Compact binary coalescences under the microscope: data analysis tools for the present and for the future

Giulia Pagano

Università di Pisa, INFN Pisa

giulia.pagano@pi.infn.it

September 23, 2019 - 2nd year PhD seminar

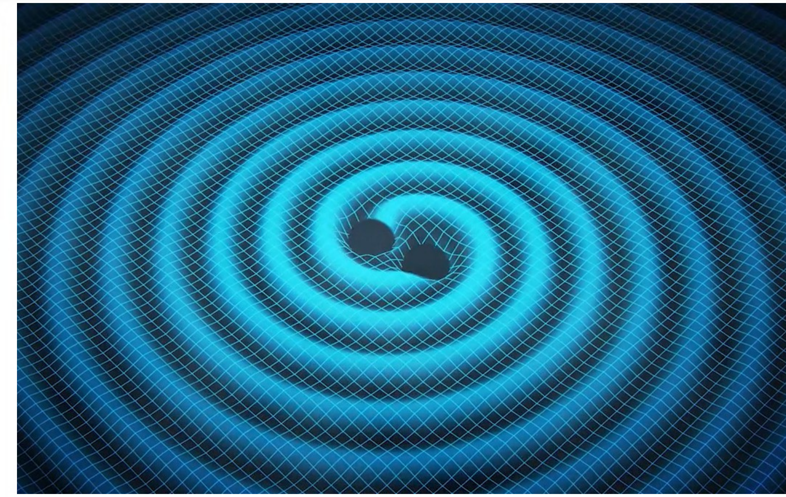
Outlook

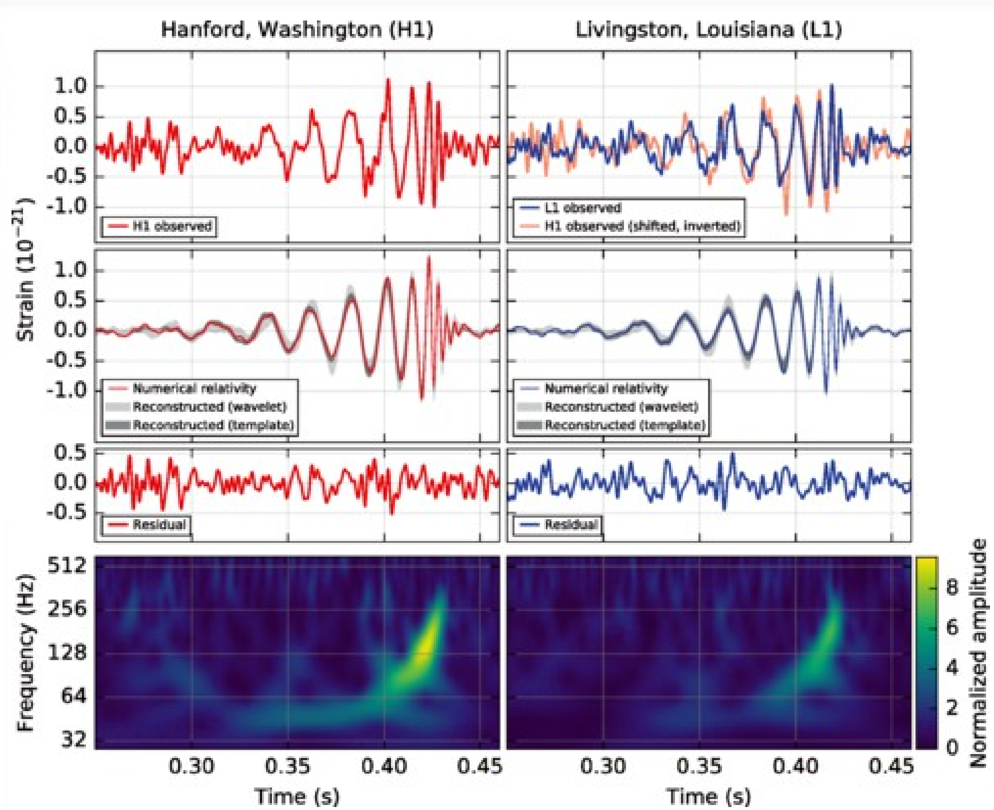
- Introduction
- Data analysis in the era of gravitational wave astronomy
- gwmodel
- microlensingGW
- Conclusions

Introduction

“Ladies and gentlemen, we have detected gravitational waves. We did it!”

David Reitze - Executive Director of the LIGO Laboratory (11/02/2016)





Phys. Rev. Lett., 116, 061102 (2016)

Data analysis in the era of gravitational wave astronomy

What's next for data analysis?

Improvement

- Independent inference
- Usability
- Flexibility

New science

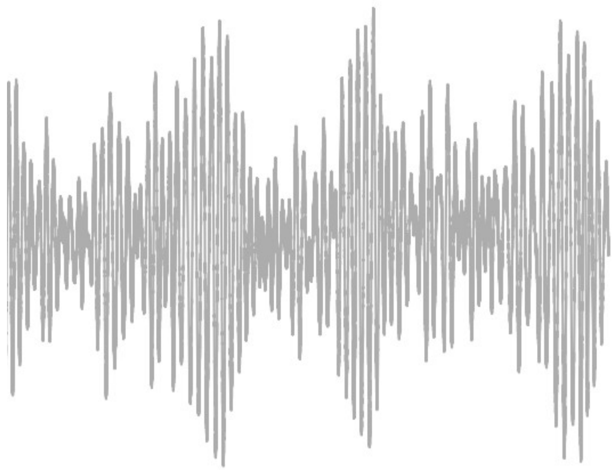
- Dedicated tools for cutting-edge research
- Address unobserved phenomena

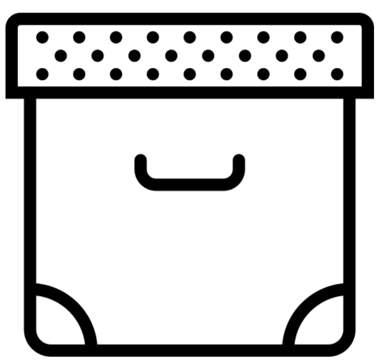
gwmodel

microlensingGW

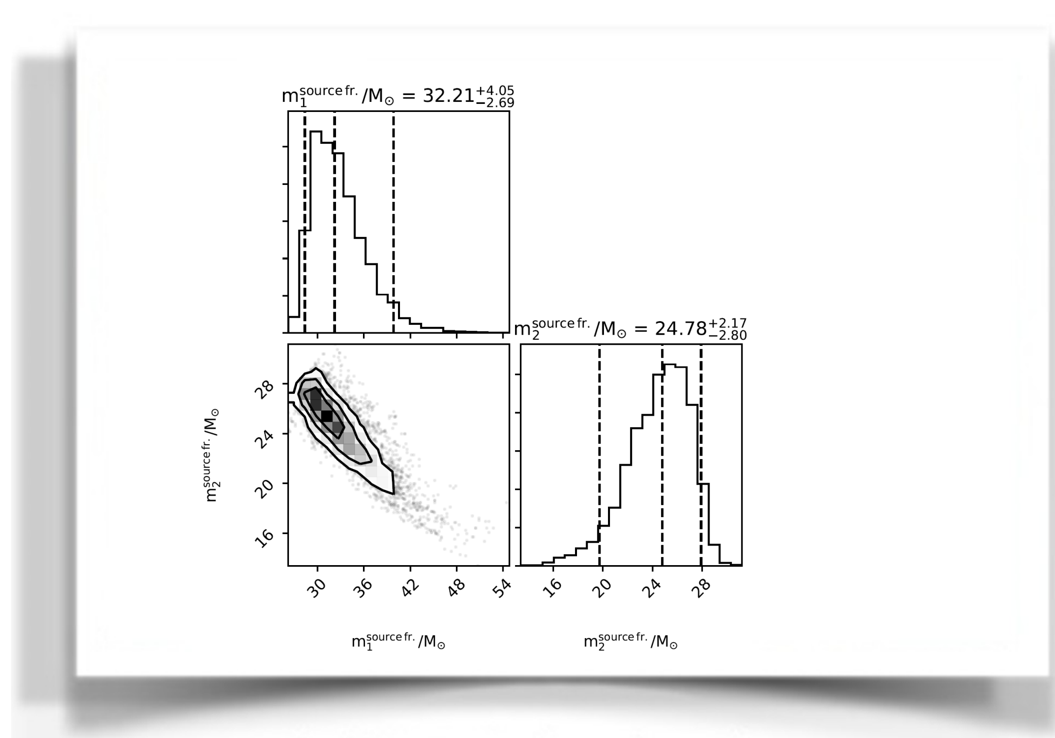
gwmodel

CBC inference pipeline

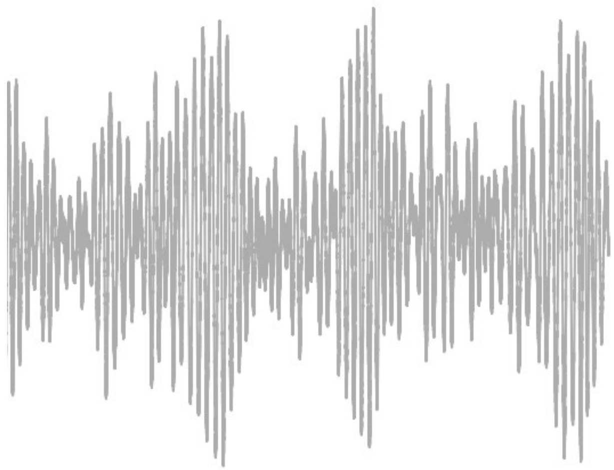


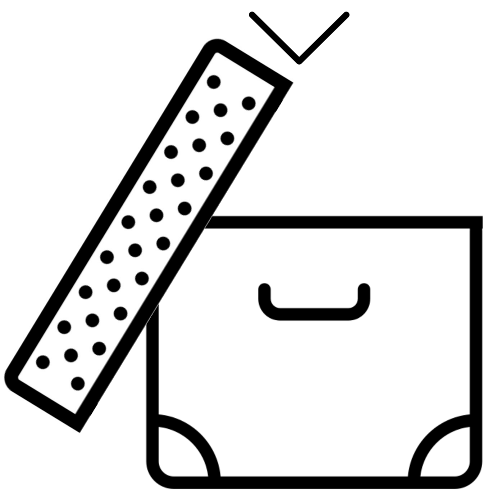


pipeline

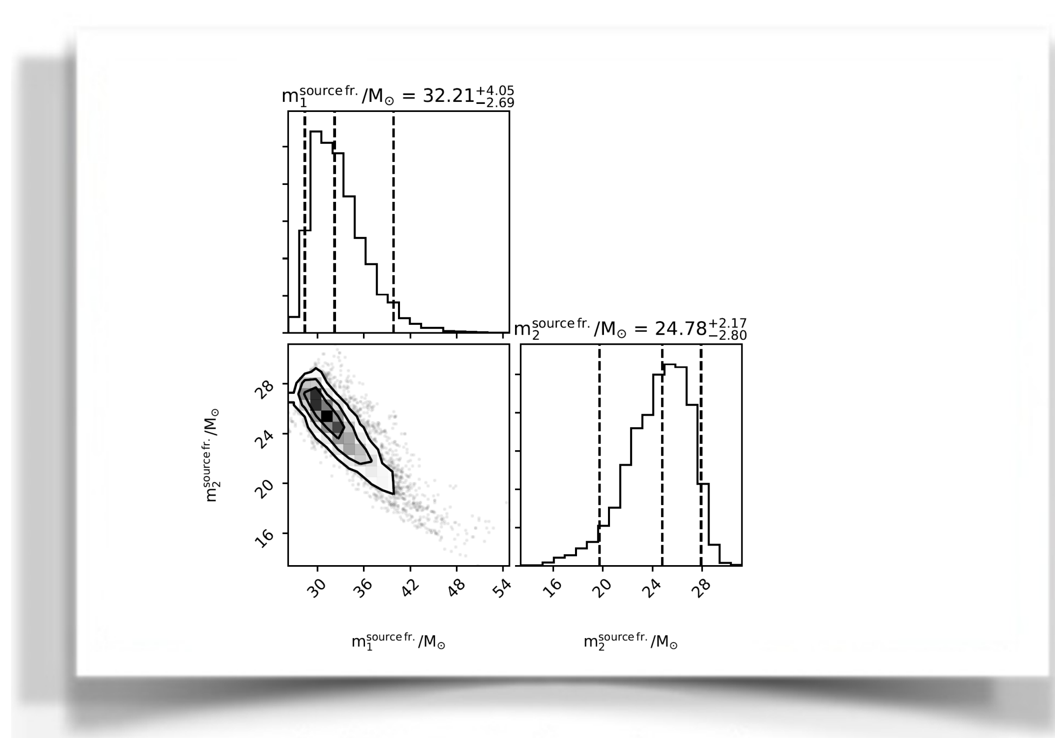


CBC inference pipeline

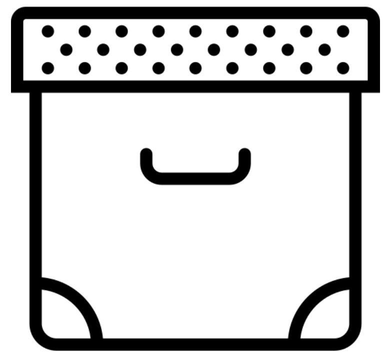




pipeline



gwmodel



pipeline

cbcmodel

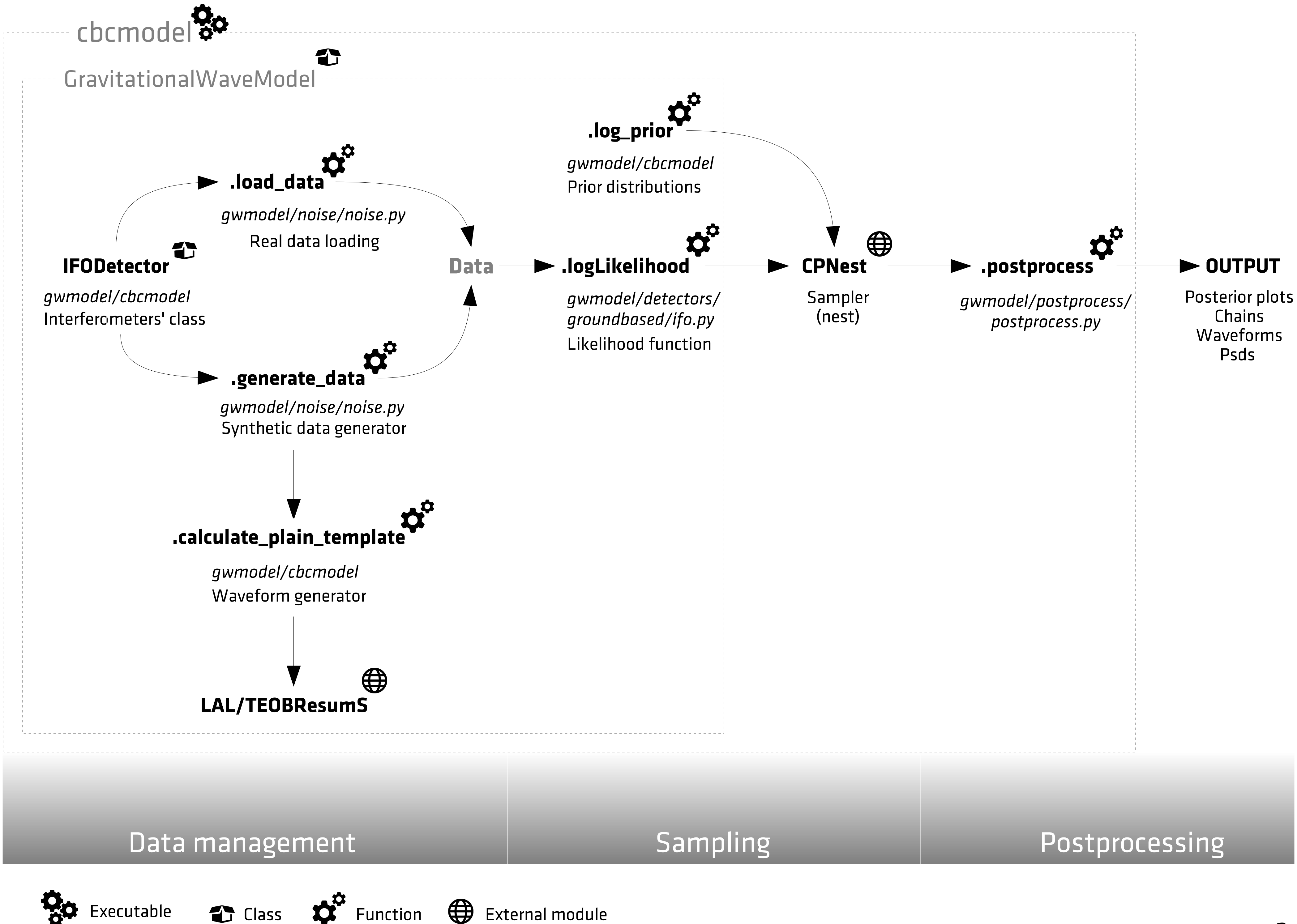
Inference

- Time/frequency domain
- Real/simulated signals
- Arbitrary number of ground-based detectors
- Supports any LAL waveform

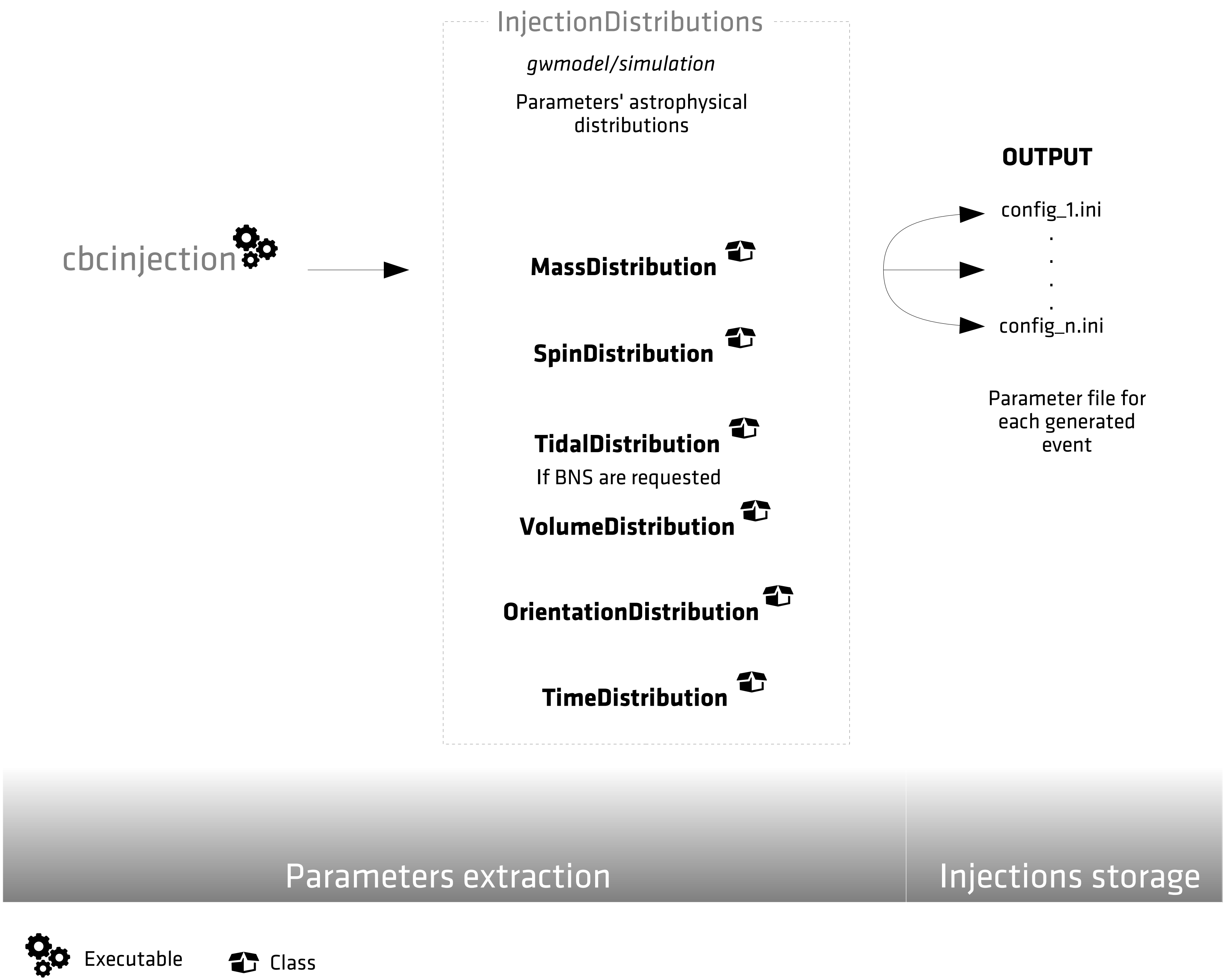
cbcinjection

Mock signals

- Arbitrary astrophysical distributions


cbcmodel
GravitationalWaveModel
.log_prior
gwmodel/cbcmodel
Prior distributions
.load_data
gwmodel/noise/noise.py
Real data loading
IFODetector
gwmodel/cbcmodel
Interferometers' class
Data
.logLikelihood
gwmodel/detectors/
groundbased/ifo.py
Likelihood function
CPNest
Sampler
(nest)
.postprocess
gwmodel/postprocess/
postprocess.py
OUTPUT
Posterior plots
Chains
Waveforms
Psds
.generate_data
gwmodel/noise/noise.py
Synthetic data generator
.calculate_plain_template
gwmodel/cbcmodel
Waveform generator
LAL/TEOBResumS
Data management
Sampling
Postprocessing
Executable
Class
Function
External module

cbcinjection

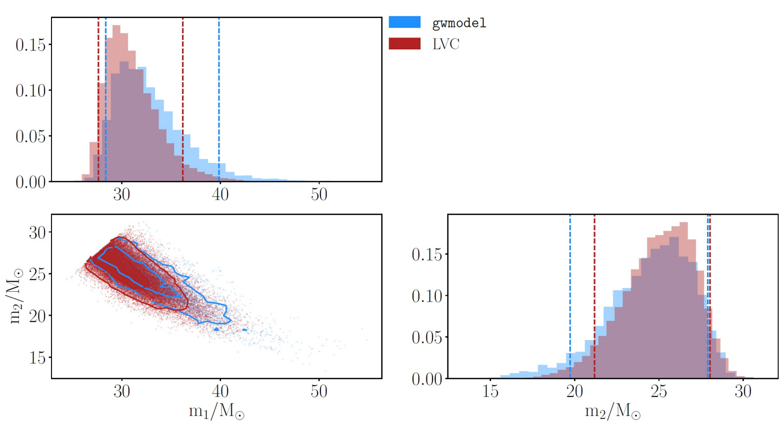


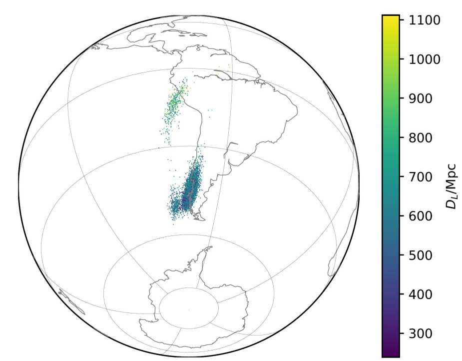
Results

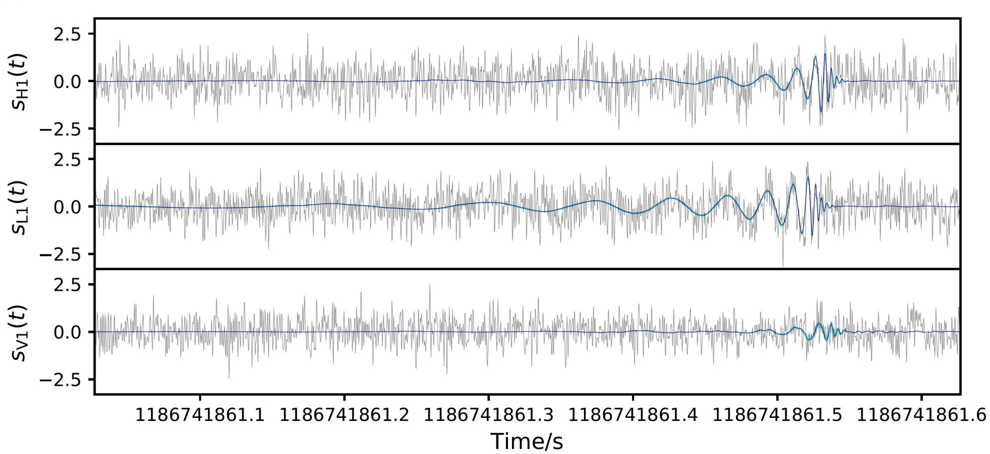
- Tested on **O1** and **O2** events, results compatible with LVC

GW170814

- IMRPhenomPv2, estimated PSD, bounds informed by LVC results







Parameter	LVC	gwmodel
Source frame primary mass $m_1/M_\odot$	$30.7^{+5.7}_{-3.0}$	$32.2^{+7.6}_{-3.8}$
Source frame secondary mass $m_2/M_\odot$	$25.3^{+2.9}_{-4.1}$	$24.8^{+3.1}_{-5.1}$
Source frame chirp mass $\mathcal{M}_c/M_\odot$	$24.2^{+1.4}_{-1.1}$	$24.5^{+1.2}_{-1.1}$
Source frame final mass $M_f/M_\odot$	$53.4^{+3.2}_{-2.4}$	$54.1^{+3.7}_{-2.4}$
Effective inspiral spin χ_{eff}	$0.07^{+0.12}_{-0.11}$	$0.00^{+0.18}_{-0.24}$
Final spin a_f	$0.72^{+0.07}_{-0.05}$	$0.67^{+0.06}_{-0.09}$
Luminosity distance D_L/Mpc	580^{+160}_{-210}	627^{+163}_{-213}

Future developments

p-p plot as final validation

- 100 simulated events study
- Eventual debugging

Speed up performances

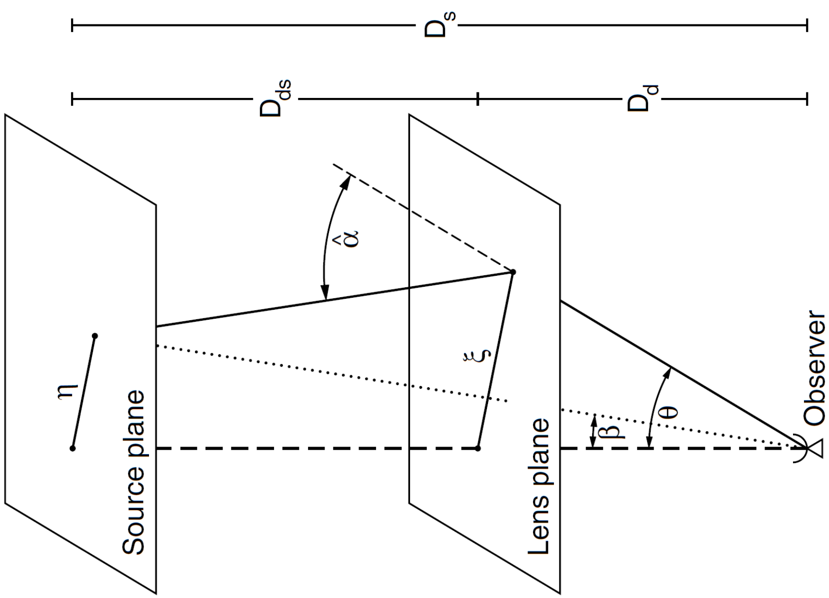
- Investigation of the critical passages

Public release

- Organisation, documentation, license

microlensingGW

Lensing



$$\boldsymbol{\beta} = \boldsymbol{\theta} - \boldsymbol{\alpha}(\boldsymbol{\theta}), \; \boldsymbol{\alpha} = \boldsymbol{\nabla}\psi$$

Schneider, Kochanek and Wambsganss
Gravitational Lensing: Strong, Weak and Micro (2006)

Strong lensing

- Multiple images
- Magnification/demagnification
- Time delay

Microlensing

- Stellar mass lenses
- Too small separations for optical resolution (μarcsec)
- Mesurable in time resolution (ms) by LIGO/Virgo

GWs as a unique probe

Infer lens properties unobservable in optical

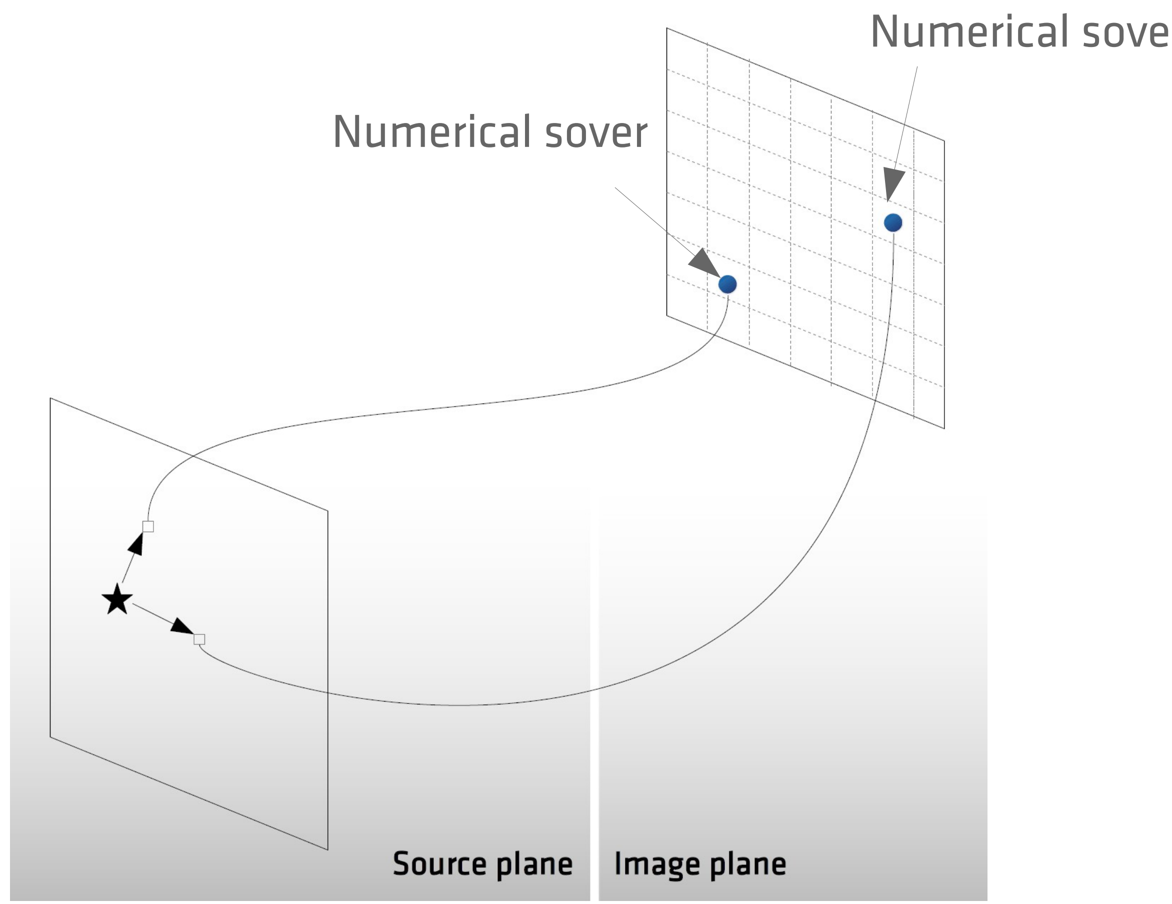
Criticalities

- Need to solve two non-linear, algebraic, coupled equations in two dimensions
- No procedure is guaranteed to find a complete set of solutions in 2D (Press et al., 2007)

Standard approach

$$\beta = \theta - \alpha(\theta)$$

- Ray-shooting + numerical solver

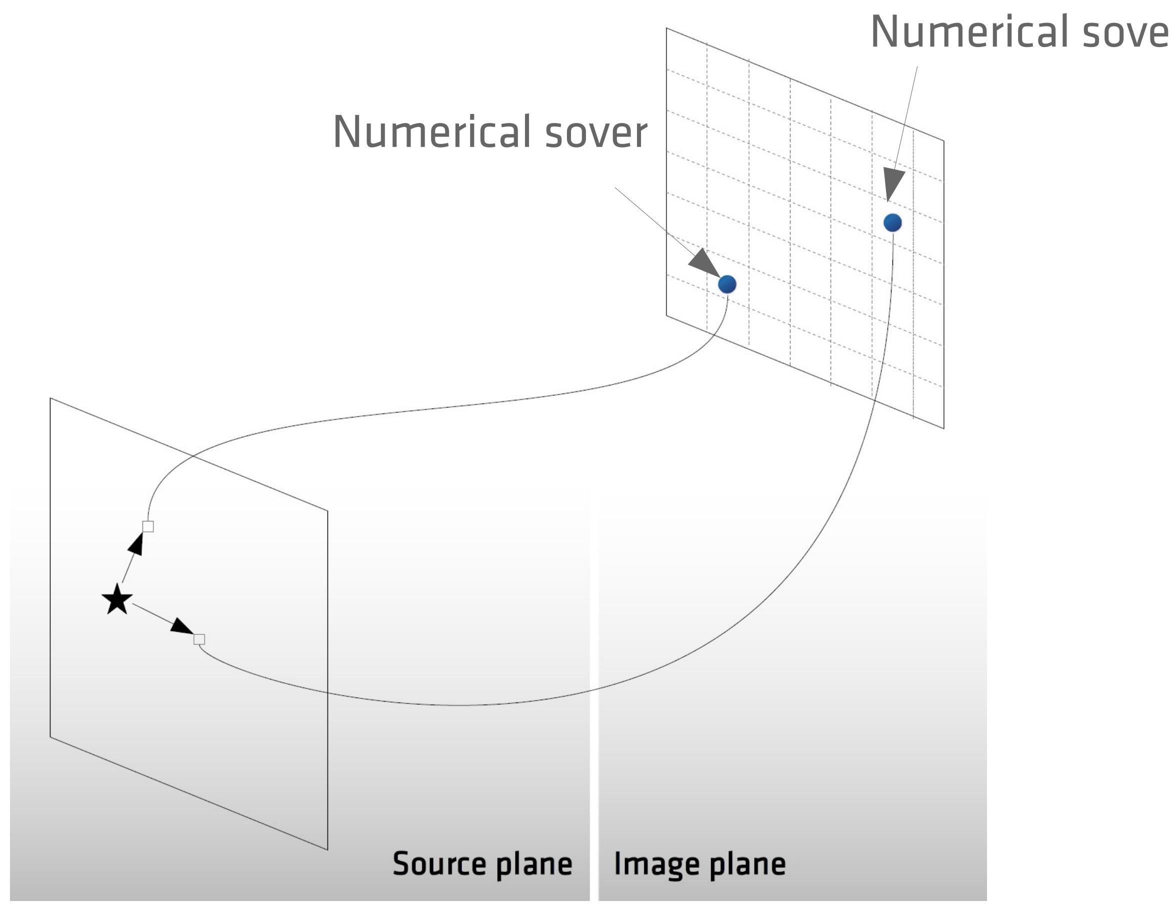


Criticalities

- Need to solve two non-linear, algebraic, coupled equations in two dimensions
- No procedure is guaranteed to find a complete set of solutions in 2D (Press et al., 2007)

Standard approach

- **Not suitable to microimages** separation
- **Not guarantee to handle** potentials with very **different scales** (galaxy + the halo of stellar mass lenses)

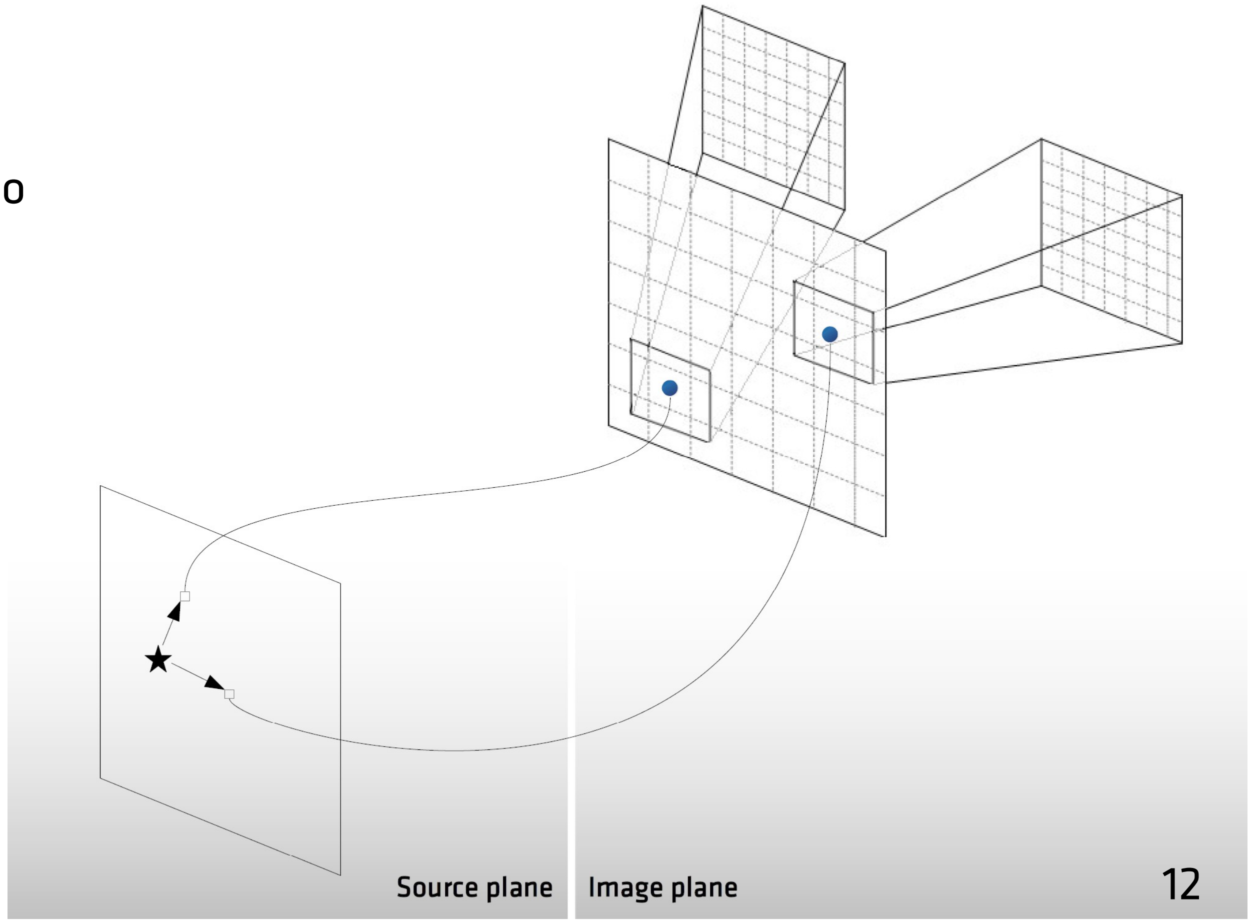


microlensingGW

- **New algorithm** specifically tailored for microlensing
- Direct access to **waveforms** and **strains** thanks to its integration with **gwmodel**
- Necessary for systematic parameter space investigations, model selection, detectability studies, ...

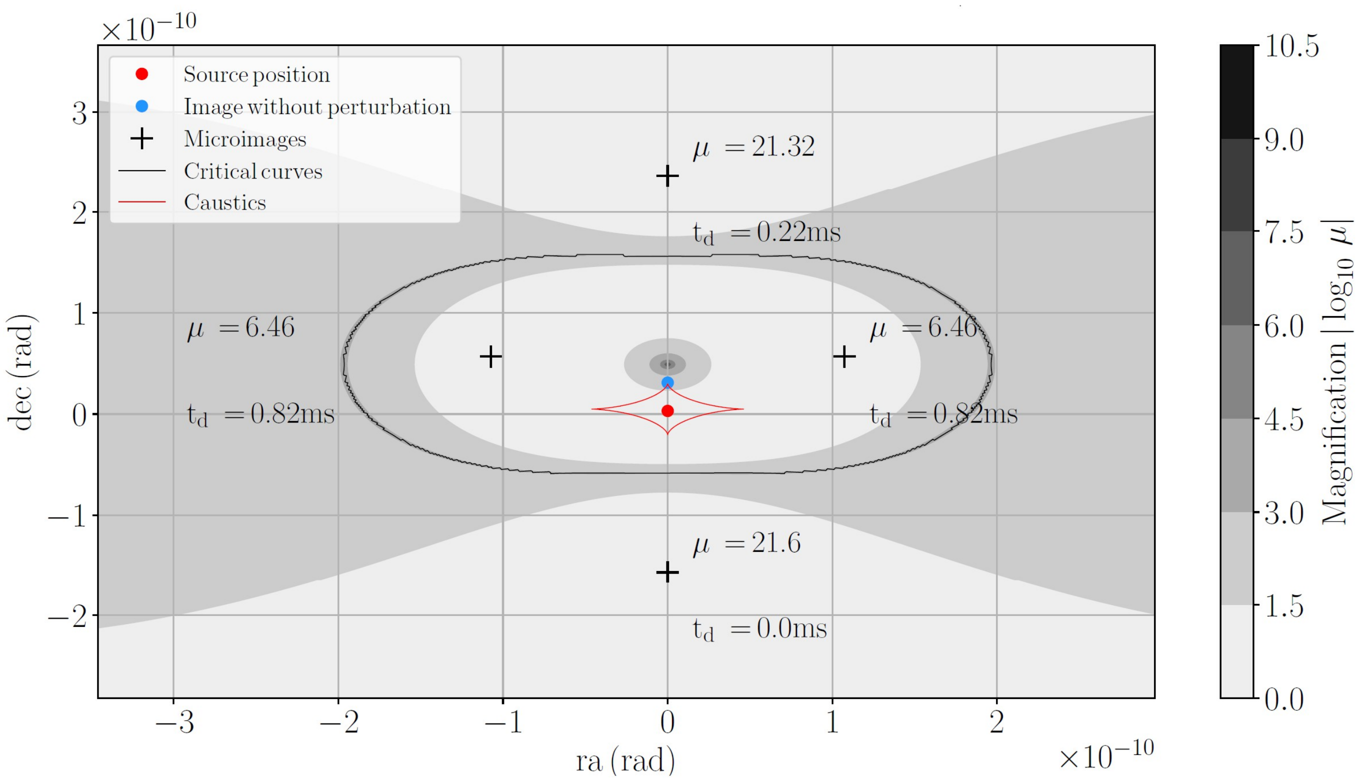
A new strategy

- **Split the solution** of the system into strong lenses and strong lenses + microlenses
- **Dynamical iteration** on **adaptive grids**
- The potential determines the **stopping condition** of the integration



Results

Validation on the literature



Diego et al. (2019)

- Galaxy +100 Msun point mass
- Reproduces images' geometry, magnifications and time delays

$$\psi_G = \frac{\kappa}{2}(\theta_x^2 + \theta_y^2) + \frac{\gamma}{2}(\theta_x^2 - \theta_y^2)$$ > Total magnification = 30

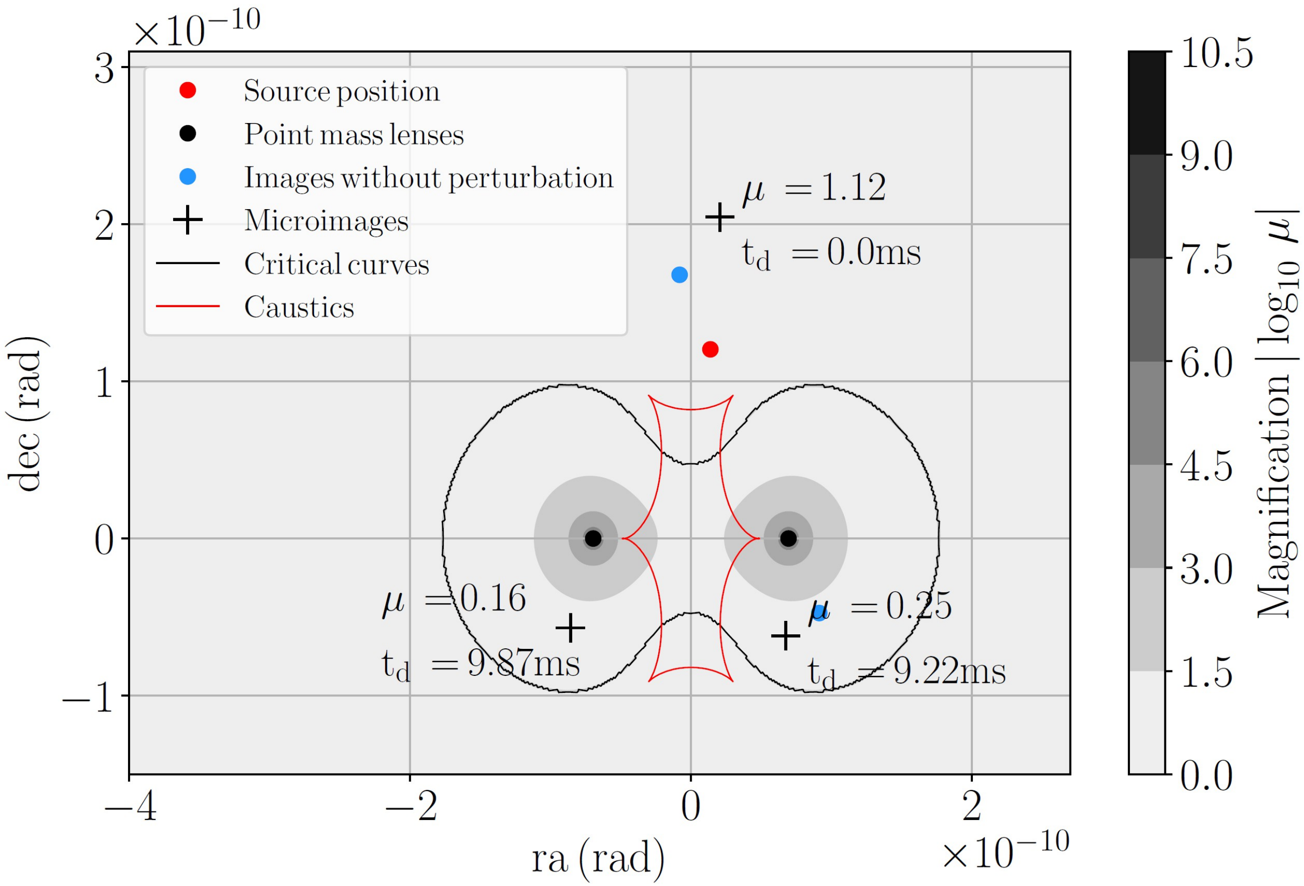
κ → 0.39

γ → -0.06

$$\Psi_{PS} = \frac{4GMD_{ds}}{c^2 D_d D_s} \ln(\boldsymbol{\theta} - \boldsymbol{\theta_0})$$

Results

Validation on the literature

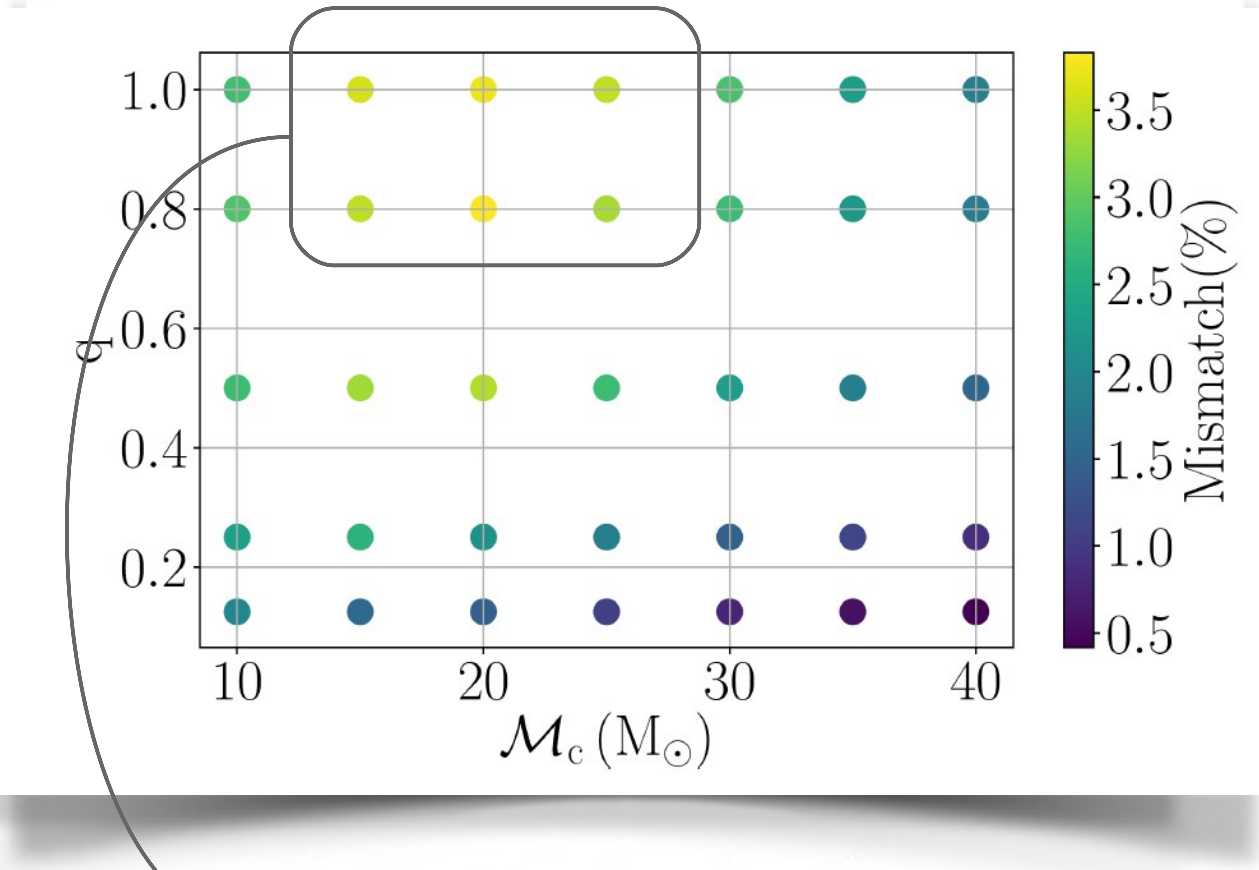


Schneider, Weiss (1986)

- Binary point mass lens
- Reproduces images' geometry and positions

Results

New findings



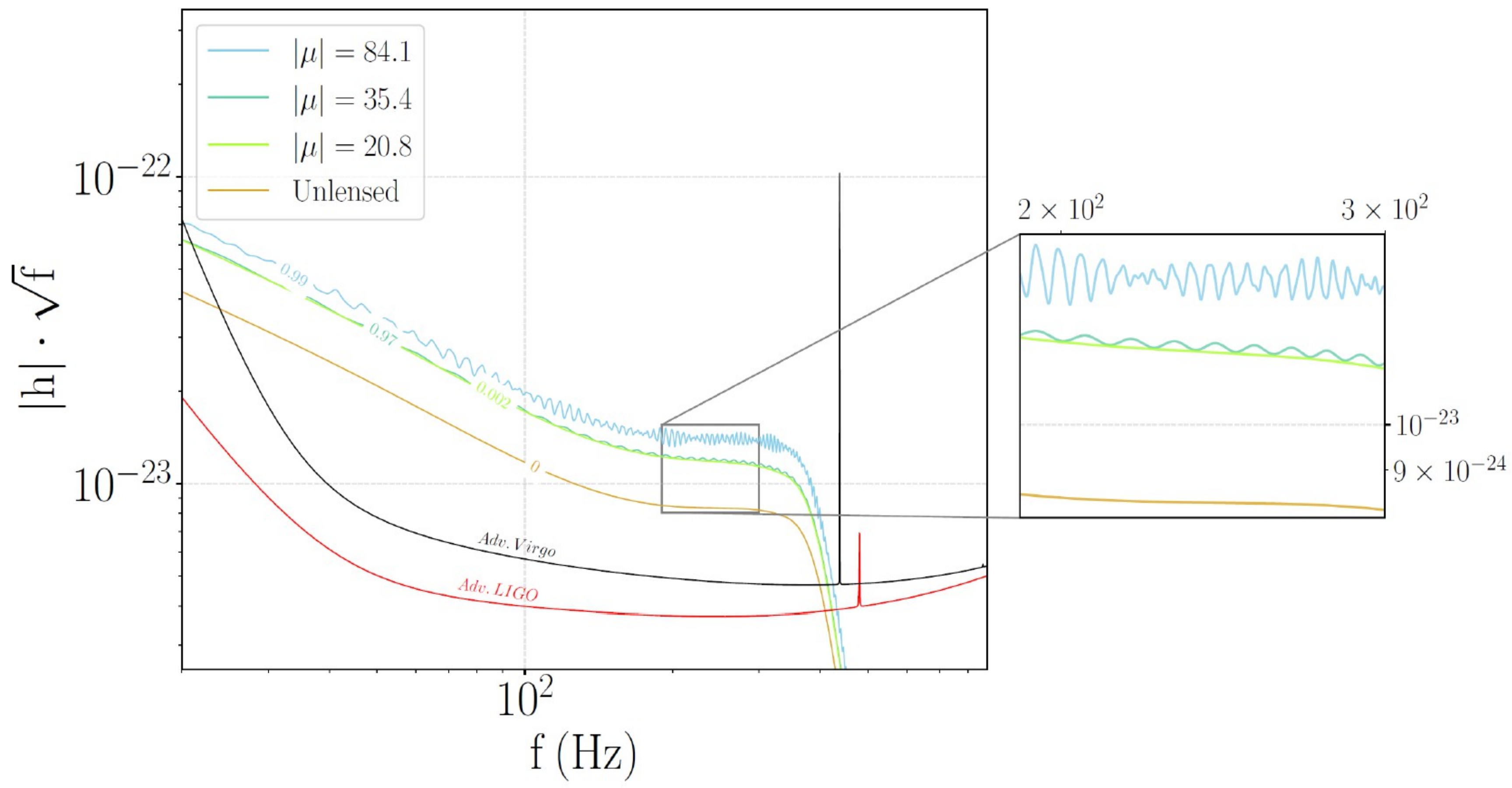
Distinguishable as lensed events

Diego et al. (2019)

- GW150914-like source
- H1-L1 network at O1 sensitivity

Results

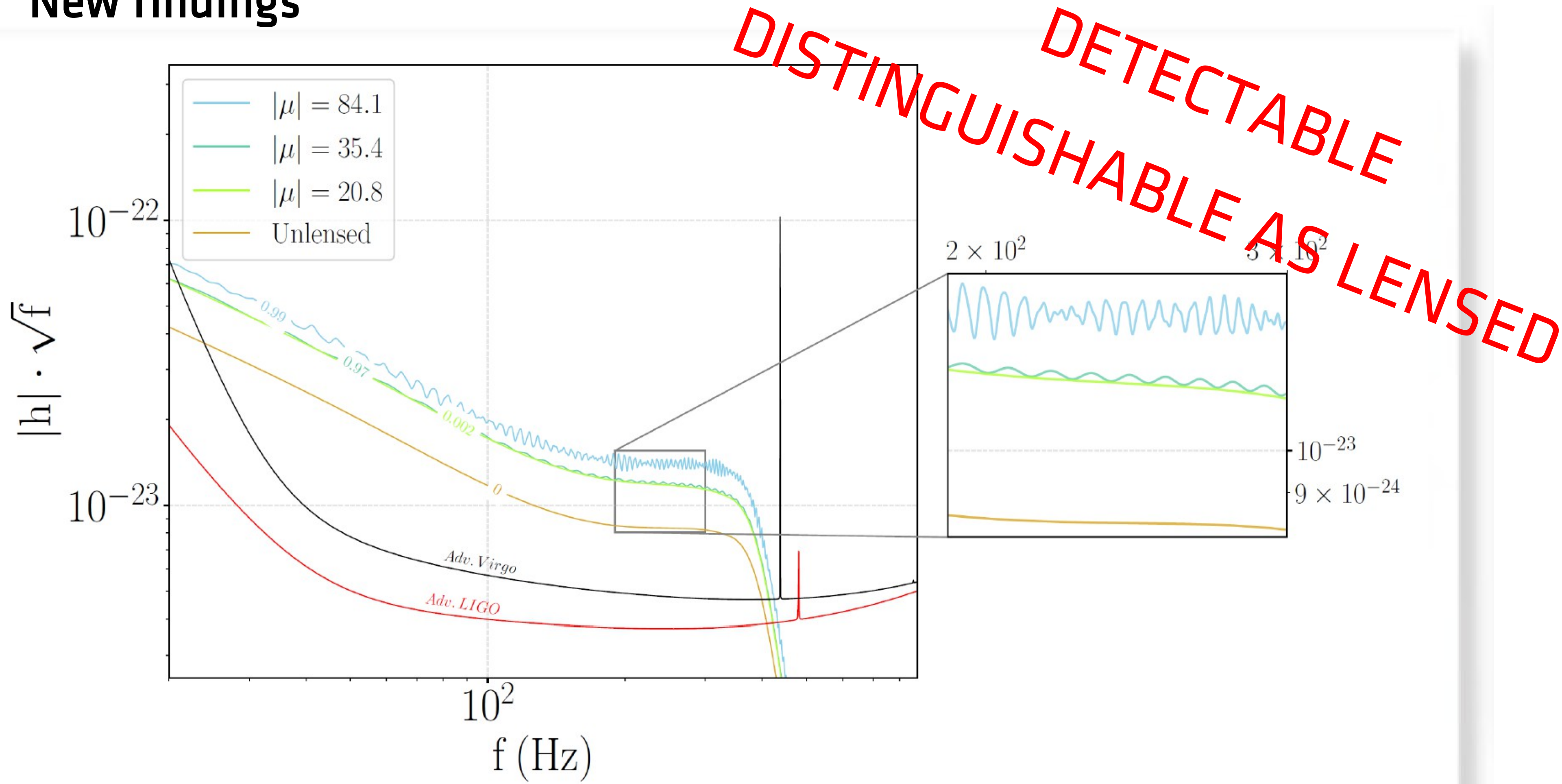
New findings



- Elliptical galaxy + ~900 microlenses
- Core radius 500pc, $\epsilon = 0.1$
- Non spinning binary, (Mc=20, q=0.8)
- H1-L1-V1 network at design sensitivity

Results

New findings



- Elliptical galaxy + ~900 microlenses
- Core radius 500pc, $\epsilon = 0.1$
- Non spinning binary, (Mc=20, q=0.8)
- H1-L1-V1 network at design sensitivity

Future developments

Improvement of the magnification routine

- More refined approximation to the magnification factor

Public realease

- Organisation, documentation, license

Model investigation

- Background distribution, macromodel

Conclusions

gwmodel

- Functional Python pipeline for CBC inference
- Flexible, extendible, accessible to both experts and beginners
- Reproduces LIGO/Virgo results

microlensingGW

- First Python software for microlensing (of GWs)
- New solving algorithm
- Waveforms and strains as byproducts
- Validated on the literature
- First systematic assessment of detectability and lensed distinguishability on realistic models

THANK YOU!

BACK UP SLIDES

Bayesian inference

- **Probability distributions** are recovered **for each parameter**
- Uses **Bayes' theorem**

Bayes' theorem

$$p(\vec{\theta}|dHI) = p(\vec{\theta}|HI) \cdot \frac{p(d|\vec{\theta}HI)}{p(d|HI)}$$

Bayesian inference

- **Probability distributions** are recovered **for each parameter**
- Uses **Bayes' theorem**

Bayes' theorem

$$p(\vec{\theta}|d\,H\,I) = p(\vec{\theta}|H\,I) \cdot \frac{p(d|\vec{\theta}\,H\,I)}{p(d|H\,I)}$$

The data

$d_H(t)$, $d_L(t)$, $d_V(t)$

The model

GR signal

Noise

Prior information

Any **information** available **before** analysing the **data**

Bayesian inference

- **Probability distributions** are recovered **for each parameter**
- Uses **Bayes' theorem**

Bayes' theorem

$$p(\vec{\theta}\,|\,d\,H\,I) = p(\vec{\theta}\,|\,H\,I) \cdot \frac{p(d|\vec{\theta}\,H\,I)}{p(d|H\,I)}$$

The data

$d_H(t)$, $d_L(t)$, $d_V(t)$

The model

GR signal

Noise

Prior information

Any **information** available **before** analysing the **data**

Bayesian inference

- **Probability distributions** are recovered **for each parameter**
- Uses **Bayes' theorem**

Bayes' theorem

$$p(\vec{\theta}\,|\,d H I) = p(\vec{\theta}\,|\,H I) \cdot \frac{p(d|\vec{\theta}\,H I)}{p(d|H I)}$$

The data

$d_H(t)$, $d_L(t)$, $d_V(t)$

The model

GR signal

Noise

Prior information

Any **information** available **before** analysing the **data**

Bayesian inference

- **Probability distributions** are recovered **for each parameter**
- Uses **Bayes' theorem**

Bayes' theorem

$$p(\vec{\theta}|dHI) = p(\vec{\theta}|HI) \cdot \frac{p(d|\vec{\theta}\,HI)}{p(d|HI)}$$

Posterior

Prior

Likelihood

Model selection

Bayes' theorem

$$p(H_1|DI) = p(H_1|I)\frac{p(D|H_1I)}{p(D|I)}$$

$$p(H_2|DI) = p(H_2|I)\frac{p(D|H_2I)}{p(D|I)}$$

Odds ratio and Bayes' factor

$$O_{1,2} \equiv \frac{\cancel{p(H_1|I)}}{\cancel{p(H_2|I)}}\frac{p(D|H_1I)}{p(D|H_2I)}$$

gwmodel: inference model

Priors

Parameter	Distribution	Reparametrisation	Prior bounds
m_i	uniform	$\mathcal{M}_c, q$	$\mathcal{M}_c \in [8.5, 70]\mathrm{M}_\odot, q \in [0.125, 1]$
D_L	D_L^2	$\ln D_L$	$[\ln(1/\mathrm{Mpc}), \ln(3000/\mathrm{Mpc})]$
right ascension (ϕ)	uniform		$[0, \pi]$
declination (θ)	$\cos\theta$		$[-\pi/2, \pi/2]$
a_i	uniform		$[0, 0.8], [0, 1]^*$
θ_{li}	$\sin\theta_l$		$[0, \pi], \{0, \pi\}^*$
ϕ_{li}	uniform		$[0, 2\pi]$, UNDEF.*
inclination (ι)	"		$[0, \pi]$
ψ	"		$[0, \pi]$
ϕ_0	"		$[0, 2\pi]$
t_c	"		$[t_c^{\mathrm{guess}} - 0.05, t_c^{\mathrm{guess}} + 0.05]\mathrm{s}$
Λ_i	"		$[1, 5000]$

Likelihood

$$\ln p(d|\vec{\theta}, H, I) = -\frac{2}{T} \sum_{IFO} \int df \frac{|\tilde{d}(f) - \tilde{h}(f)|^2}{S_n(f)}$$

Simulation distributions

Parameter	Distributions	Output	Frame
m_i	uniform, power law, Gaussian (source frame)	$\mathcal{M}_c, q$	source frame, detector frame
D_L	D_L^2 or redshift extracted according to uniform comoving volume density		
right ascension (ϕ)	uniform		
declination (θ)	$\cos\theta$		
a_i	uniform		
θ_{li}	$\sin\theta_l$		
ϕ_{li}	uniform		
inclination (ι)	$\sin\iota$		
ψ	uniform		
ϕ_0	"		
t_c	Poisson, gridded		
Λ_i	EOS or uniform		

cbcmodel: usage

Real data

$ cbcmodel - - config-file pipeconfig.ini

```
[input]

full-run=True
inject=
zero-noise=false
starttime=1186741854.5268
trigtime=1186741861.5268
seglen=8
sampling-rate=2048
dt=0.1
flow=20
fhigh=1024
H1-psd=
L1-psd=
V1-psd=
H1-data=/data/gwosc_catalog/H-H1_GWOSC_4KHZ_R1-1186739814-4096.txt
L1-data=/data/gwosc_catalog/L-L1_GWOSC_4KHZ_R1-1186739814-4096.txt
V1-data=/data/gwosc_catalog/V-V1_GWOSC_4KHZ_R1-1186739814-4096.txt
E1-psd=
E2-psd=
E3-psd=
E1-data=
E2-data=
E3-data=
output=/gw170814/
detectors=H1,L1,V1

[Sampler settings]

nlive=1024
poolsize=512
maxmcmc=8192
nthreads=2
verbose=3
resume=0
nhamiltonian=0

[Template settings]

approximant = IMRPhenomPv2
amp_order = 0
phase_order = -1
fref = 20.0
tidal = False

[Calibration settings]
enable-calibration-model=False
H1-bands = 5
H1-amplitudes = [0.0482,0.0482,0.0482,0.0482,0.0482]
H1-phases = [3.158,3.158,3.158,3.158,3.158]
L1-bands = 5
L1-amplitudes = [0.0823,0.0823,0.0823,0.0823,0.0823]
L1-phases = [4.196,4.196,4.196,4.196,4.196]

[Priors]

spin1-min=0.0
spin1-max=0.8
spin2-min=0.0
spin2-max=0.8
phi0-min=
phi0-max=
ra-min=
ra-max=
dec-min=
dec-max=
tc-min=
tc-max=
q-min=0.125
q-max=1.0
mc-min = 15
mc-max = 40
inclination-min=
inclination-max=
psi-min=
psi-max=
logdistance-min=0.0
logdistance-max=7.6
lambda-min=
lambda-max=
```

cbcmodel: usage

Real data

$ cbcmodel - - config-file pipeconfig.ini

Simulated signals

$ cbcmodel - - injection-file event_0.ini - - config-file pipeconfig.ini

```
[mass parameters]
m1 = 19.302946708834618
m2 = 8.996693480778454
mc = 11.310133698048183
q = 0.4660787607448984
eta = 0.21684265110788636

[spin parameters]
spin1 = 0.44694931320679365
spin2 = 0.09026274524901048
theta_l1 = 1.015889256075944
theta_l2 = 1.103349347601922
phi_l1 = 2.2678390242422837
phi_l2 = 2.204428071142985
spin_support = volumetric

[tidal parameters]
lambda1 = 0.0
lambda2 = 0.0
eos = None

[extrinsic parameters]
logdistance = 7.541588024175696
redshift = 0.3544909435087297
ra = 5.489818751365282
dec = -0.004189383708262077
tc = 3.0092469946733753

[cosmological parameters]
h = 0.7
omega_m = 0.3
omega_lambda = 0.7
w0 = -1.0
w1 = 0.0
w2 = 0.0

[orientation parameters]
psi = 3.0729561106171928
inclination = 1.999742986590998
phi0 = 0.6505154083075078

[waveform parameters]
approximant = IMRPhenomPv2
amp_order = 0
phase_order = -1
fref = 100.0
```

cbcinjection: usage

$ cbcinjection - - config-file simulation.ini - - output-folder ./Test

```
[waveform settings]

approximant=TaylorF2
amp_order=0
phase_order=-1
fref = 100.0

[mass distribution]



q_min=0.8
q_max=1.0
m1_min = 10.0
m1_max = 20.0
m2_min = 10.0
m2_max = 20.0
mc_min = 1.0
mc_max = 50.0
distribution = uniform
frame = detector_frame
variable = component_masses

[spin distribution]

spin1_min=0.0
spin1_max=0.99
spin2_min=0.0
spin2_max=0.99
theta_1l_min = 0.0
phi_1l_min = 0.0
theta_1l_max = np.pi
phi_1l_max = 2.0*np.pi
theta_2l_min = 0.0
phi_2l_min = 0.0
theta_2l_max = np.pi
phi_2l_max = 2.0*np.pi
distribution = aligned

[orientation distribution]

inclination_min=0.0
inclination_max=np.pi
psi_min=0.0
psi_max=np.pi
phi0_min=0.0
phi0_max=2.0*np.pi

[volume distribution]

; Set cosmology True to sample the redshift and convert to the distance.
cosmology=True
distance_min=1.0
distance_max=1000.0
redshift_min=0.0
redshift_max=0.5
ra_min=0.0
ra_max=2.0*np.pi
dec_min=-np.pi/2.0
dec_max=np.pi/2.0

[time distribution]

tc_min=0.0
tc_max=100.0
distribution = gridded
n = 100

[tidal distribution]

eos=
lambda1_min=
lambda1_max=
lambda2_min=
lambda2_max=
```

Lensing theory

Time delay

$$t_d(\boldsymbol{\theta}, \boldsymbol{\beta}) = \frac{(1+z_L)D_d D_s}{cD_{ds}}\left[\frac{1}{2}|\boldsymbol{\theta} - \boldsymbol{\beta}|^2 - \psi(\boldsymbol{\theta}) + \Phi_m(\boldsymbol{\beta})\right]$$

Amplification factor

$$F(\omega_{GW}, \boldsymbol{\beta}) = \frac{D_s D_d}{D_{ds}} \frac{\omega_{GW}}{2\pi i} \frac{(1+z_L)}{c} \int d^2\theta \exp\left[i\omega_{GW} t_d(\boldsymbol{\theta}, \boldsymbol{\beta})\right]$$

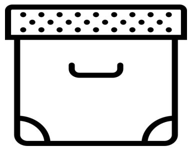
$$h_{\text{LENSED}} = F \cdot h_{\text{UNLENSED}}$$

Geometrical optics

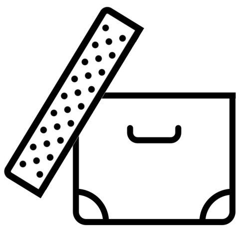
$$F_{geo} = \sum_j |\mu_j|^{1/2} exp[i\omega_{GW} t_d(\boldsymbol{\theta}_j, \boldsymbol{\beta}) - i\pi n_j]$$

$$\mu_j = \det \frac{\partial \boldsymbol{\beta}}{\partial \boldsymbol{\theta}}\Big|_{\boldsymbol{\theta}_j}$$

microlensingGW: usage



Default settings



Additional funtionalities to tune
the iteration parameters
(problematic potentials)

```
$ python MyLensingScript.py  - - injection-file event_0.ini  - - config-file  pipeconfig.ini
   - - output-folder ./Test
```

- Must define the lensing potentials, e.g. through LENSTRONOMY

  (Birrer, Amara arXiv:1804.012)

```python
# microlensingGW imports
from microlensinggw.waveform.waveform import gwmodel_strain, gwmodel_lensed # gwmodel integrated lensing routines
from microlensinggw.solver.microlensing import microimages                 # microlensing solver
from microlensinggw.parser.parser import parse                             # parser script to parse from command line


                            ##########################################################################################
                            #                   compute the unlensed waveform with gwmodel                           #
                            ##########################################################################################


# parse the ini files from command line
injection_file, config_file, output_folder = parse()
# compute the projected strain in time domain and frequency domain with gwmodel
times, frequencies, strain_td, strain_fd, hptilde, hctilde, psd, seg_len, sampling_rate, snr_quantities = gwmodel_strain(injection_file,config_file)
```

```python
                            ##########################################################################################
                            #      solve for the images, time delays, magnification etc. of the given model       #
                            ##########################################################################################



# solver settings
kwargs = {'NearSource': False,
          'scaled': False,
          'img_idx': 0}

# Solve for the images and related quantities
microImgRA, microImgDEC, macroImgRA, macroImgDEC, F, Mu, td, diff = microimages(f                    = frequencies,
                                                                                 y0                   = SourceRA,
                                                                                 y1                   = SourceDEC,
                                                                                 zS                   = z_source,
                                                                                 lens_model_list      = LensModelList,
                                                                                 redshift_list        = RedshiftList,
                                                                                 funct_list           = LensClassList,
                                                                                 macro_index          = [0],
                                                                                 search_window_macro  = 3,
                                                                                 search_window        = 0.0000001,
                                                                                 precision_limit      = 10**(-20),
                                                                                 verbose              = True,
                                                                                 signal_length        = 1.44,
                                                                                 cosmo                = cosmology,
                                                                                 optimization         = False,
                                                                                 **kwargs)
```

```python
                            ##########################################################################################
                            #                 compute the lensed strain and waveform with gwmodel                    #
                            ##########################################################################################

lensed_strain_fd, lensed_strain_td, hptilde_lensed, hctilde_lensed = gwmodel_lensed(hptilde,
                                                                                     hctilde,
                                                                                     strain_fd,
                                                                                     Fmag,
                                                                                     seg_len,
                                                                                     snr_quantities)
```

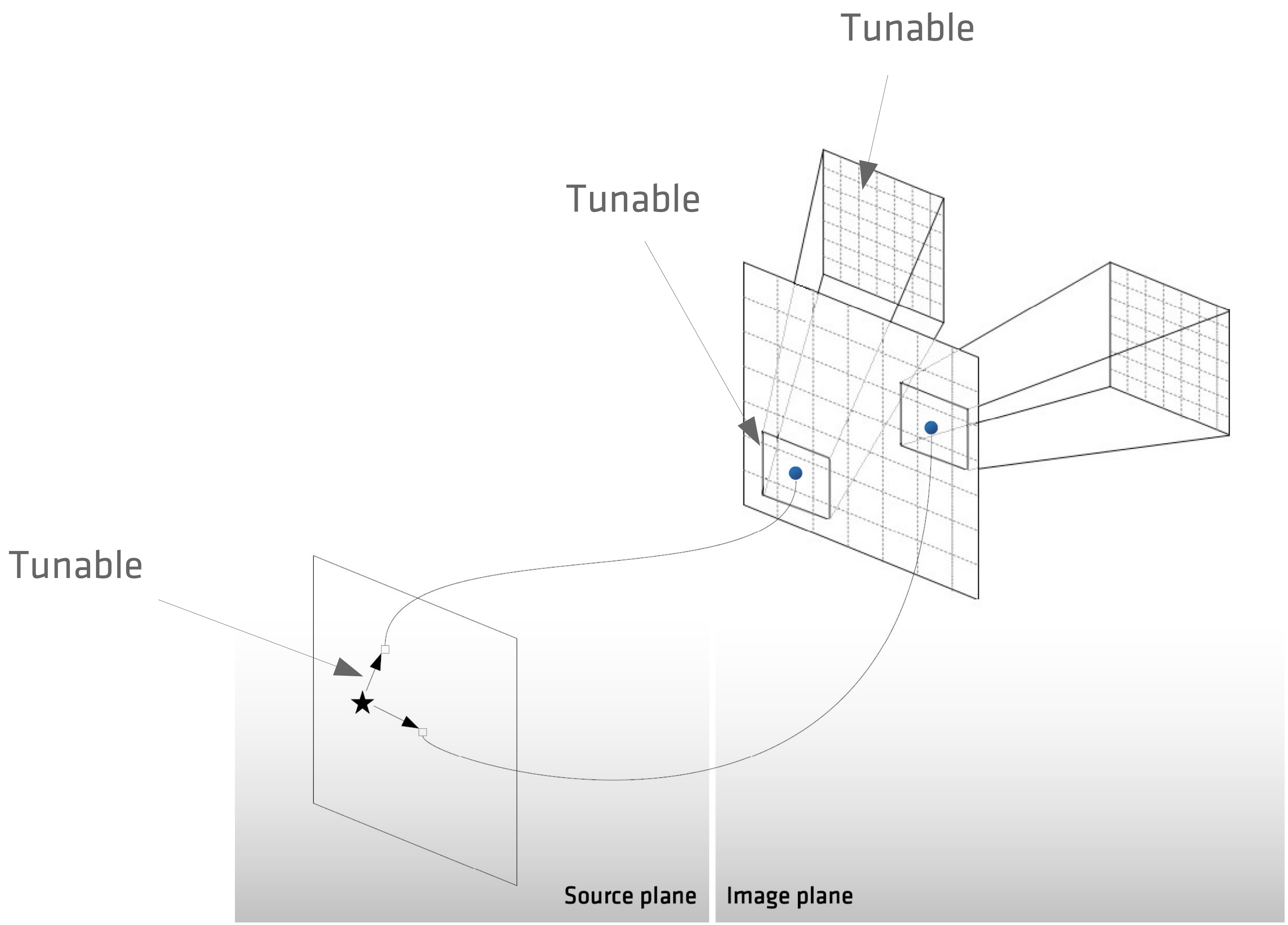
microlensingGW in details

Solving algorithm

- It first considers the macromodel only and centres a window on the source, then pixels it;
- the centre of each pixel is ray-shooted and evaluated against the goodness criterion;
- good pixels whose distance from the source already satisfies the requested precision limit are accepted as solutions;
- the remaining pixels are considered as centres of a new window proportional to the old tile, which is then pixelled again;
- the whole thing is repeated until no more good regions are found, or the pixel size threshold of 10^{-25} *radians* is reached: this allows to find the macroimages;
- the complete lens model is considered: if the user specified around which macroimage to look for microimages, points 1.-5. are repeated around that point; if not, each maroimage is checked against.

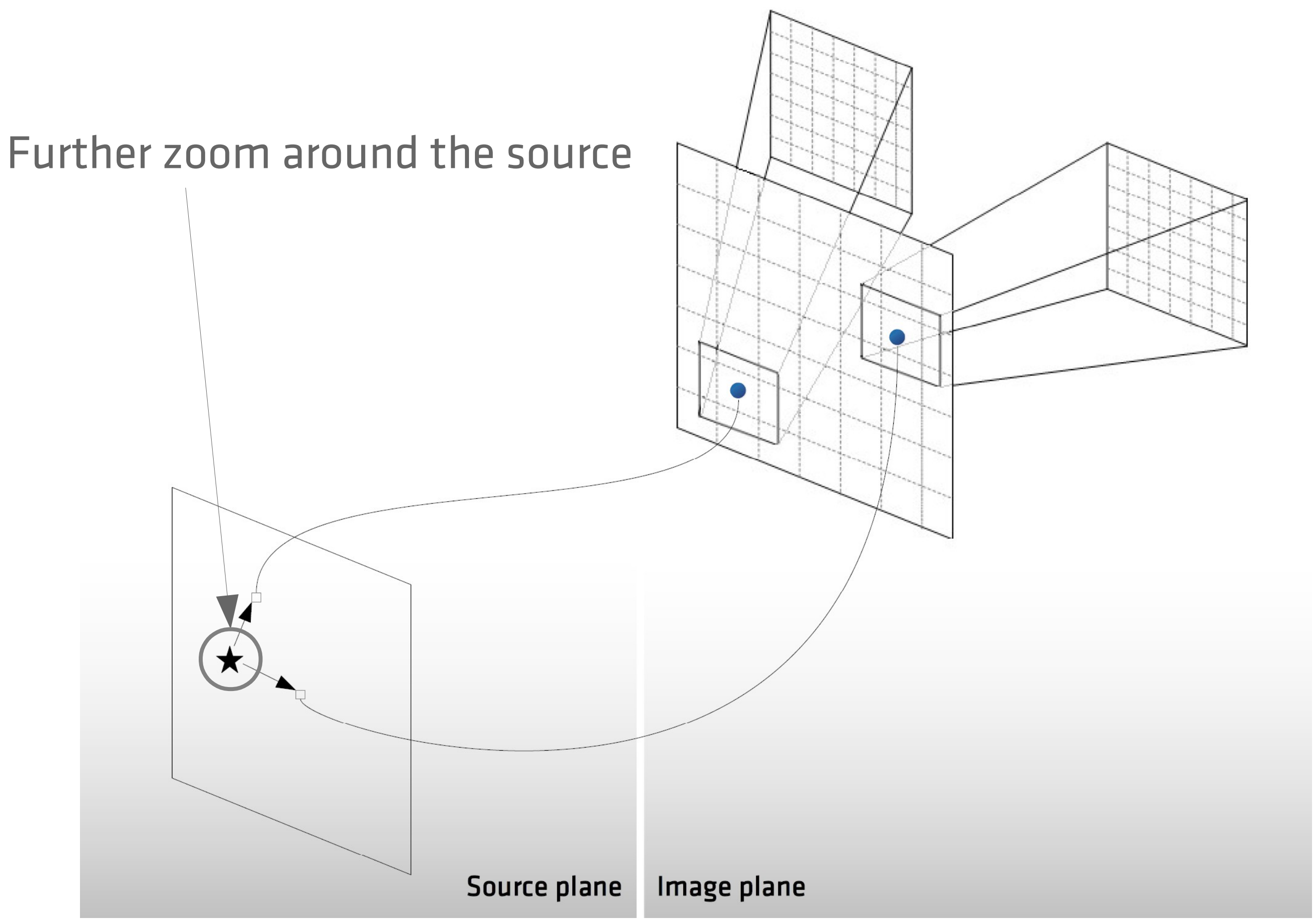
microlensingGW: additional features

Optimization mode



microlensingGW: additional features

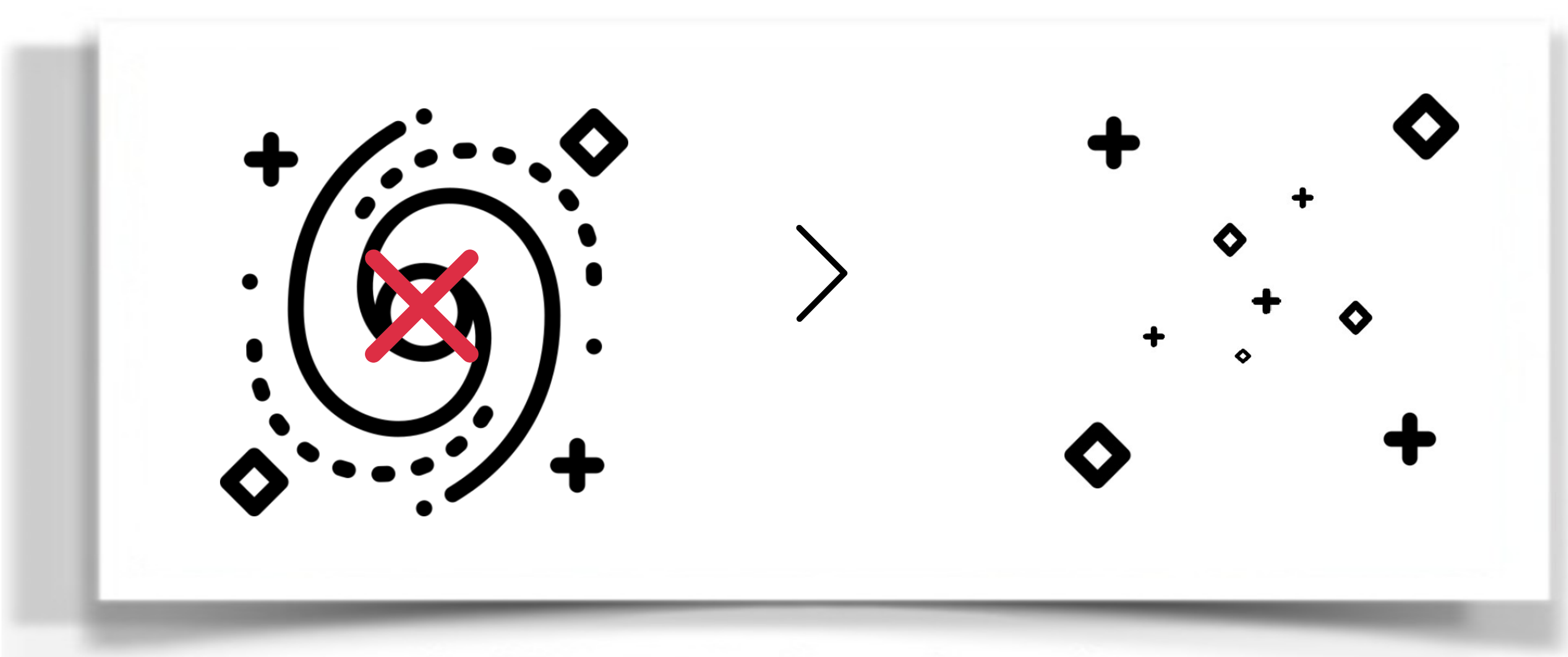
NearSource



microlensingGW: additional features

OnlyMicro

Only solves for the microlenses bakground



microlensingGW: additional features

Complete list of tunable settings

option	optimization	description	default
macro_index		list of potentials which make up the macromodel	None
search_window_macro		size of the first macromodel grid	None
search_window		size of the first complete model grid	None
optimization_window	1	zoom windows multiplying factor, macromodel	2
optimization_window_micro	1	zoom windows multiplying factor, complete model	2
optimization_pixels	1	pixels of the zooms, macromodel	30
optimization_pixels_micro	1	pixels of the zooms, complete model	30
min_dist_requirement	1	minimum rayshooted distance from the source, macromodel	None
min_dist_requirement_micro	1	minimum rayshooted distance from the source, complete model	None
improvement	1	contraction of the minumum distance at each iteration, macromodel	1
improvement_micro	1	contraction of the minumum distance at each iteration, complete model	1
overalp_condition		distance below which solutions are considered overlaps, macromodel	10^{-15}
overalp_condition_micro		distance below which solutions are considered overlaps, complete model	10^{-15}
precision_limit	0	precision of the recovered solutions, all models	10^{-20}
optimisation_precision_limit	1	precision of the recovered solutions, macromodel	10^{-20}
optimisation_precision_limit_micro	1	precision of the recovered solutions, complete model	10^{-20}
NearSource		enables further zoom near the source position	False
only_micro		considers only the microlenses background	False
scaled		converts the output in scaled units	False
scaleFactor		conversion factor	1
img_index		macroimage selected for the zoom	full set