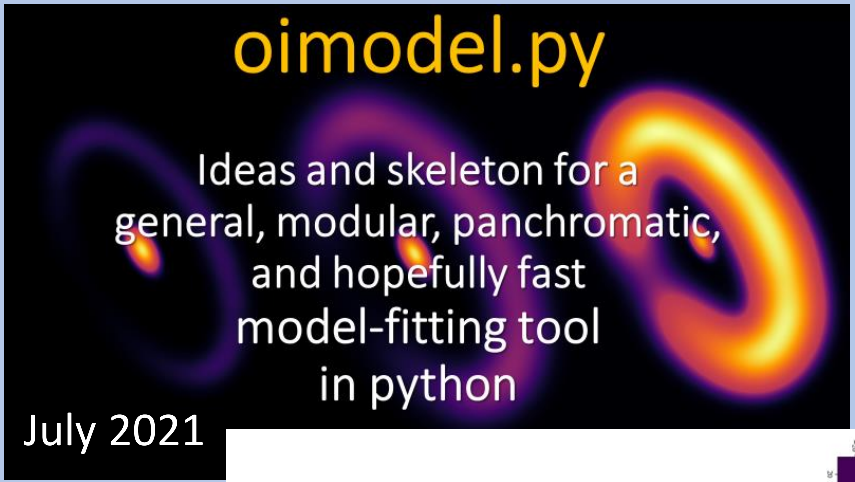
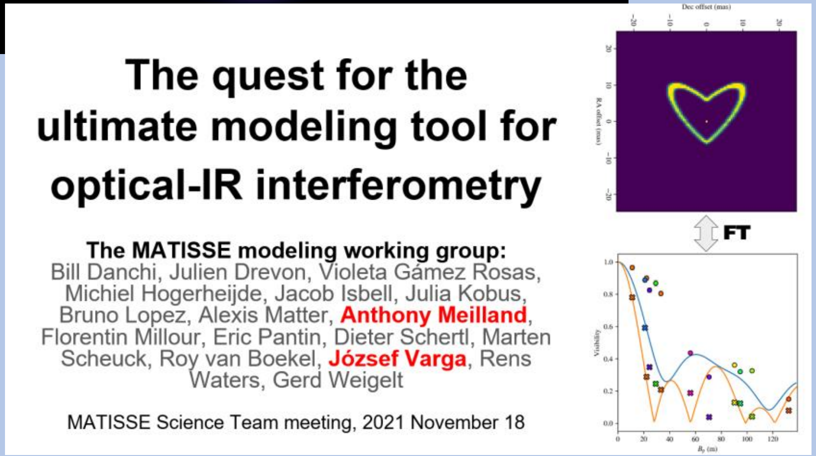
OIMODELER

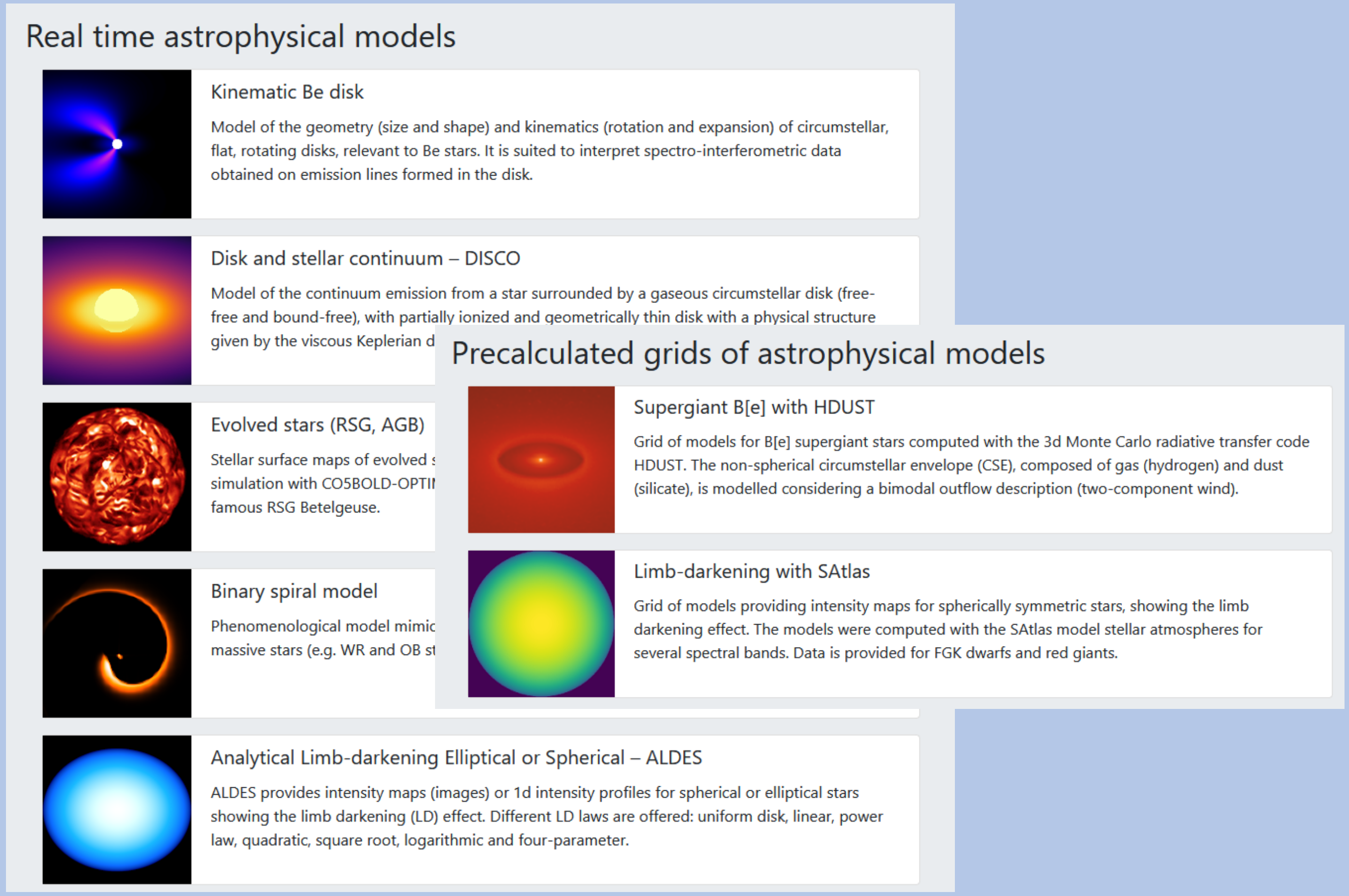
A MODULAR MODELLING SOFTWARE FOR OPTICAL INTERFEROMETRY

Context of the project (2021)



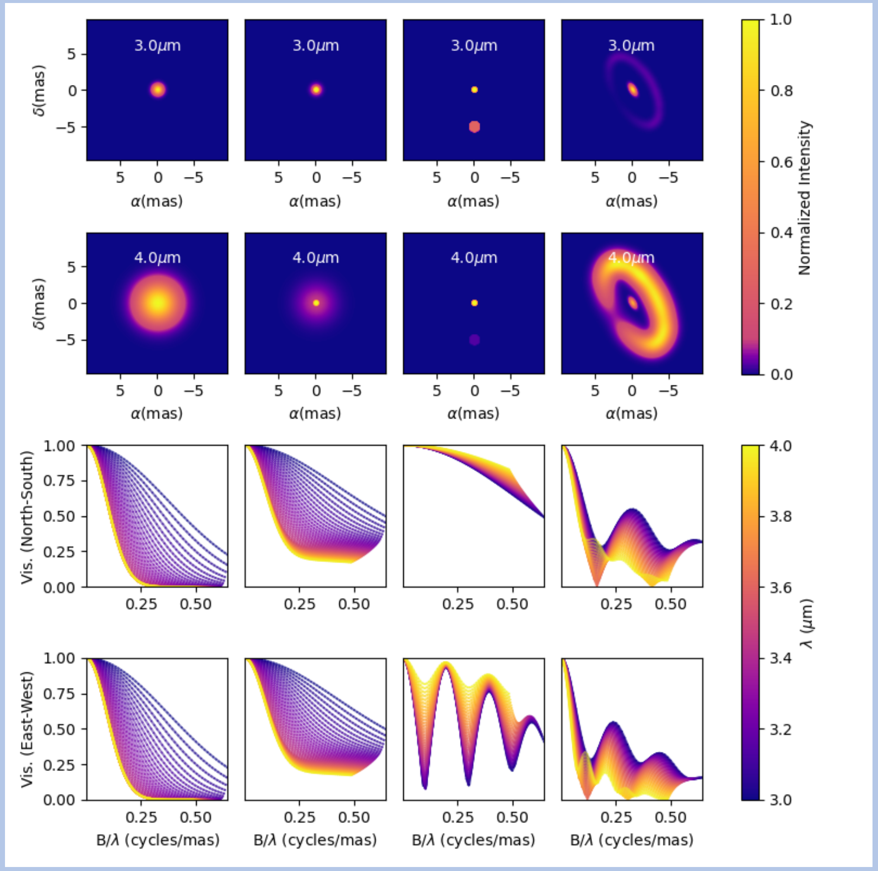


MATISSE group



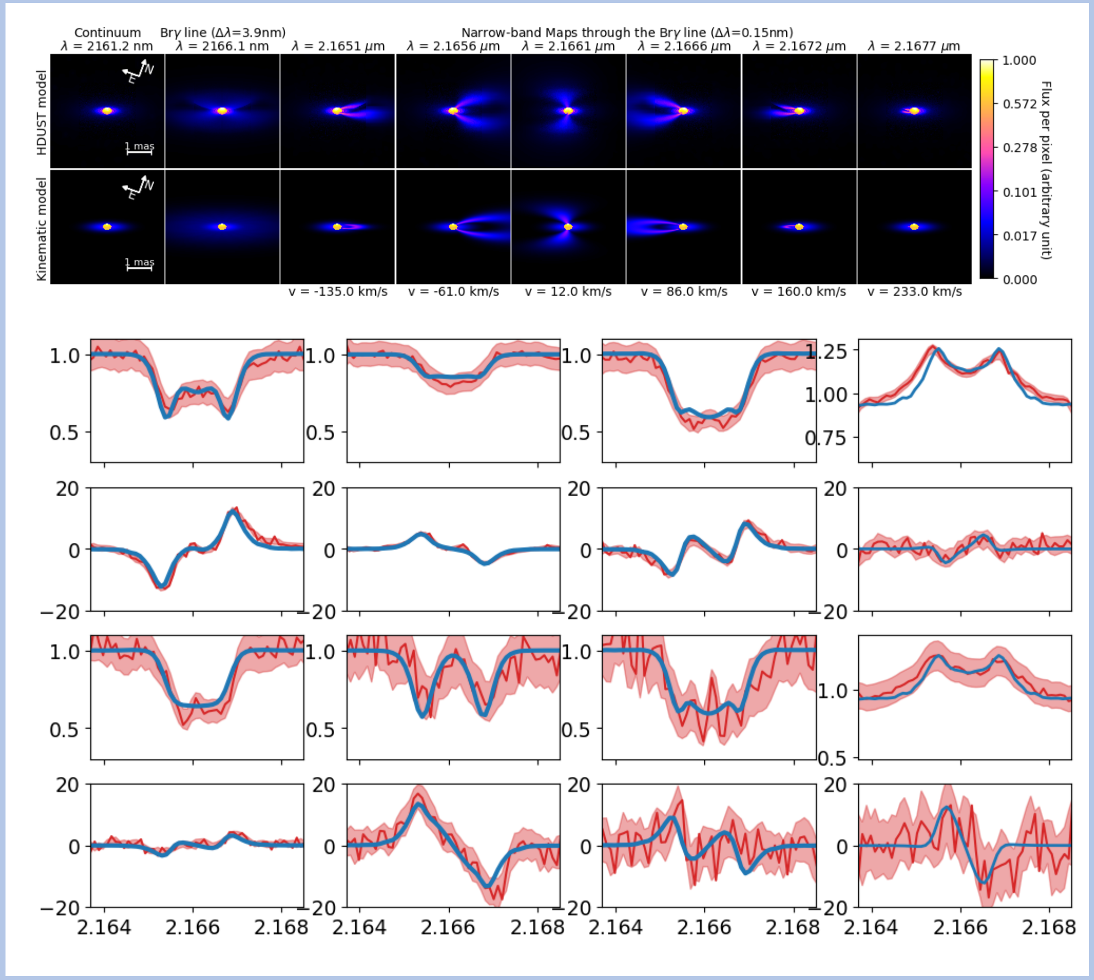
JMMC/AMHRA

Context of the project



Oimodel.py (2021)

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

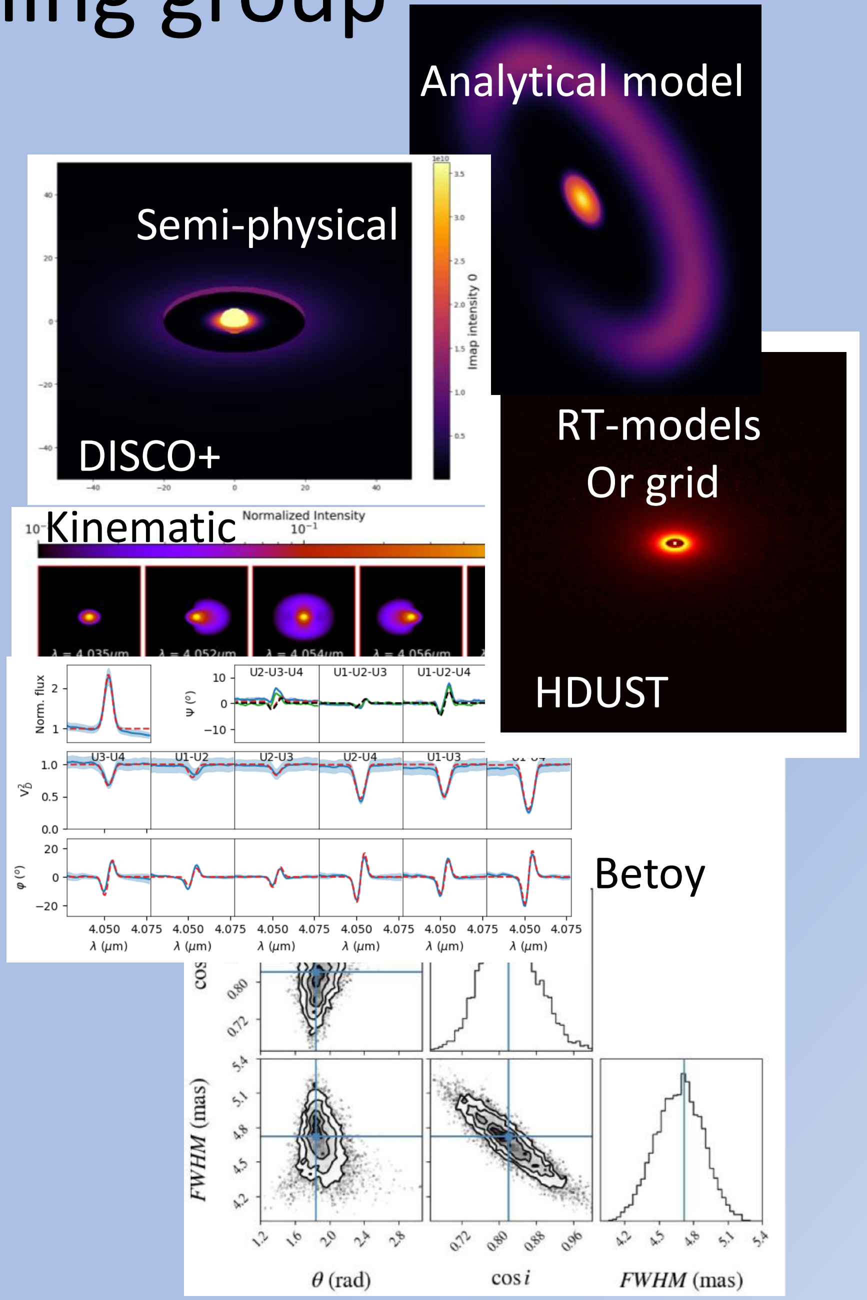


Cubetools.py (2017)

Computing interferometric measurements from chromatic image-cubes

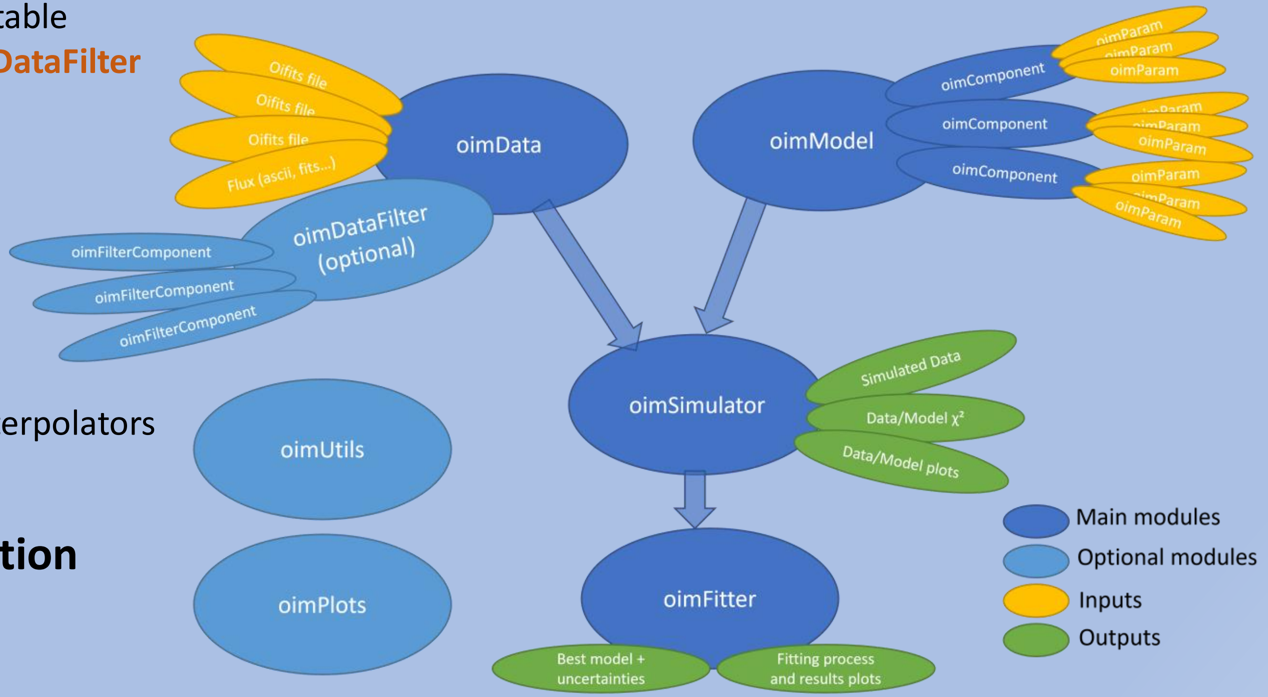
Discussion in MATISSE modelling group
By the end of 2021

- Python3
- Modularity and flexibility
  - Analytical models in Fourier-plan (LITPro Like)
  - 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 (interpolated, temperature based…)
  - Chromatic components (such as temperature gradient, binary orbit)
  - Kinematics through line models
- Ability to use interferometric data from all instruments (OIFITS2 format)
- 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)



oimodeler coding started in 2022

- **oimData: Wrapper for oifits data is astropy.io.fits format**
  - Contains both the oifits data and “optimized” vectors of data and coordinates (u,v,wl,t)
  - Import flux data (from ascii) : oimFluxData ⇔ conversion to OI_FLUX table
  - Possibility of filtering/modifying data (cut, smooth, bin, reflag…): oimDataFilter



- **oimModel: “lego” Model class**
  - oimComponents: components or “bricks” of the model
    - Fourier-based analytical formulas
    - 2D-image-based: computed using FFT
    - 1D-image-based: using Hankel transform (experimental)
  - oimParam: components parameters
    - Can be chromatic and/or time-dependent using parameter interpolators
    - Can be linked together by mathematical formula

- **oimSimulator: simulate data from model and χ^2 computation**
  - Can simulate any kind of oifits2 data for all instruments:
    VIS2DATA, VISAMP (absolute, differential, correlated flux)
    VISPHI (absolute and differential), T3AMP & T3PHI, FLUXDATA
  - Two modes: fast (only χ^2 for model-fitting), slow (simulated data for plotting and saving)

- **oimFitter: Model fitting class(es)**
  - Currently: only an emcee-based implemented: oimFitterEmcee (MCMC)
  - Use the oimSimulator for χ^2 computation
  - Minimized χ^2 and give best (or median) parameters and uncertainties (emcee “style”)

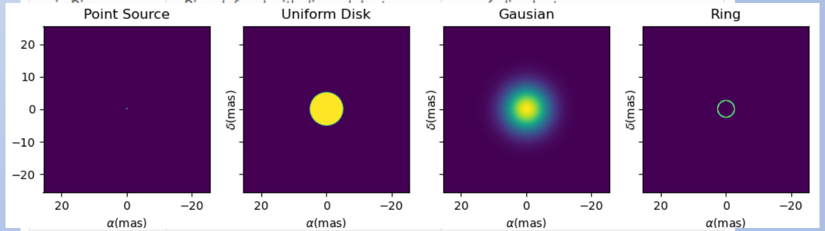
- **oimUtils & oimPlots : Helper classes**

astRotator
oimComponentFitsImage

Example Fourier and image components

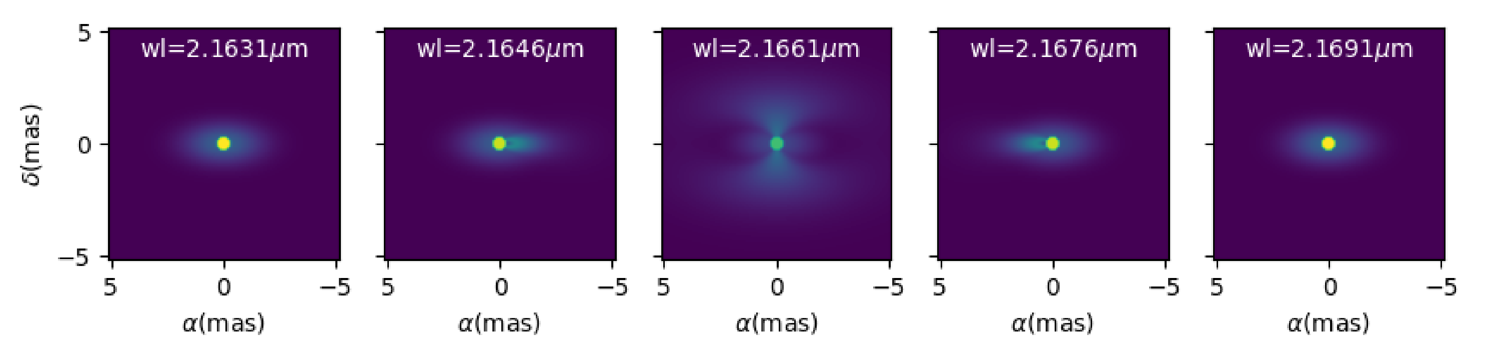
Components defined in Fourier plan

Class	Description	Parameters
oimPt	Point source	x, y, f
oimBackground	Background	x, y, f
oimUD	Uniform Disk	x, y, f, d
oimEllipse	Uniform Ellipse	x, y, f, d, pa, elong
oimGauss	Gaussian Disk	x, y, f, fwhm
oimEGauss	Elliptical Gaussian Disk	x, y, f, fwhm, pa, elong
oimIRing	Infinitesimal Ring	x, y, f, d
oimEIRing	Elliptical Infinitesimal Ring	x, y, f, d, pa, elong
oimESKIRing	Skewed Infinitesimal Elliptical Ring	x, y, f, d, skw, skwPa, pa, elong
oimPowerLDD	Power Law Limb Darkened Disk	x, y, f, d, a
oimSqrtLDD	Squared Root Limb Darkened Disk	x, y, f, d, a1, a2
oimLorentz	Pseudo-Lorenztian	x, y, fwhm
oimELorentz	Elliptical Pseudo-Lorenztian	x, y, f, fwhm, pa, elong
oimConvolutor	Convolution between 2 components	Parameters from the 2 components

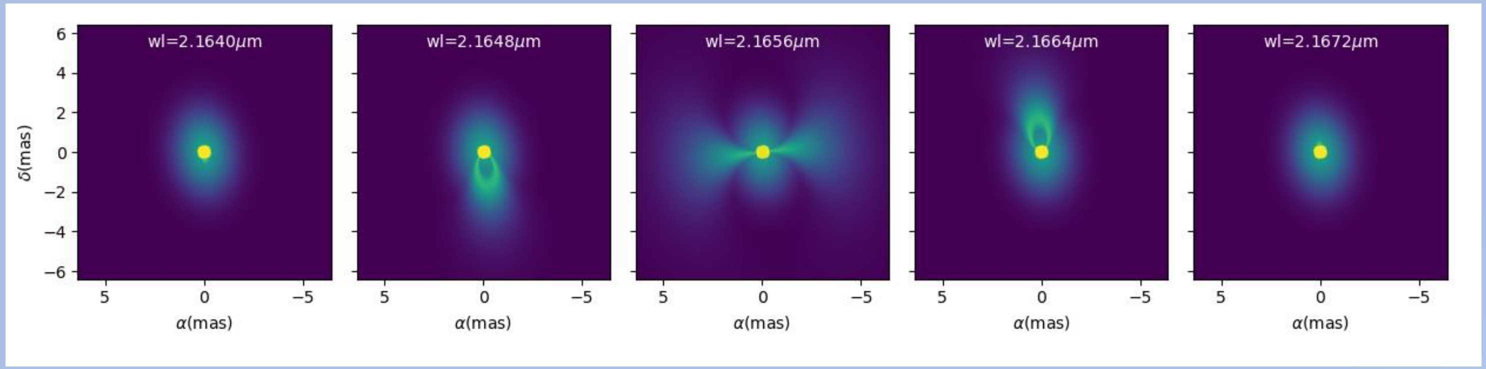


oimComponentFitsImage (2D-image-plan)

with AMHRA Kinematic Be disk image

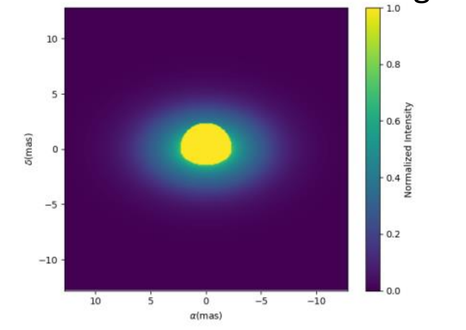


oimKinematicDisk (2D-image-plan)

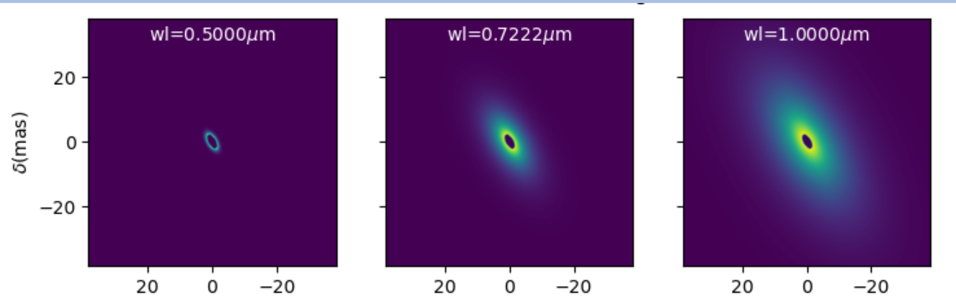


oimComponentFitsImage (2D-image-plan)

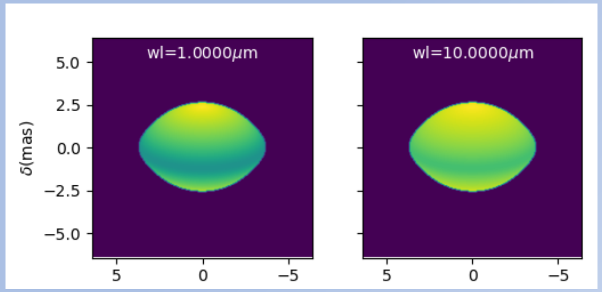
with AMHRA DISCO image



oimRadialProfile (1D-image-plan)

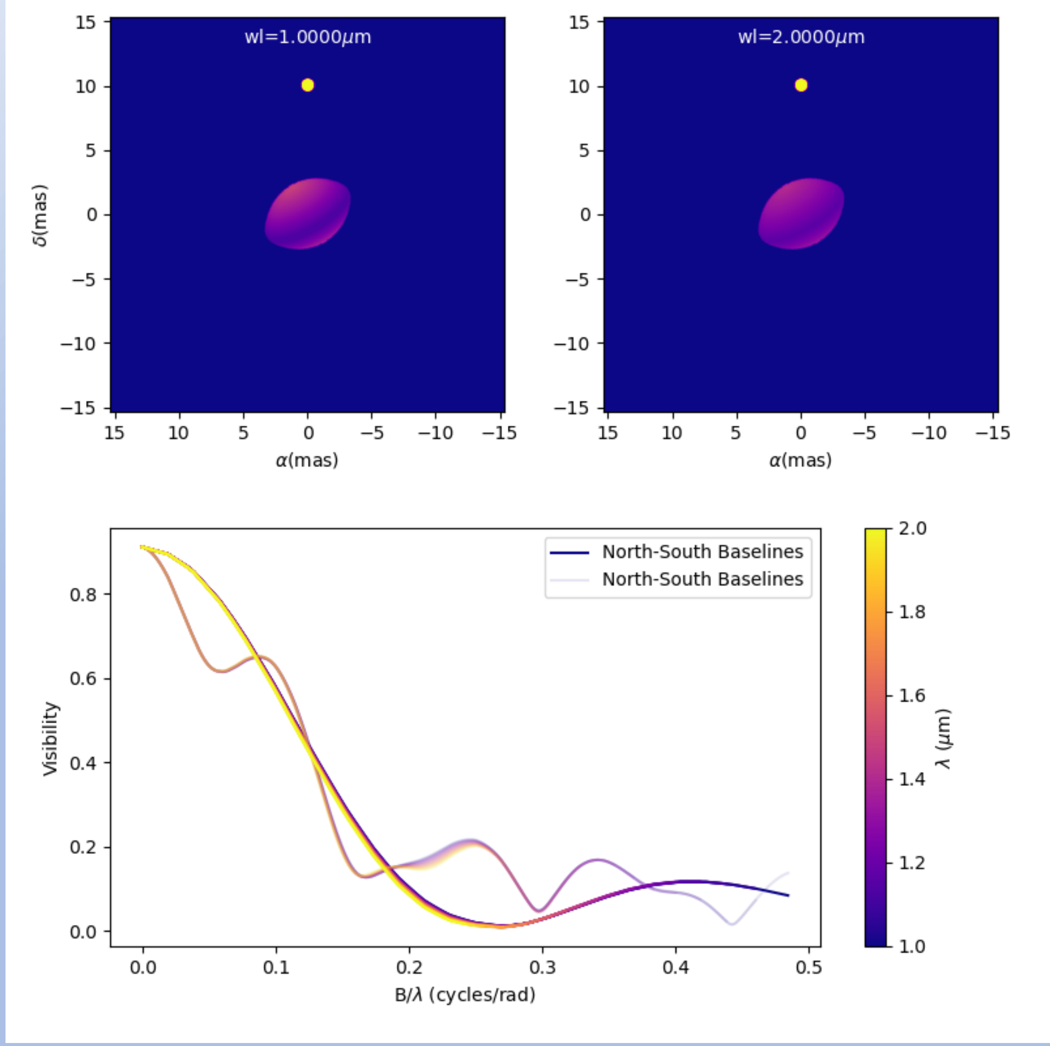


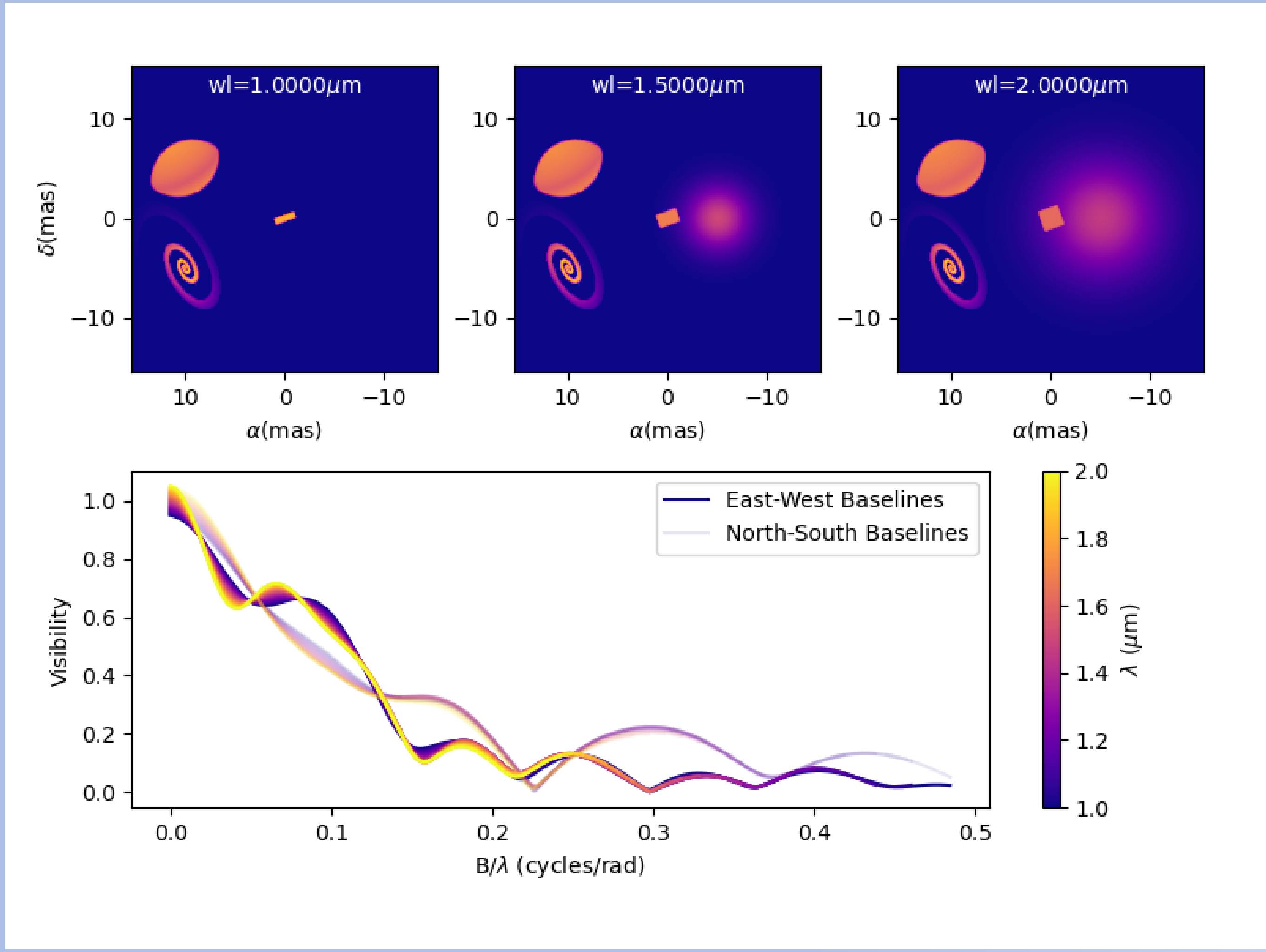
oimFastRotator (2D-image-plan)



Composite models

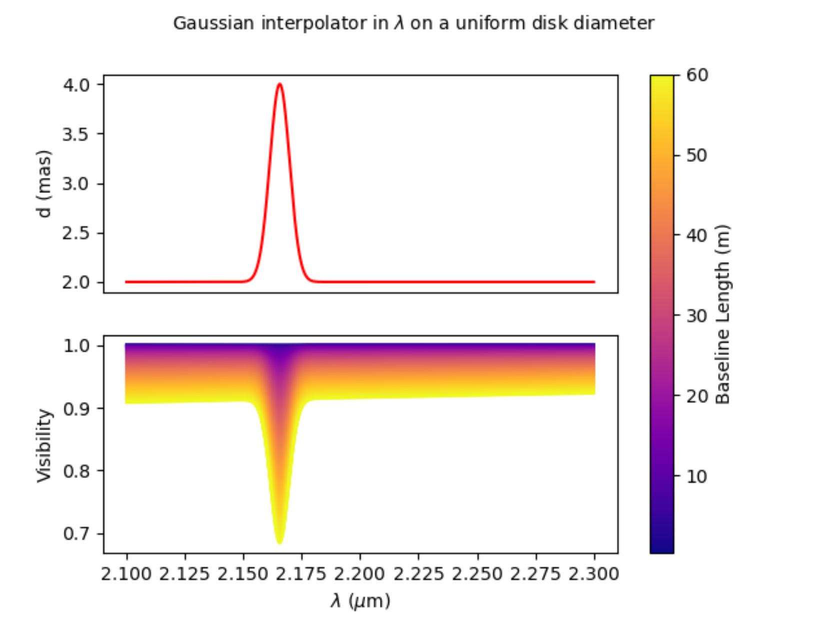
with components defined in Image & Fourier plan

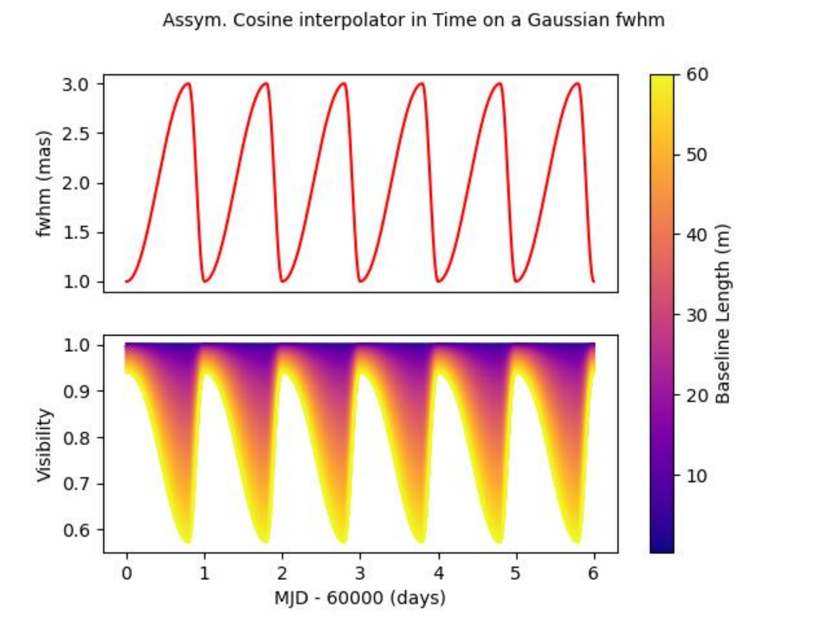




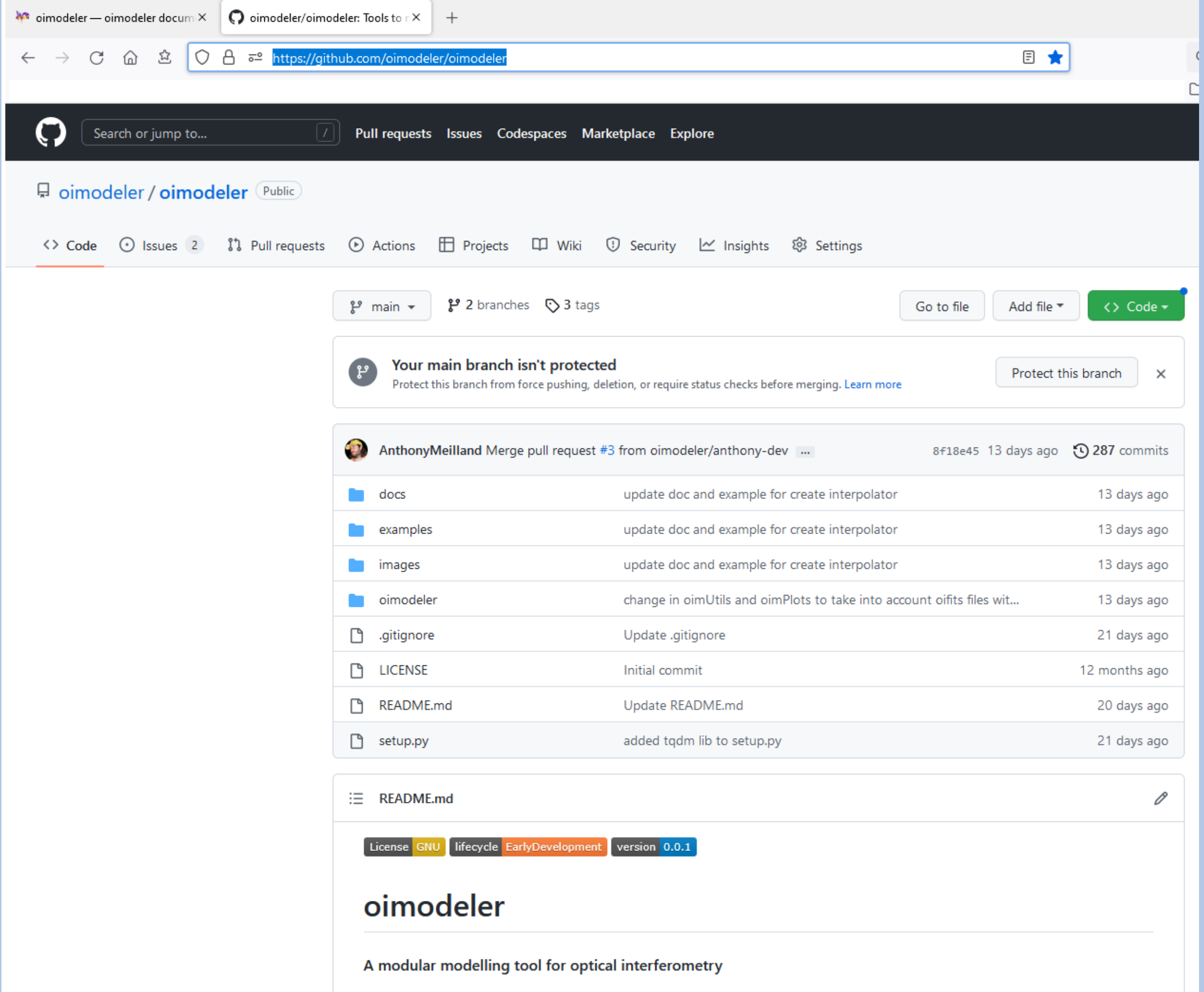
Chromaticity and time dependence

Class name	oimInterp macro	Description	Parameters
oimParamInterpolatorWl	"wl"	Interp between key wl	wl, values
oimParamInterpolatorTime	"time"	Interp between key time	mjd, values
oimParamGaussianWl	"GaussWl"	Gaussian in wl	val0, value, x0, fwhm
oimParamGaussianTime	"GaussTime"	Gaussian in time	val0, value, x0, fwhm
oimParamMultipleGaussianWl	"mGaussWl"	Multiple Gauss. in wl	val0 and value, x0, fwhm
oimParamMultipleGaussianTime	"mGaussTime"	Multiple Gauss. in time	val0 and value, x0, fwhm
oimParamCosineTime	"cosTime"	Asym. Cosine in Time	T0, P, values (optional x0)
oimParamPolynomialWl	"polyWl"	Polynomial in wl	coeffs
oimParamPolynomialTime	"polyTime"	Polynomial in time	coeffs



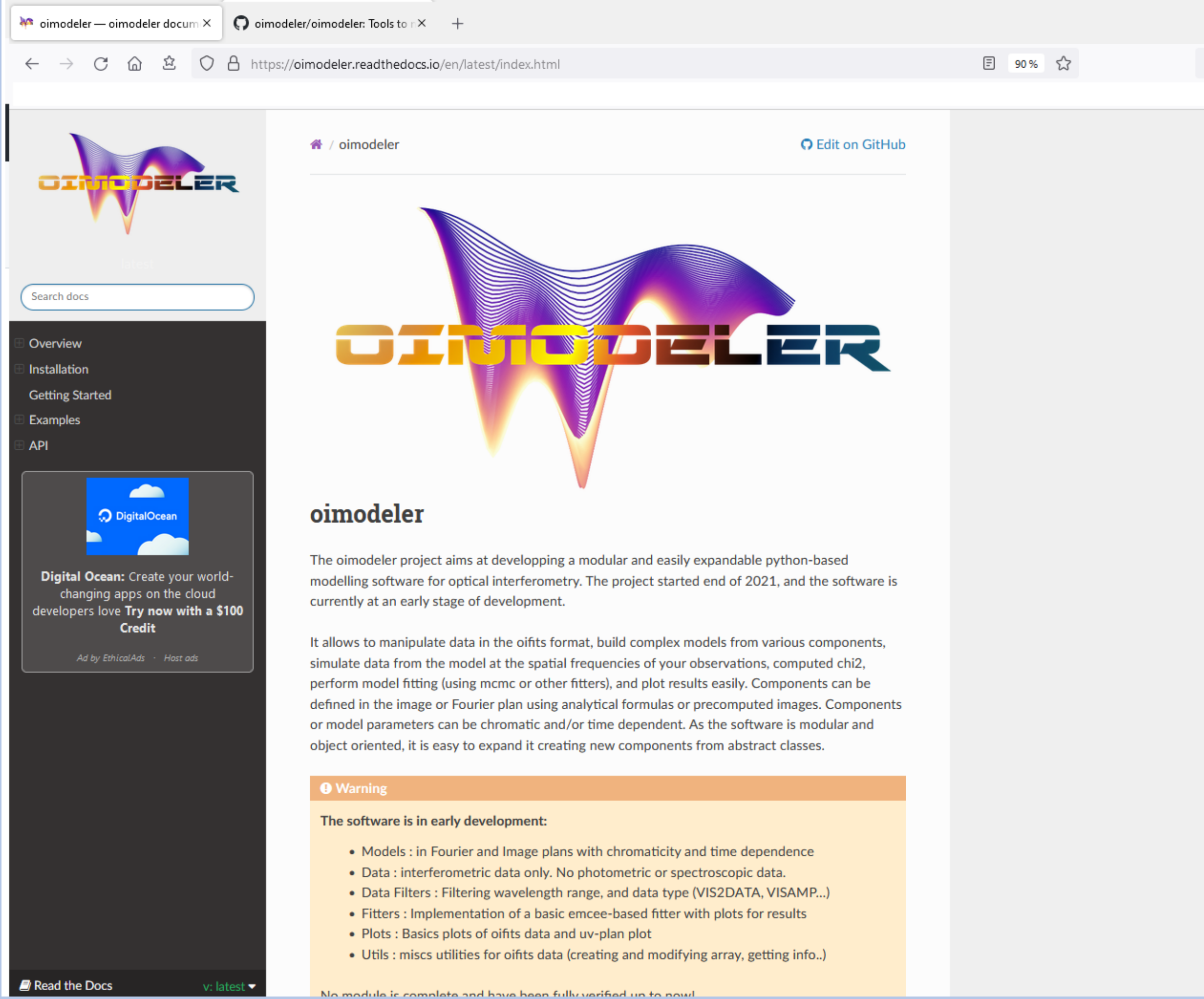


oimodeler on the web



Code available on github
+ automatic installation through pip
https://github.com/oimodeler

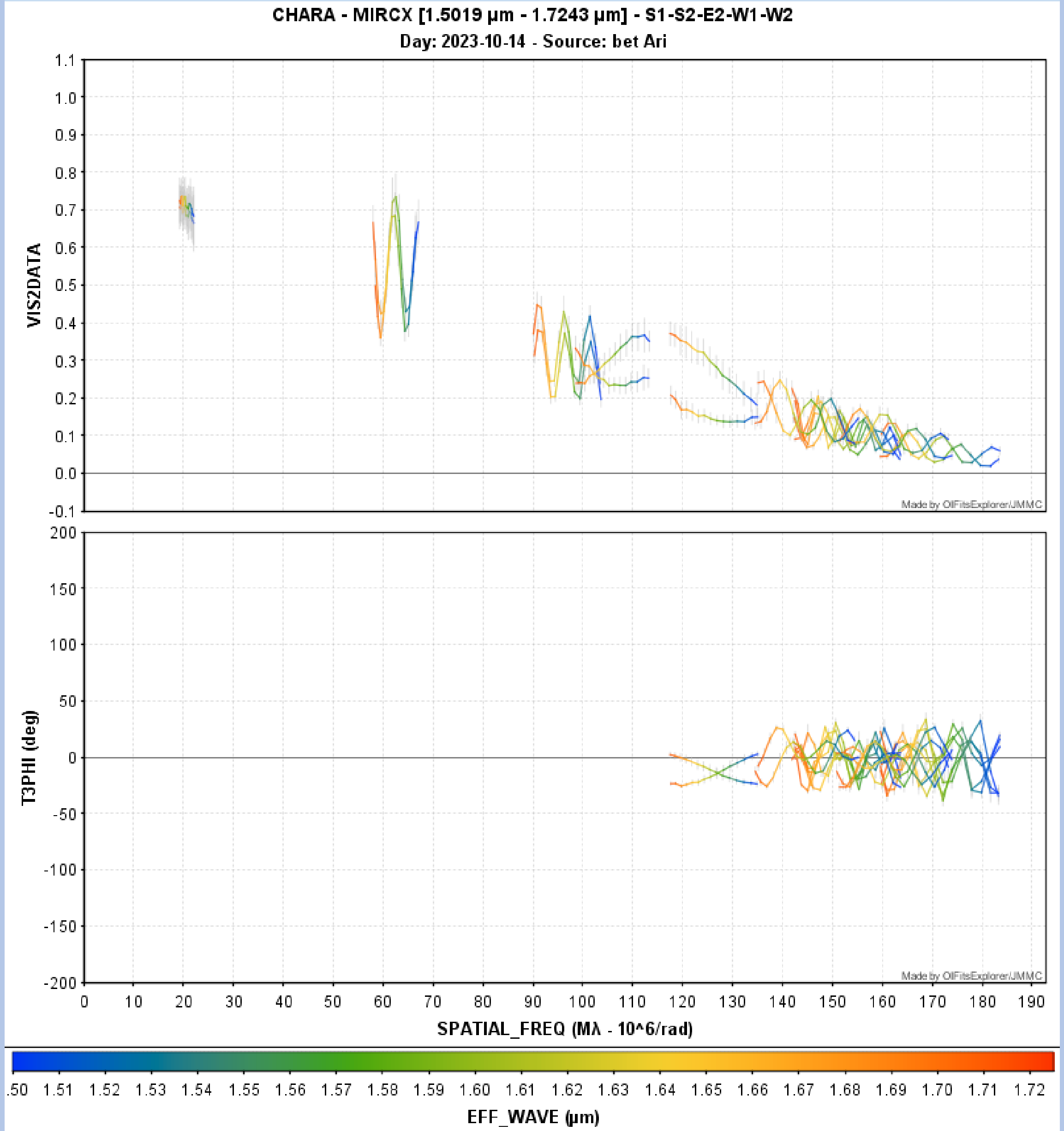
TIME CONSUMMING !



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

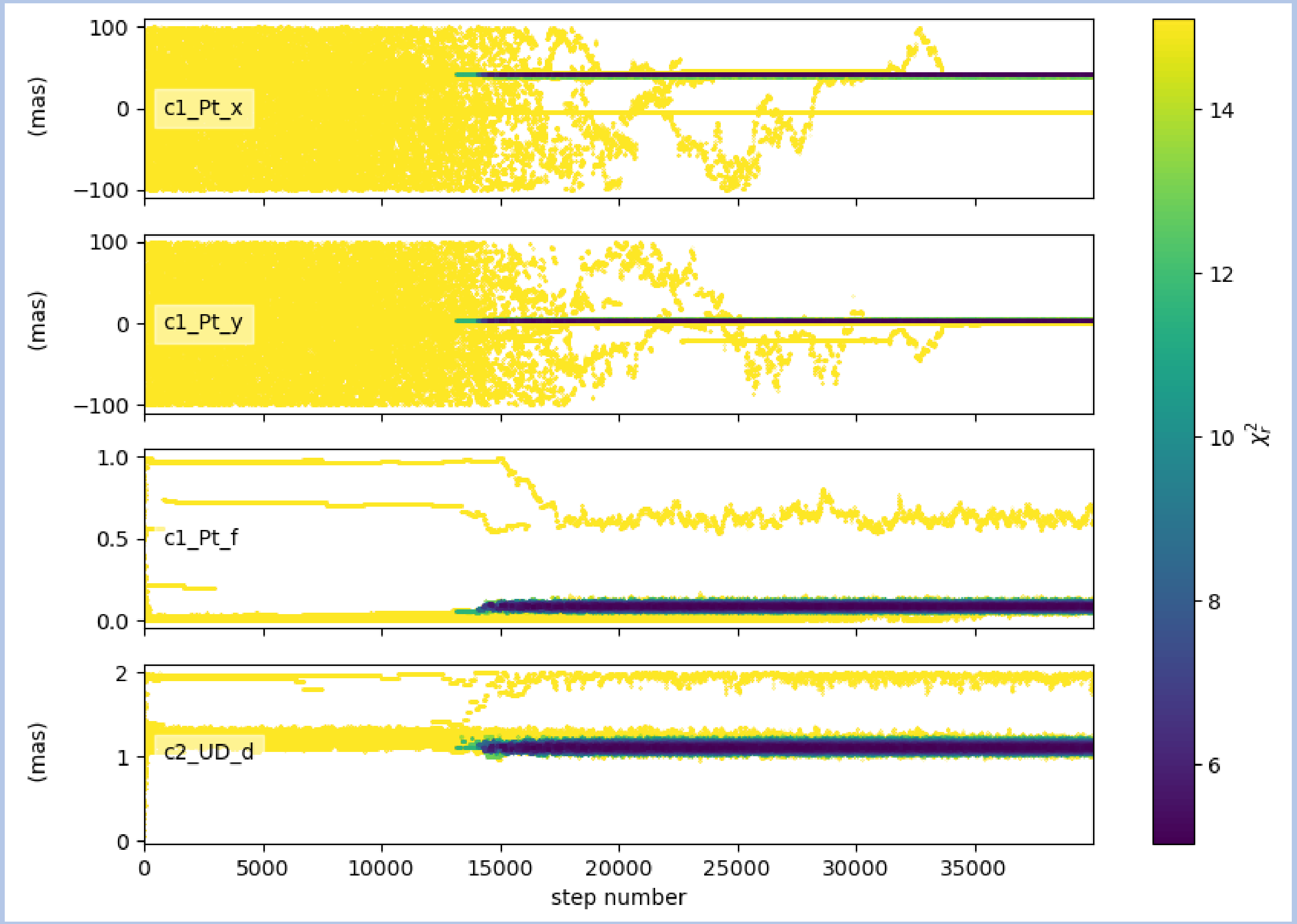
VERY VERY TIME CONSUMMING !

Example fitting real Data (MIRCX)




```python
import oimodeler as oim

#%% model definition
ud1 = oim.oimUD()
pt= oim.oimPt()
m1 = oim.oimModel(pt,ud1)

pt.params['x'].set(free=True,min=-100,max=100)
pt.params['y'].set(free=True,min=-100,max=100)
ud1.params["f"]=oim.oimParamNorm(pt.params["f"])
ud1.params["d"].max  = 2

#%% fitting the data
files = ["file1.fits","files2.fits"]
fit1 = oim.oimFitterEmcee(files,m1,nwalkers=32,dataTypes=["VIS2DATA","T3PHI"])
fit1.prepare()
fit1.run(nsteps=20000,progress=True)

#%% plotting results
figWalkers1, axeWalkers1 = fit1.walkersPlot(chi2limfact=5)
figCorner1, axeCorner1 = fit1.cornerPlot(discard=15000,chi2limfact=10)
print(fit1.getResults(mode="best",discard=15000))
print("Chi2r = {}".format(fit1.simulator.chi2r))
figSim1, axSim1 = fit1.simulator.plot(["VIS2DATA","T3PHI"],xunit="cycle/mas")
```

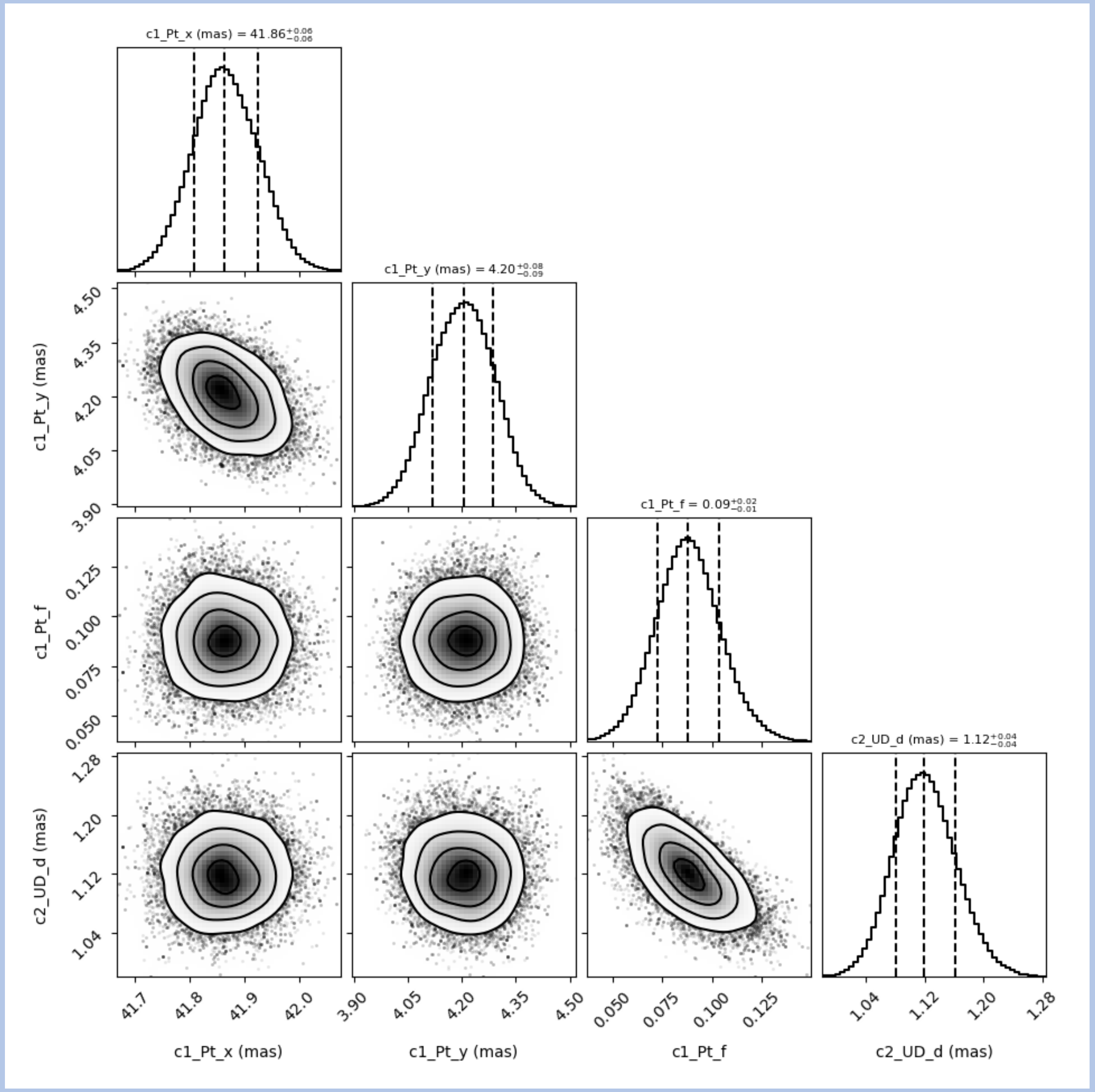
Example fitting real Data (MIRCX)



Walker plot from MCMC run

Showing the convergence of most of the walker to a “global” minimum

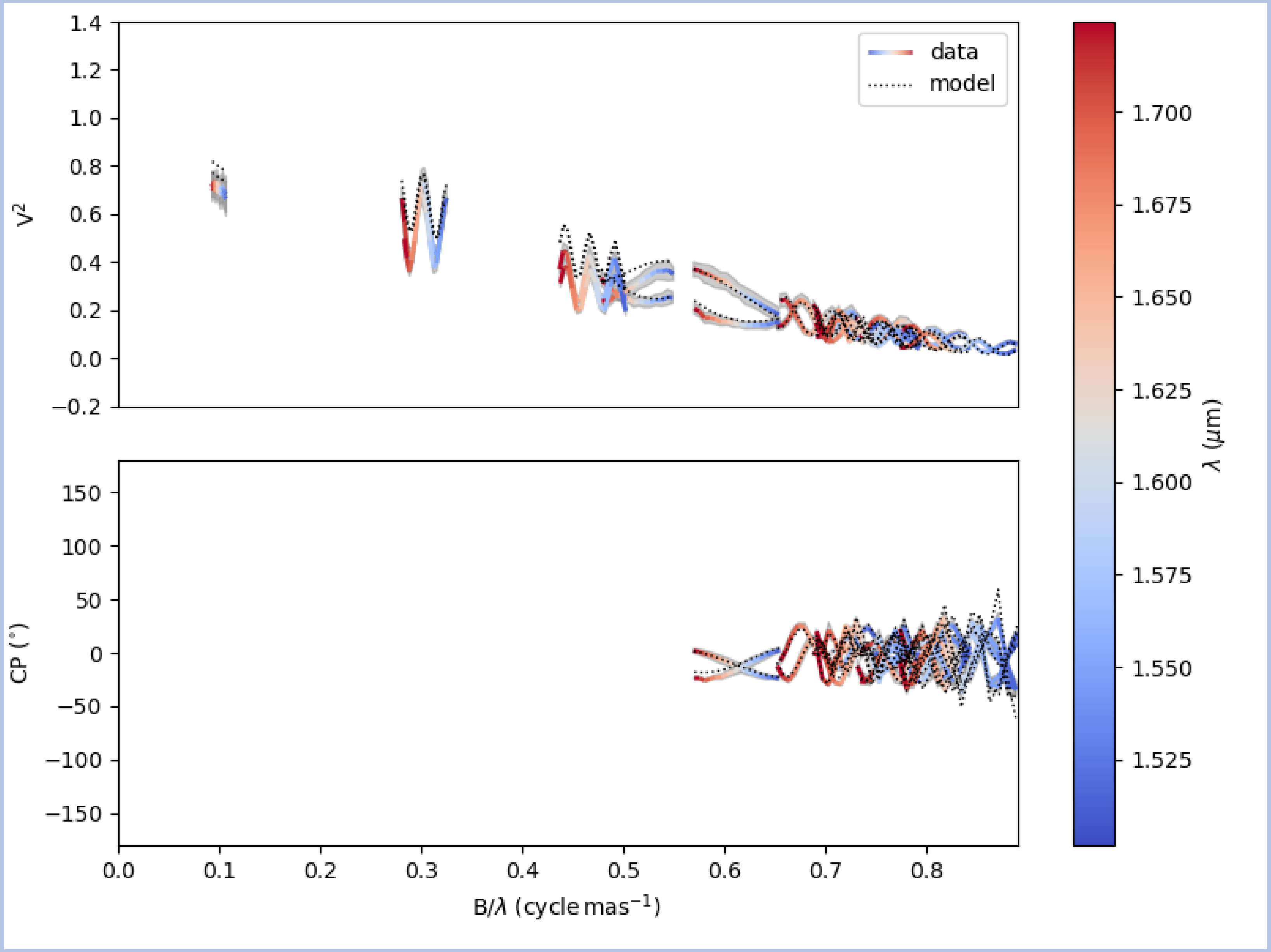
Discussion: Global minimum search



Corner plot from 5000 last step (removing non-converged walkers)

Discussion: uncertainties on parameters

Example fitting real Data (MIRCX)



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…)

Slide from JMMC annual meeting in January 2023

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…)

Conclusion from 2023 development

I mainly focused on:

- Developing new features (it's fun)
- Correcting bugs
- Writing documentation: very time consumming

But a general purposed code needs more than that:

- Extensive tests (unitary + full tests or real data)
- Optimization

Community is building up around oimodeler
(mainly in MATISSE and SPICA groups)
but more users than developpers

- Alexis Matter & Martin Scheuk (MPIA)
  - temperature interpolator
  - temperature gradient disk
- SPICA group : various LDD implementation and test

Conclusions

On oimodeler development

- Code is working although not all advanced features are fully tested
- Development is slower than what I expected
- Need software-and-web-engineering support (test, documentation, deployment, optimization)
- Could/should it become the matrix of a community tool? Support from JMMC?

General Comments on model-fitting (to start our discussion)

- Fitting algorithm
  - Which to use? LM, MCMC, nested-sampling, IA-based …
  - How to optimize global minimum search?
  - What is uncertainties estimations?
  - What about Grid of precomputed models?
- Image-plan models
  - FFT vs DFT (2D)
  - Hankel transform on intensity profiles (1D)
  - Sampling problem ⇔ accuracy vs speed
- Data in OIFITS2 format
  - Uncertainties & Correlation: No instrument uses the OI_CORR table
  - Differential phase: not really well defined in the format?
- Need of common dataset to test modelling software
  - simulated with reference software (ASPRO?)
  - published data from all instruments and various cases