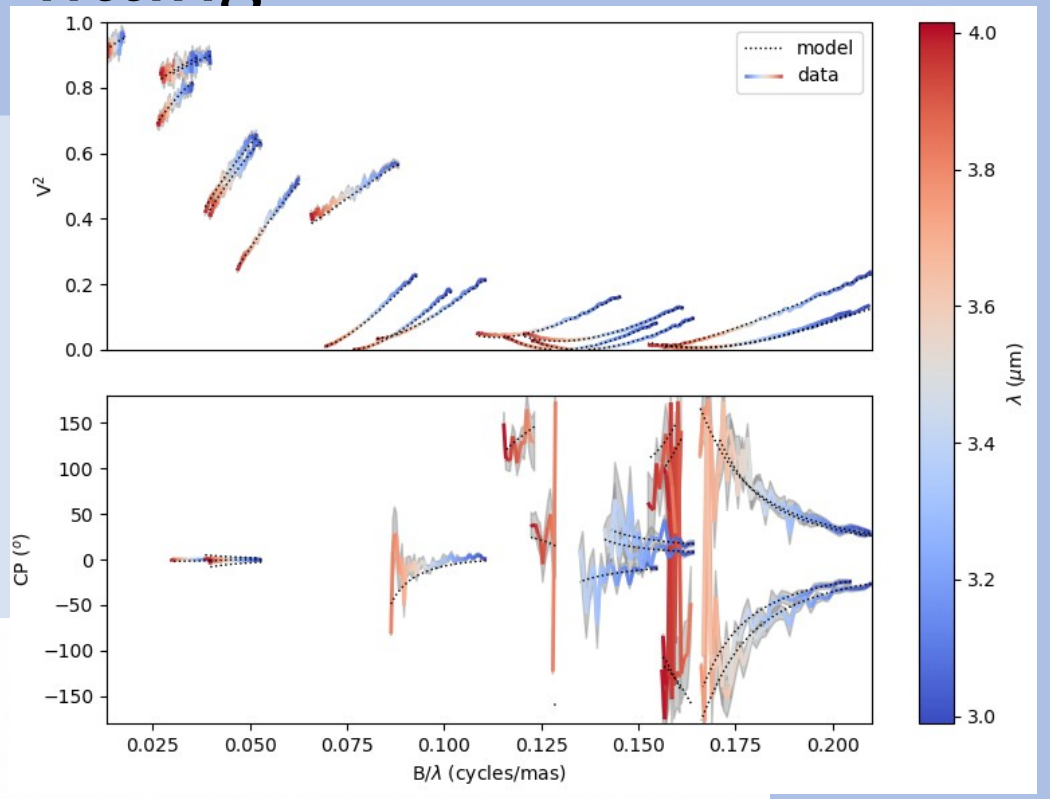
Chromatic model-fitting

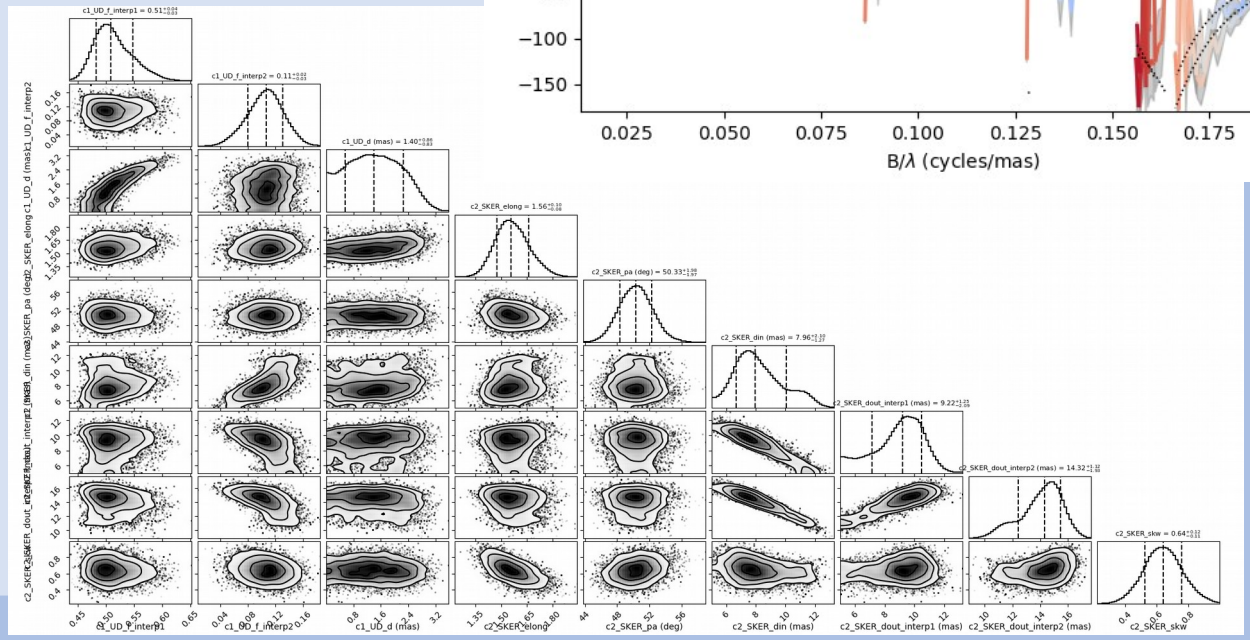
```
import oimodeler as oim

star=oim.oimUD(d=1,f=oim.oimInterp("wl",wl=[3e-6,4e-6],values=[0.5,0.1]))
disk=oim.oimESKRing(din=8,dout=oim.oimInterp("wl",wl=[3e-6,4e-6],values=[9,14]),elong=1.5,skw=0.8,pa=50)
disk.params["skwPa"]=oim.oimParamLinker(disk.params["pa"],"add",90)
disk.params['f']=oim.oimParamNorm(star.params['f'])
model=oim.oimModel(star,disk)

params=model.getFreeParameters()
params['c1_UD_f_interp1'].set(min=0.0,max=1)
params['c1_UD_f_interp2'].set(min=0.0,max=1)
params['c1_UD_d'].set(min=0,max=5,free=True)
params['c2_SKER_pa'].set(min=0.,max=180)
params['c2_SKER_elong'].set(min=1,max=3)
params['c2_SKER_din'].set(min=5,max=20.)
params['c2_SKER_skw'].set(min=0,max=1.)
params['c2_SKER_dout_interp1'].set(min=5.,max=30.)
params['c2_SKER_dout_interp2'].set(min=5.,max=30.)

fit=oim.oimFitterEmcee(files,model,nwalkers=30)
fit.prepare(init="random")
fit.run(nsteps=2000,progress=True)

figWalkers,axeWalkers=fit.walkersPlot()
figCorner,axeCorner=fit.cornerPlot(discard=1000)
median,err_l,err_u,err=fit.getResults(mode='median',discard=1000)
figSim,axSim=fit.simulator.plot(["VIS2DATA","T3PHI"])
figImg,axImg,im=model.showModel(256,0.1,wl=[wl[0],wl[-1]])
```



Chromatic model-fitting

```
import oimodeler as oim

star=oim.oimUD(d=1,f=oim.oimInterp("wl",wl=[3e-6,4e-6],values=[0.5,0.1]))
disk=oim.oimESKRing(din=8,dout=oim.oimInterp("wl",wl=[3e-6,4e-6],values=[9,14]),elong=1.5,skw=0.8,pa=50)
disk.params["skwPa"]=oim.oimParamLinker(disk.params["pa"],"add",90)
disk.params['f']=oim.oimParamNorm(star.params['f'])
model=oim.oimModel(star,disk)

params=model.getFreeParameters()
params['c1_UD_f_interp1'].set(min=0.0,max=1)
params['c1_UD_f_interp2'].set(min=0.0,max=1)
params['c1_UD_d'].set(min=0,max=5,free=True)
params['c2_SKER_pa'].set(min=0.,max=180)
params['c2_SKER_elong'].set(min=1,max=3)
params['c2_SKER_din'].set(min=5,max=20.)
params['c2_SKER_skw'].set(min=0,max=1.)
params['c2_SKER_dout_interp1'].set(min=5.,max=30.)
params['c2_SKER_dout_interp2'].set(min=5.,max=30.)

fit=oim.oimFitterEmcee(files,model,nwalkers=30)
fit.prepare(init="random")
fit.run(nsteps=2000,progress=True)

figWalkers,axeWalkers=fit.walkersPlot()
figCorner,axeCorner=fit.cornerPlot(discard=1000)
median,err_l,err_u,err=fit.getResults(mode='median',discard=1000)
figSim,axSim=fit.simulator.plot(["VIS2DATA","T3PHI"])
figImg,axImg,im=model.showModel(256,0.1,wl=[wl[0],wl[-1]])
```





Creating new Fourier components

```
class oimBox(oim.oimComponentFourier):
    name="2D Box"
    shortname = "BOX“

    def __init__(self,**kwargs):
        super().__init__(**kwargs)
        self.params["dx"]=oim.oimParam(name="dx", value=1,description="Size in x",unit=u.mas)
        self.params["dy"]=oim.oimParam(name="dy", value=1,description="Size in y",unit=u.mas)
        self._eval(**kwargs)

    def _visFunction(self,ucoord,vcoord,rho,wl,t):
        x=self.params["dx"](wl,t)*self.params["dx"].unit.to(u.rad)*ucoord
        y=self.params["dy"](wl,t)*self.params["dy"].unit.to(u.rad)*vcoord

        return np.sinc(x)*np.sinc(y)

    def _imageFunction(self,xx,yy,wl,t):
        return ((np.abs(xx)<=self.params["dx"](wl,t)/2) &
                (np.abs(yy)<=self.params["dy"](wl,t)/2)).astype(float)
```

Initialization function

- Call parent __init__
- Define parameters (oimParam)
- Call eval function

visibility function

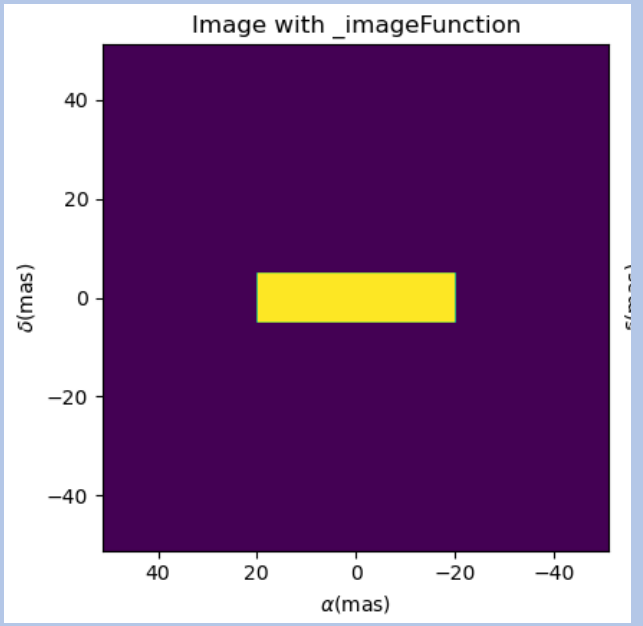
- formula as function of u and v or ρ (and optionally λ and t)

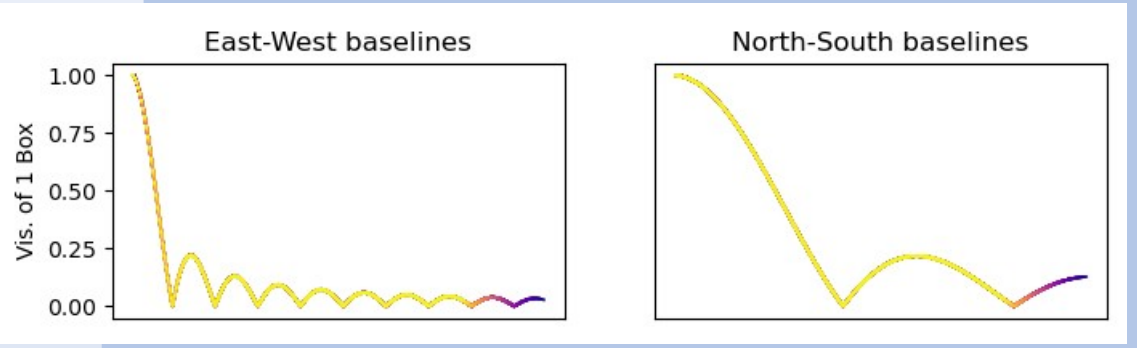
Image function

- formula as function of x and y (and optionally λ and t)

Creating new Fourier components

```
class oimBox(oim.oimComponentFourier):
    name="2D Box"
    shortname = "BOX“

    def __init__(self,**kwargs):
        super().__init__(**kwargs)
        self.params["dx"]=oim.oimParam(name="dx", value=1,description="Size in x",unit=u.mas)
        self.params["dy"]=oim.oimParam(name="dy", value=1,description="Size in y",unit=u.mas)
        self._eval(**kwargs)

    def _visFunction(self,ucoord,vcoord,rho,wl,t):
        x=self.params["dx"](wl,t)*self.params["dx"].unit.to(u.rad)*ucoord
        y=self.params["dy"](wl,t)*self.params["dy"].unit.to(u.rad)*vcoord

        return np.sinc(x)*np.sinc(y)

    def _imageFunction(self,xx,yy,wl,t):
        return ((np.abs(xx)<=self.params["dx"](wl,t)/2) &
                (np.abs(yy)<=self.params["dy"](wl,t)/2)).astype(float)
```



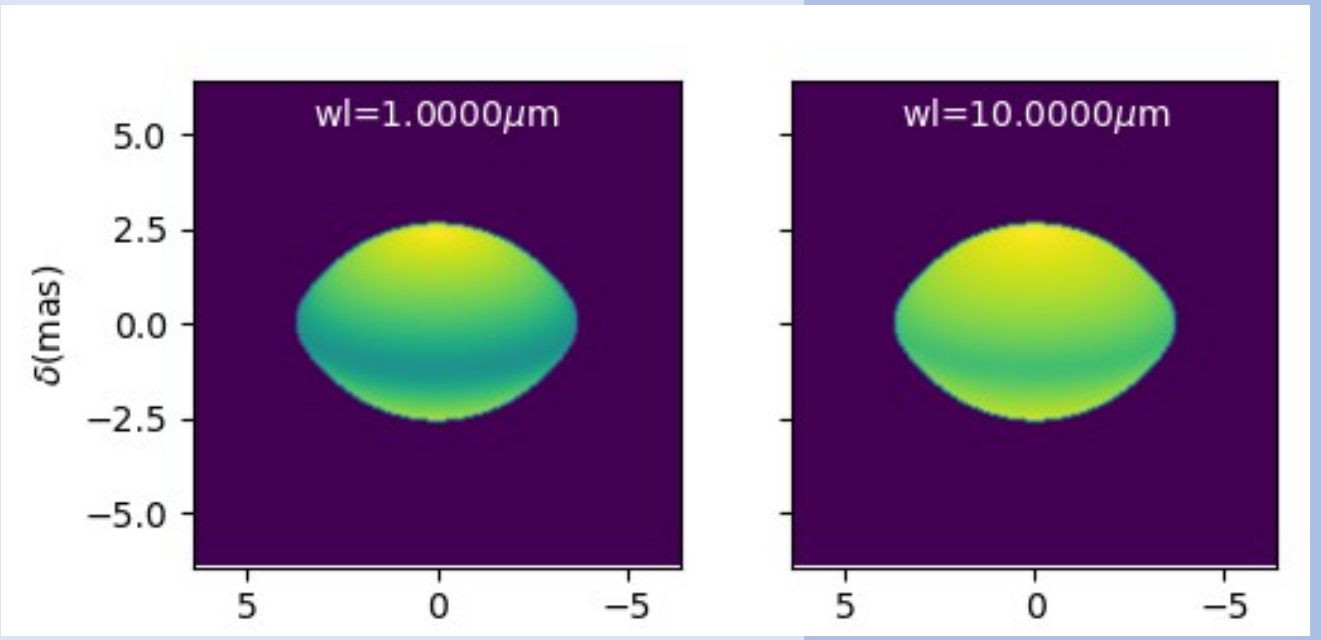



```
b1=oimBox(dx=40,dy=10)
m1=oim.oimModel(b1)
```

```
m1.showModel(512,0.2,axe=ax[0],colorbar=False)
vis=np.abs(m.getComplexCoherentFlux(spfx,spfy)
```

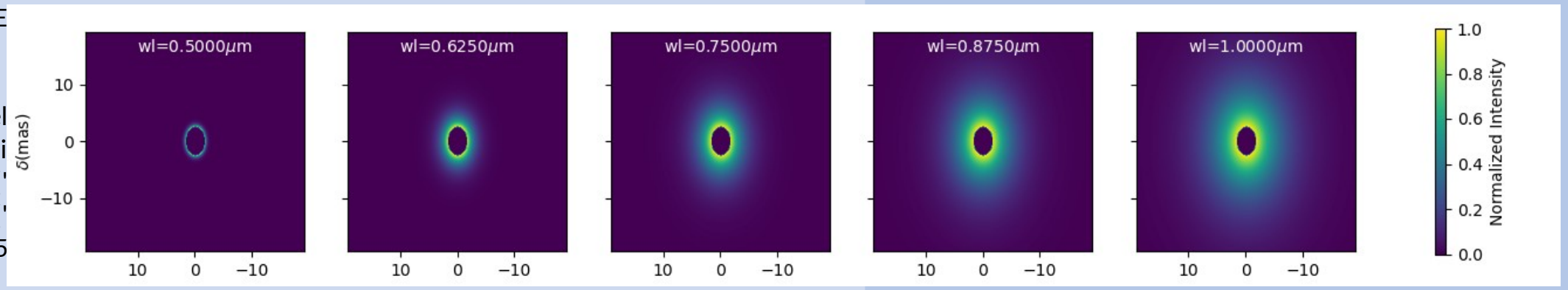
Creating new Image components (external code)

```
class oimFastRotator(oim.oimComponentImage):
    name="Fast Rotator"
    shortname="FRot"
    def __init__(self,**kwargs):
        super(). __init__(**kwargs)
        self.params["incl"]=oim.oimParam(name="incl",value=0,description="Inclination angle",unit=units.deg)
        self.params["rot"]=oim.oimParam(name="rot",value=0,description="Rotation Rate",unit=units.one)
        self.params["Tpole"]=oim.oimParam(name="Tpole",value=20000,description="Polar Temperature",unit=units.K)
        self.params["dpole"]=oim.oimParam(name="dplot",value=1,description="Polar diameter",unit=units.mas)
        self.params["beta"]=oim.oimParam(name="beta", value=0.25,description="Gravity Darkening Exponent",unit=units.one)
        self._t = np.array([0])
        self._wl = np.linspace(0.5e-6,15e-6,num=10)
        self._eval(**kwargs)

    def _internalImage(self):
        dim=self.params["dim"].value
        incl=self.params["incl"].value
        rot=self.params["rot"].value
        Tpole=self.params["Tpole"].value
        dpole=self.params["dpole"].value
        beta=self.params["beta"].value

        im=fastRotator(dim,1.5,incl,rot,Tpole,self._wl,beta=beta)
        im=np.tile(np.moveaxis(im,-1,0)[None,:,:,:],(1,1,1,1))
        self._pixSize=1.5*dpole/dim*units.mas.to(units.rad)
        return im
```

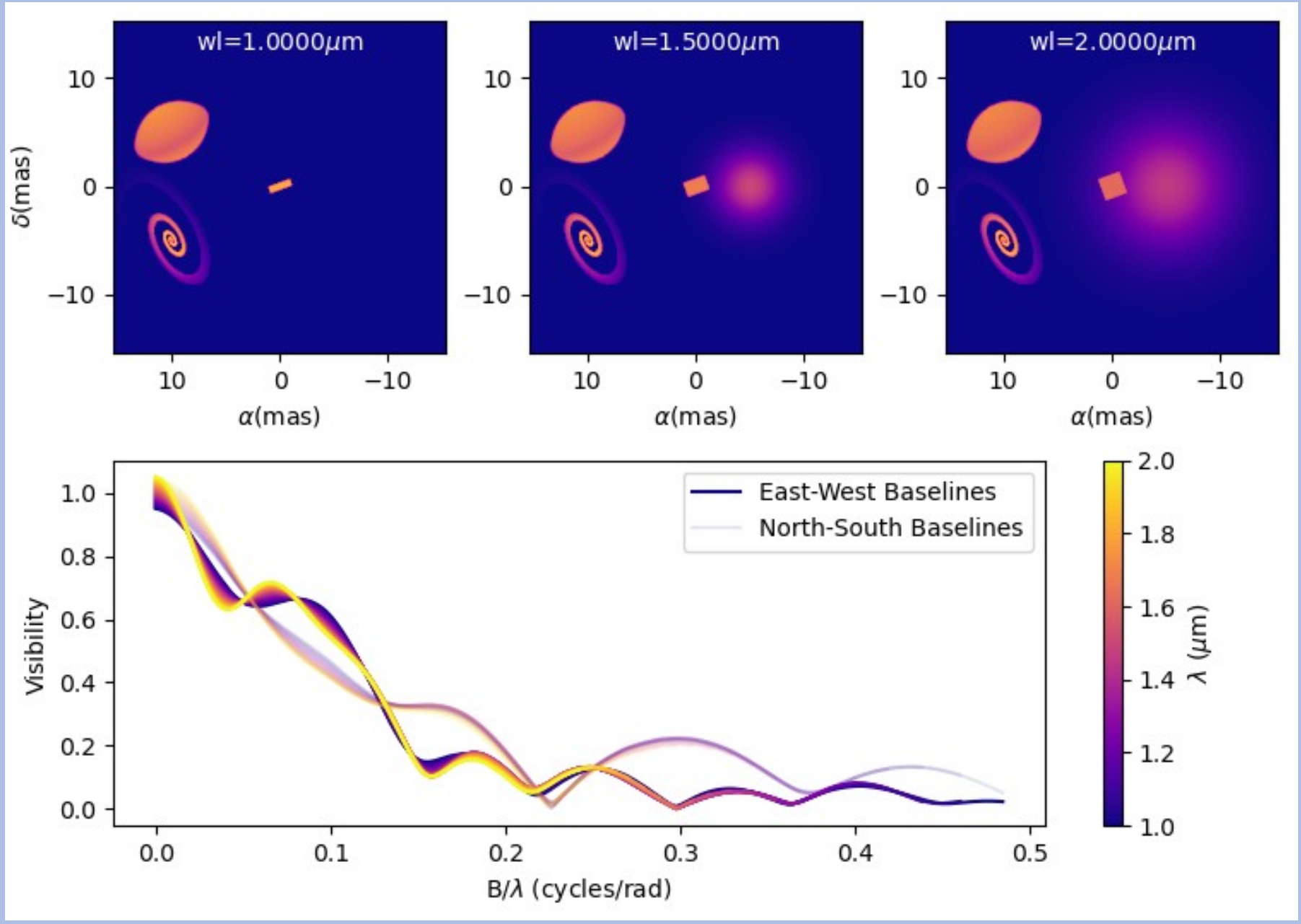


Creating new Radial Profile components (analytical)

```
class oimExpRing(oim.oimComponentRadialProfile):

    name="A ring with a descreasing exponential profil"
    shortname = "E
    elliptic=True

    def __init__(sel
        super().__ini
        self.params['
        self.params['
        self._dim=25

        self._t = np.array([0])
        self._wl = np.array([0.5e-6,1e-6])
        self._r = np.arange(0, self._dim)*0.05 #in mas
        self._eval(**kwargs)

    def _radialProfileFunction(self,r,wl,t):
        r0=self.params["d"](wl,t)/2
        fwhm=self.params["fwhm"](wl,t)
        return np.nan_to_num((r>r0)*np.exp(-0.692*np.divide(r-r0,fwhm)),nan=0)
```



Combining components



TODO in 2023 …

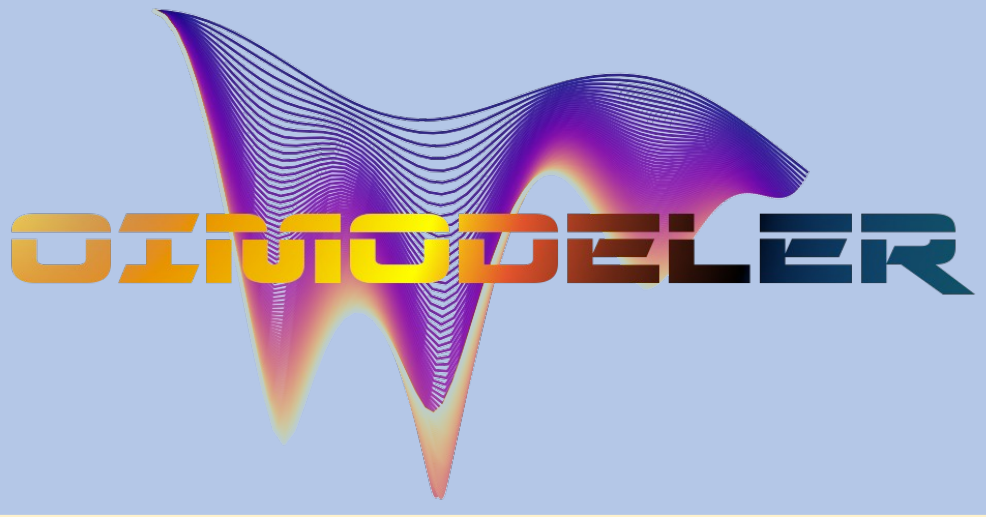
- Implement missing basic features:
  - Create components from fits files and grid
  - Saving (model, fit)
  - Flux normalization (from 1 or ad-hoc to Jy)
  - Photometric and spectroscopic data

- Add a few advanced features
  - models (rot. disk, DISCO+, AMHRA, grids?)
  - “intelligent” sampling for image based models
  - fitters (options, λ-by-λ, lmfit, chain, external constraints…)
  - filters (wl shift, smoothing, binning…)

- Extensive test of the code
  - Unitary tests for all models and features
  - Tests Simulated data (chromatic + time-dependent)
  - Real data from all known instruments

- Start working on optimization
  - Parallelization (model & fitter)
  - FFT & Hankel algorithms
  - Data optimization

- Documentation & project management (GIT…)



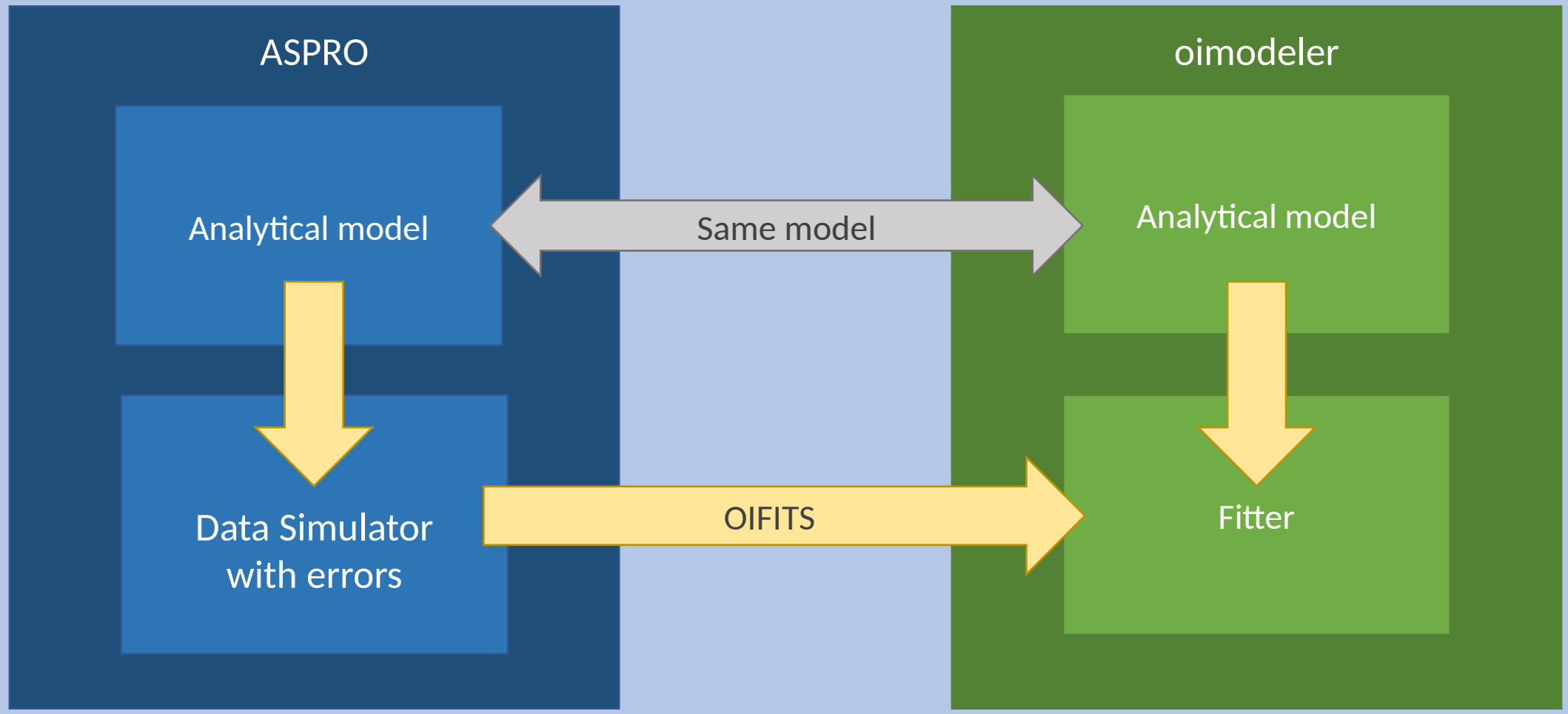
Oimodeler Team is building up…

Anthony Meilland, Jozsef Varga, Alexis Matter, Marten Scheuck, Armando Domiciano de Souza …

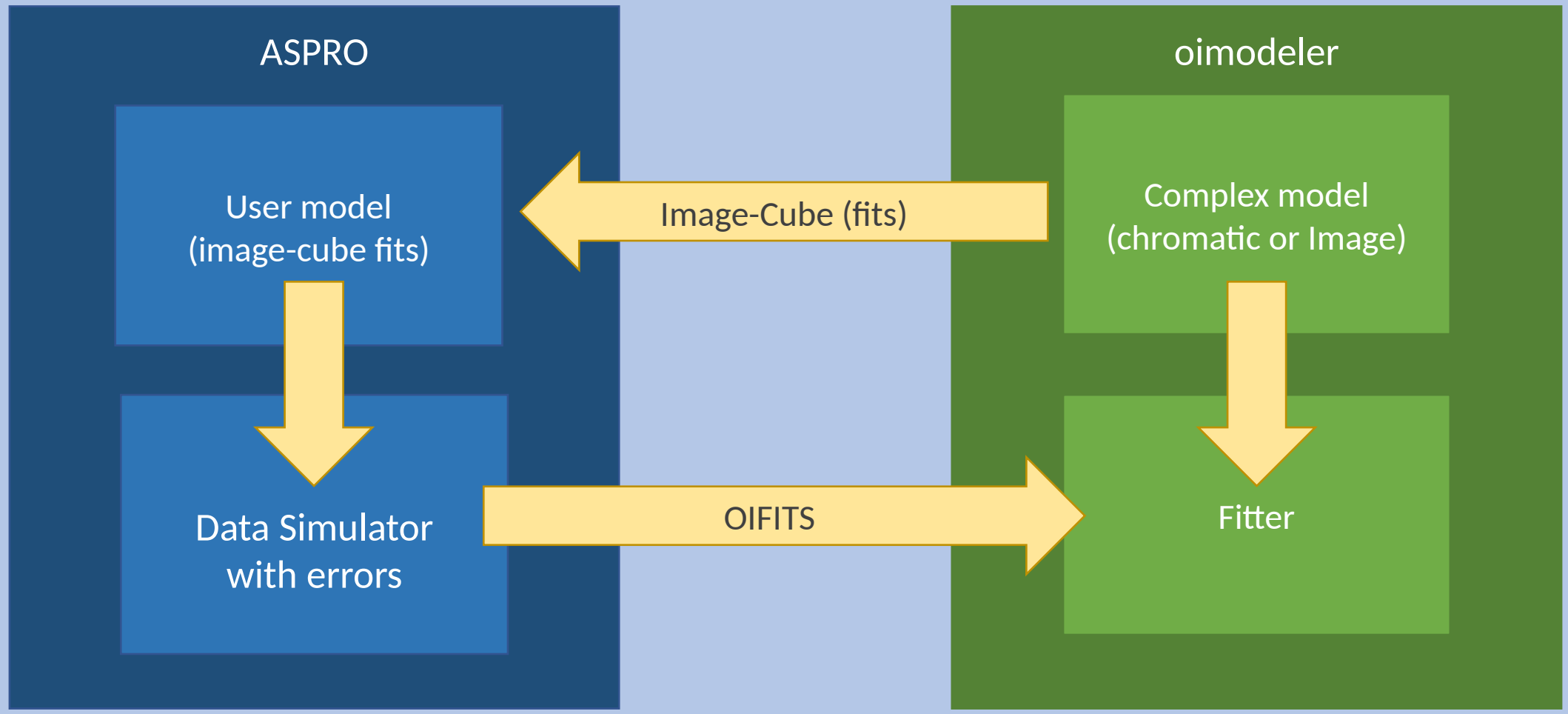
master and phD students in Nice, Leiden and Heildelberg …

+ ANR MASSIF

How did I produce simulated data for comparison?

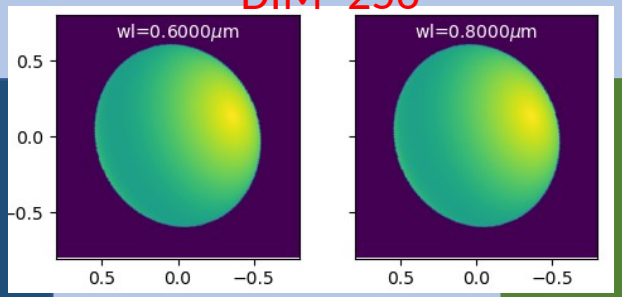


How did I produce simulated data for comparison?



Model-fitting with a image-plan model

DIM=256



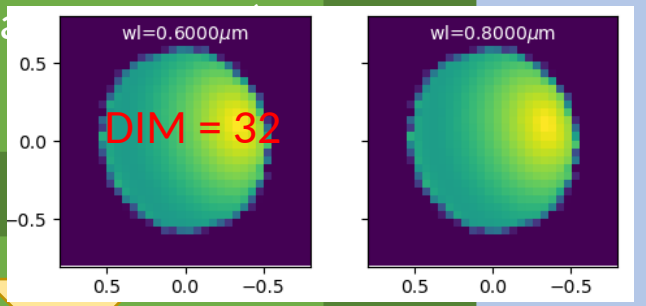
ASPRO

User model
(image-cube fits)

Image-Cube (fits)

oimodeler

Complex model
(chroma



Data Simulator
with errors

OIFITS

Fitter

2000 steps
20 walkers
40000 models
13min
(20ms/model)