

**intro and tutorial for
Gaussian multi-frequency VLBI analyses
(GaMVAs)**

by Hyeon-Woo Jeong

e-mail:

hwjeong@kasi.re.kr

hwjeongastro@gmail.com

github: <https://github.com/HWJeong1122/gamvas>

Download & Python dependencies

Download: github page (<https://github.com/HWJeong1122/gamvas>)

then, add gamvas directory to PYTHONPATH, e.g.,

```
sys.path.append('/directory/to/gamvas/') # in python
```

```
or, export PYTHONPATH='/directory/to/gamvas/:$PYTHONPATH' # in ~/.bashrc
```

[numpy](#) (pip install numpy)

[pandas](#) (pip install pandas)

[matplotlib](#) (pip install matplotlib)

[uncertainties](#) (pip install uncertainties)

[scipy](#) (pip install scipy)

[sklearn](#) (pip install sklearn)

[astropy](#) (pip install astropy)

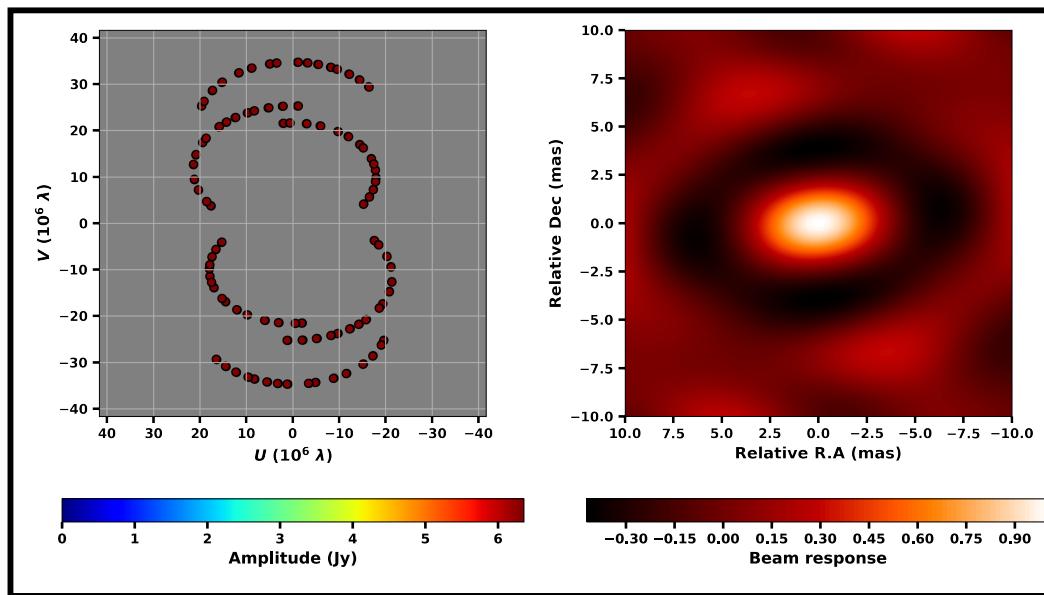
[dynesty](#) (pip install dynesty)

Intro

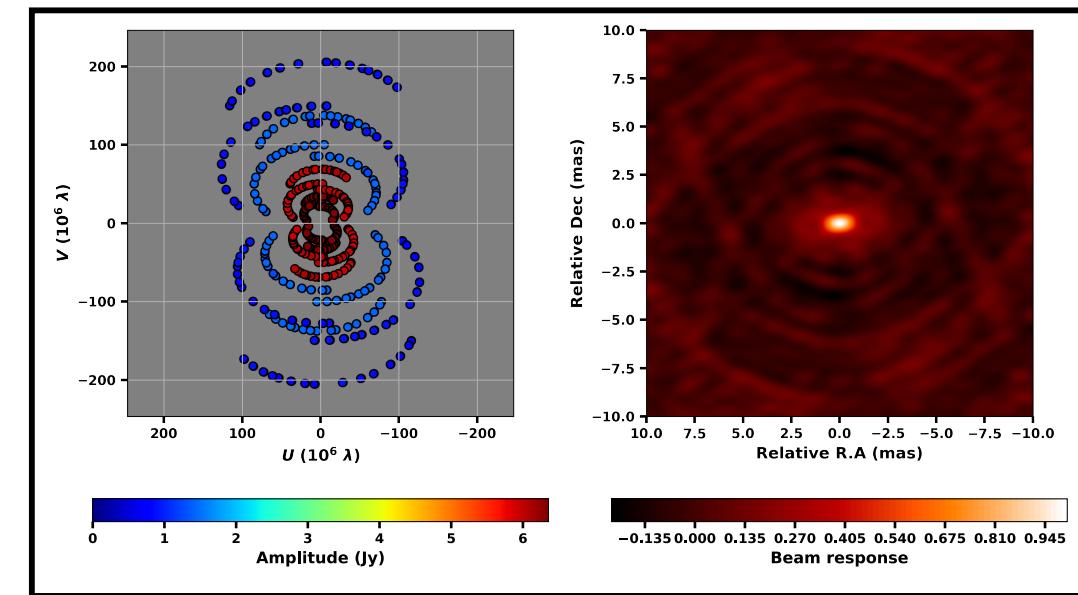
* Project code: p20st02j, p20st02k

** Source: 3C 345 (1641+399)

single-frequency



multi-frequency



Aims

1. Effective geometrical modeling for interferometers consisting small number of antenna
(improved *uv*-coverage)

2. Use of closure quantities
(not supported in Difmap)

3. Direct spectral analysis through co-identified Gaussians

Assumptions

1. Gaussian approximation for AGN jets

2. Consistent jet geometry within radio frequencies

3. Negligible core-shift effect within angular resolution & frequency range

GaMVA follows Bayesian approach via nested sampling

MCMC: estimate likelihood (& therefore, posterior distribution) directly

Nested sampling: estimate evidence based on (remaining) prior volume

$$\frac{p(\theta|x, H)}{\text{probability}} = \frac{\frac{p(x|\theta, H)p(\theta|H)}{\text{likelihood prior distribution}}}{\frac{p(x|H)}{\text{evidence (or, marginal likelihood)}}}$$

x : observed data
 θ : parameters
 H : model

Implementation: DYNESTY ([documentation](#), for more details, see [J.S. Speagle 2020 \(ADS\)](#))

maximizing evidence, $Z \equiv \int_{\Omega_\theta} \mathcal{L}(\theta) \pi(\theta) d\theta = \int_0^1 \mathcal{L}(X) dX$

stopping criteria: $\Delta \ln \hat{Z}_i \equiv \ln(\hat{Z}_i + \Delta \hat{Z}_i) - \ln(\hat{Z}_i) \lesssim \mathcal{L}^{\max} \hat{X}_i$

[^] (**hat**) denotes remaining quantities

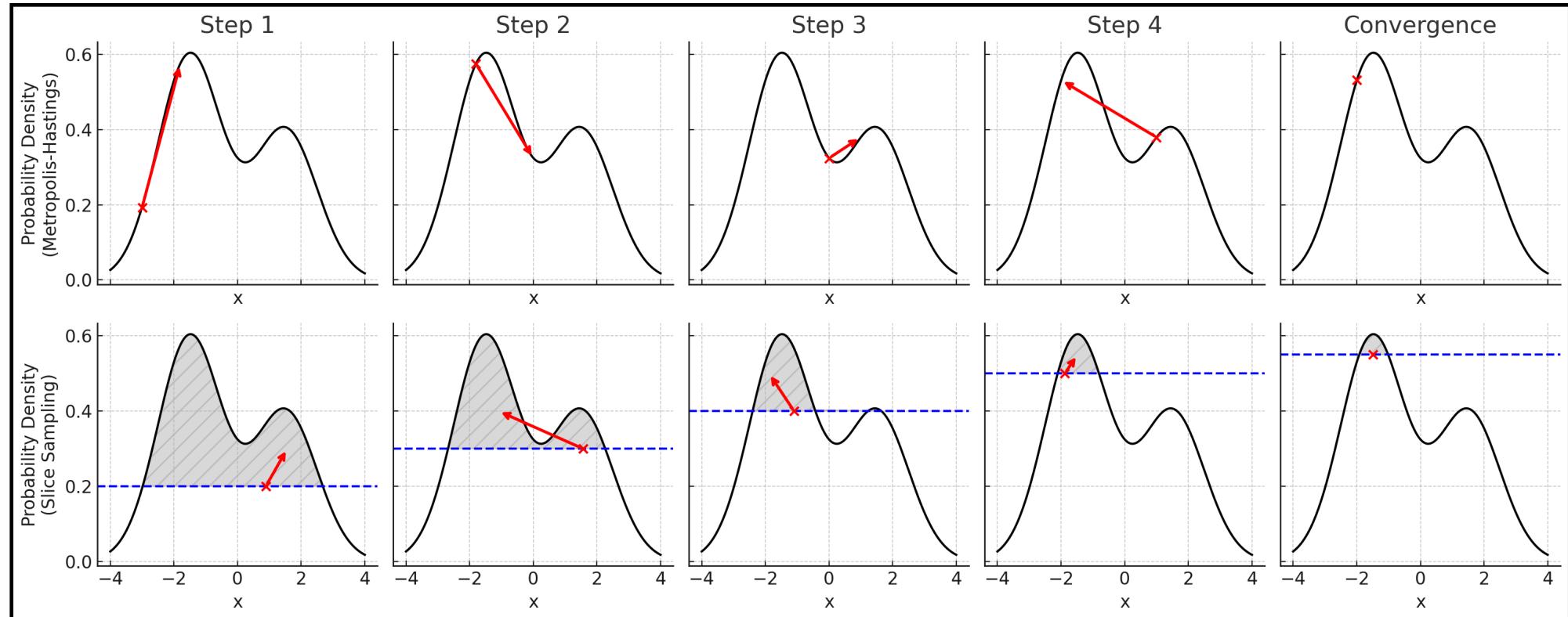
Intro // Methodologies

'Slice sampling' vs. 'random walk'

Nested sampling

Metropolis-Hastings (MCMC)

* An example for 1-D problem



** Blue dashed line denotes auxiliary variable (a threshold)

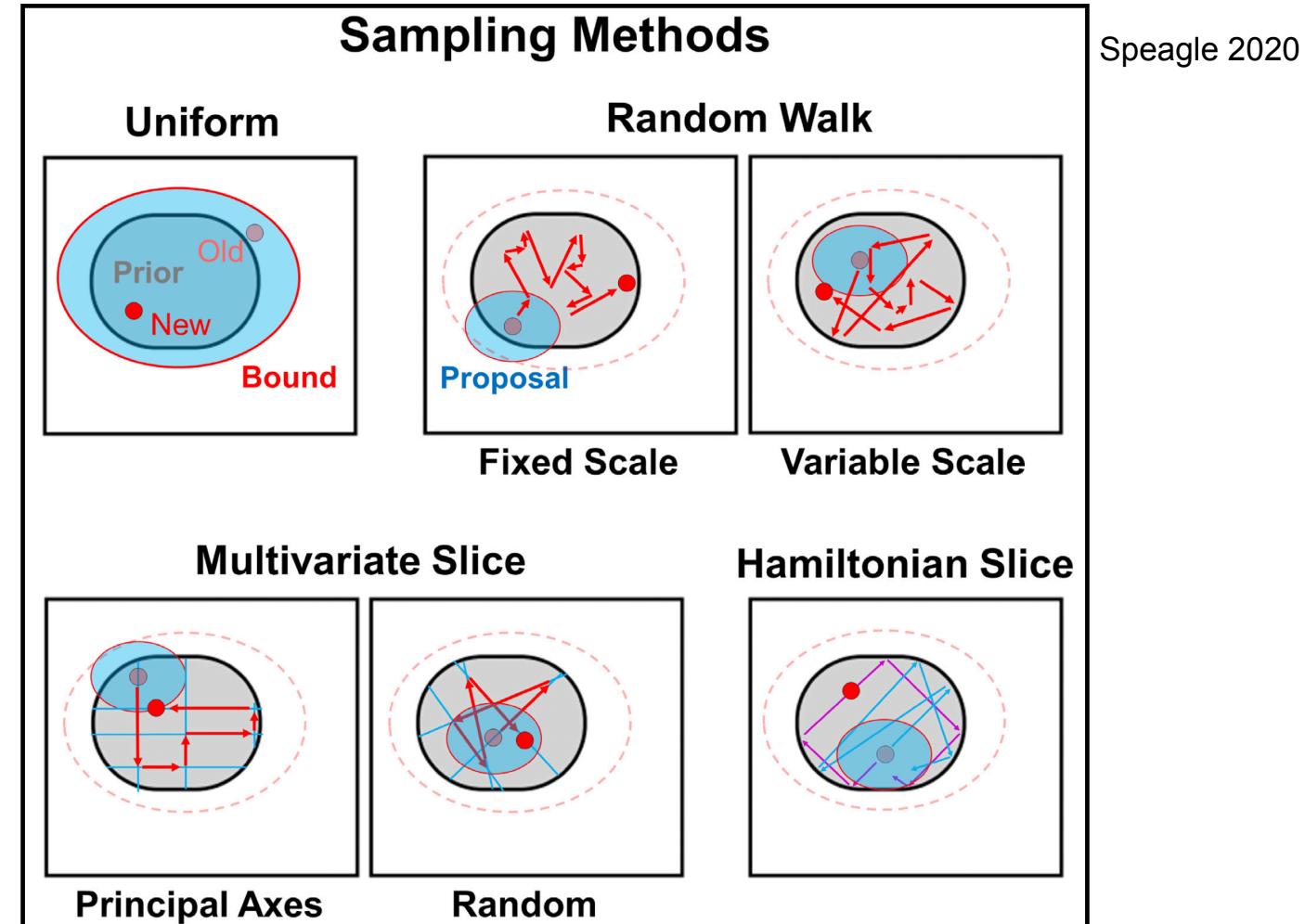
Intro // Methodologies

'Slice sampling' vs. 'random walk'

Nested sampling

Metropolis-Hastings (MCMC)

* An example for 2-D problem

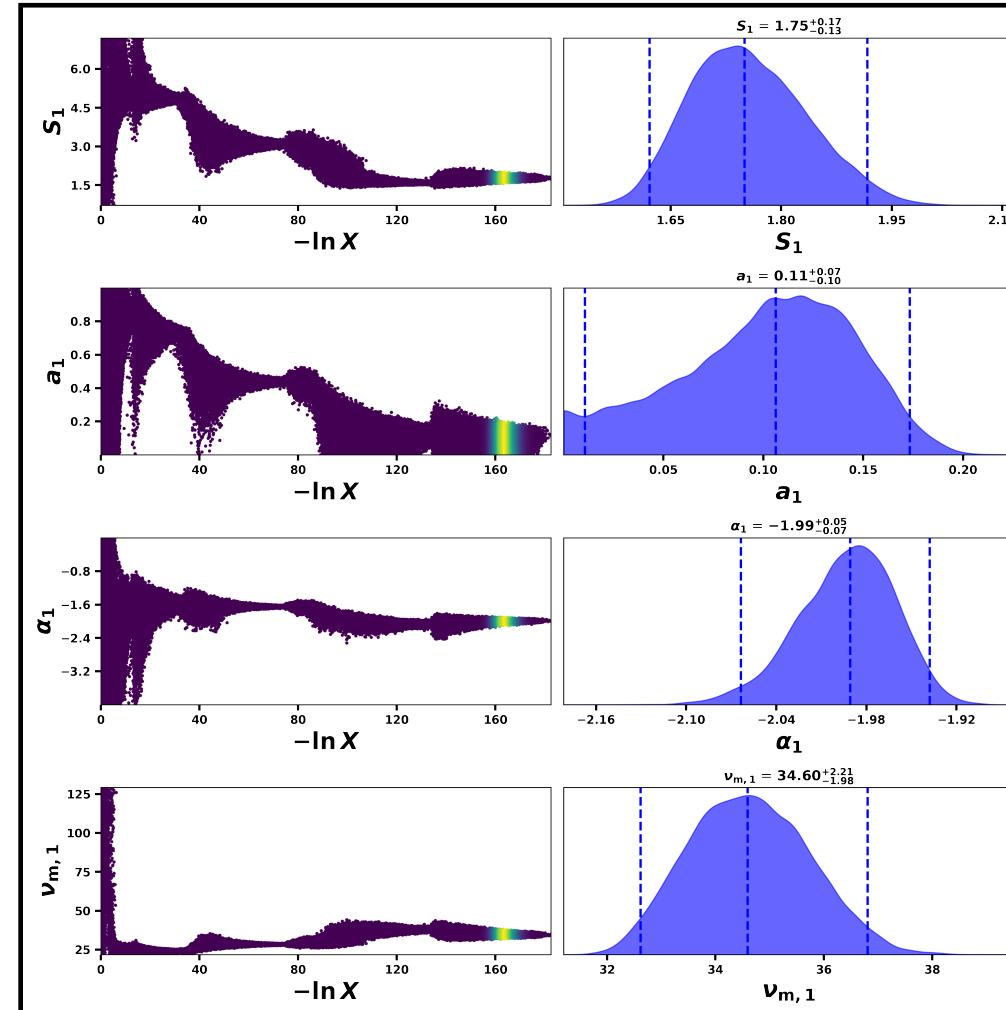


Intro // Methodologies

'Slice sampling' vs. 'random walk'

Nested sampling

Metropolis-Hastings (MCMC)



Intro // visibility of a multi-frequency Gaussian

1. complex visibility in a baseline: $\mathcal{V}(u, v) = \iint I(l, m) e^{-2\pi i(ul+vm)} dl dm$

→ circular Gaussian: $\mathcal{V}_m(u, v; \theta') = S e^{-2\pi a^2(u^2+v^2)} e^{2\pi i(ul_0+vm_0)}$
(single frequency)

$\theta' = (S, a, l_0, m_0)$

S : flux density [Jy]

a : angular size [mas]

l_0 : right ascension offset [mas]

m_0 : declination offset [mas]

2. Synchrotron self-absorption :
(Turler+1999)

$$S(\nu; S_m, \nu_m, \alpha) = S_m \left(\frac{\nu}{\nu_m} \right)^{2.5} \times \frac{1 - \exp(-\tau_m (\nu/\nu_m)^{\alpha-2.5})}{1 - e^{-\tau_m}}$$

$\theta = (S, a, l_0, m_0, \alpha, \nu_m)$

α : spectral index

ν_m : turnover frequency [GHz]

ν : observing frequency [GHz]

3. Multi-frequency Gaussian :

$$\mathcal{V}_m(u_\nu, v_\nu, \nu; \theta) = S(\nu; S_m, \nu_m, \alpha) \times e^{-2\pi a^2(u_\nu^2+v_\nu^2)} e^{2\pi i(u_\nu l_0+v_\nu m_0)}$$

Intro // objective function

1. Modeling script aims to minimize **Bayesian information crietria (BIC)**

$$\text{BIC} = k \ln(n) - 2 \ln(\mathcal{L})$$

k : degree of freedom

n : the number of data points

2. Likelihood \mathcal{L} with independent normal Gaussian noise is given by

$$\mathcal{L} = \prod_{i=1}^n \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(-\frac{1}{2} \left(\frac{\text{model}_i - \text{data}_i}{\sigma_i}\right)^2\right) \quad \sigma_i : \text{uncertainty of } i^{\text{th}} \text{ data}$$

then,

$$\text{BIC} = k \ln(n) + \sum_{i=1}^n \left[\left(\frac{\text{model}_i - \text{data}_i}{\sigma_i} \right)^2 + \ln(2\pi\sigma_i^2) \right]$$

Example run // f24sl02b

```
import gamvas as gv

# Load uv-fits files (map range: 22 mas)
uvf1 = gv.load.open_fits(path=path_ufv, file=file_k, mrng=22 * gv.mas)
uvf2 = gv.load.open_fits(path=path_ufv, file=file_q, mrng=22 * gv.mas)
uvf3 = gv.load.open_fits(path=path_ufv, file=file_w, mrng=22 * gv.mas)
uvf4 = gv.load.open_fits(path=path_ufv, file=file_d, mrng=22 * gv.mas)

# select='ll': load ll-polarization, uvw='u': use uniform visibility weighting
uvf1.load_ufv(select='ll', uvw='u')
uvf2.load_ufv(select='ll', uvw='u')
uvf3.load_ufv(select='ll', uvw='u')
uvf4.load_ufv(select='ll', uvw='u')

# uv-average in 60s (weighted-average)
uvf1.uvave(uvave=60, scanlen=300)
uvf2.uvave(uvave=60, scanlen=300)
uvf3.uvave(uvave=60, scanlen=300)
uvf4.uvave(uvave=60, scanlen=300)

# set multi-frequency uvf
uvfs = [uvf1, uvf2, uvf3, uvf4]
uvall = gv.utils.set_ufv(uvfs, type='mf')

# set data terms and weights
ftype = ['amp', 'clamp', 'clphs']
fwght = [0.01, 1, 1]    # low-weight to visibility amplitude

# set the number of CPU core (for multi-processing)
ncpu = N    # to use maximum number, ncpu = os.cpu_count()
```

Example run // f24sl02b

The following codes are optional (example)
! some commands now may be shooting problems

```
# flag visibilities having sigma higher than f
uvf.flag_sigma(sigma=f)
```

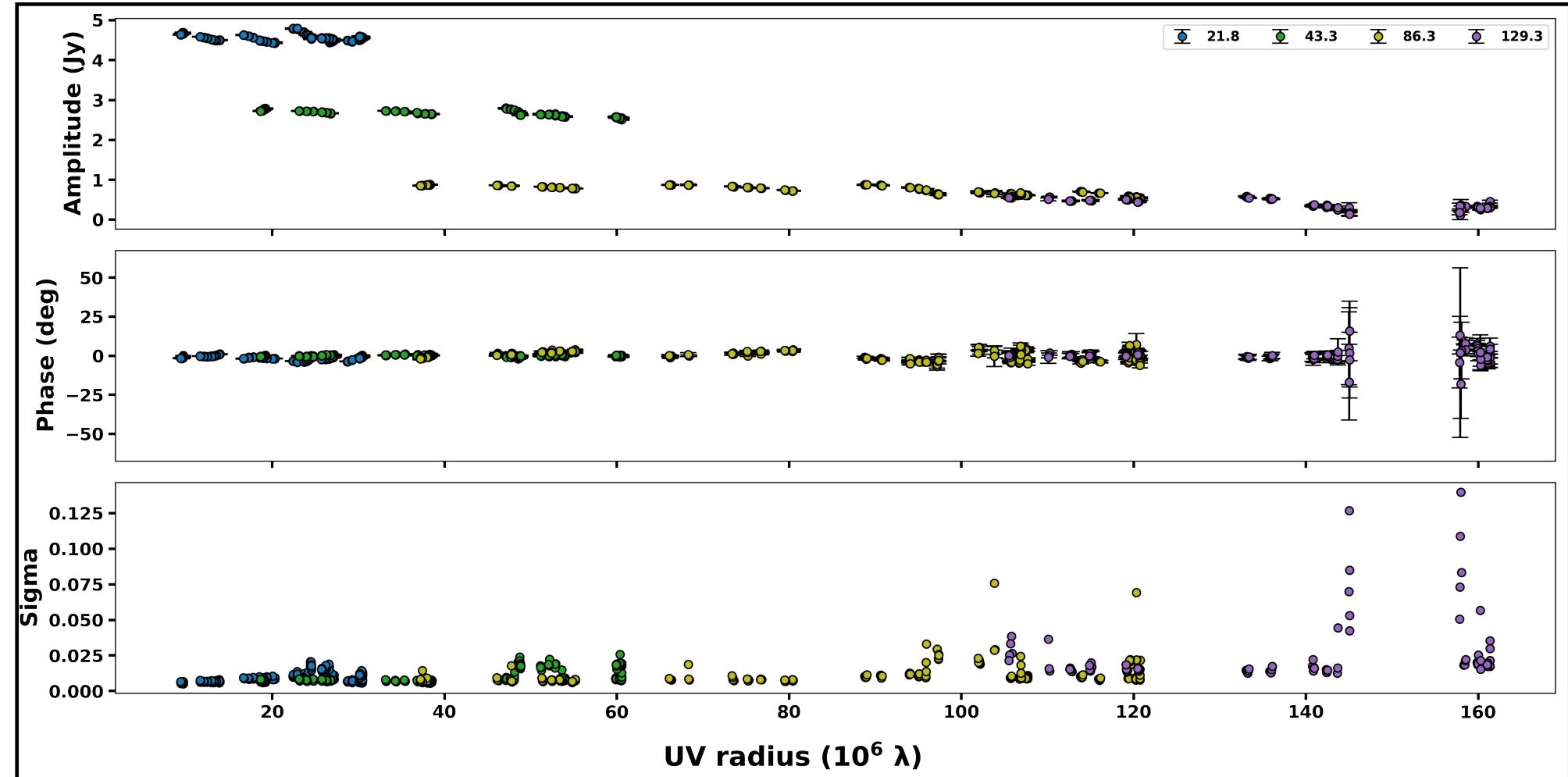
```
# flag visibilities having signal-to-noise ratio lower than f
uvf.flag_snr(snrflag=f)
```

```
# add fractional (to visibility amplitude) error to current sigma by f %
uvf.add_error_fraction(fraction=0.0f)
```

```
# modify error by a factor of f
uvf.add_error_factor(factor=f)
```

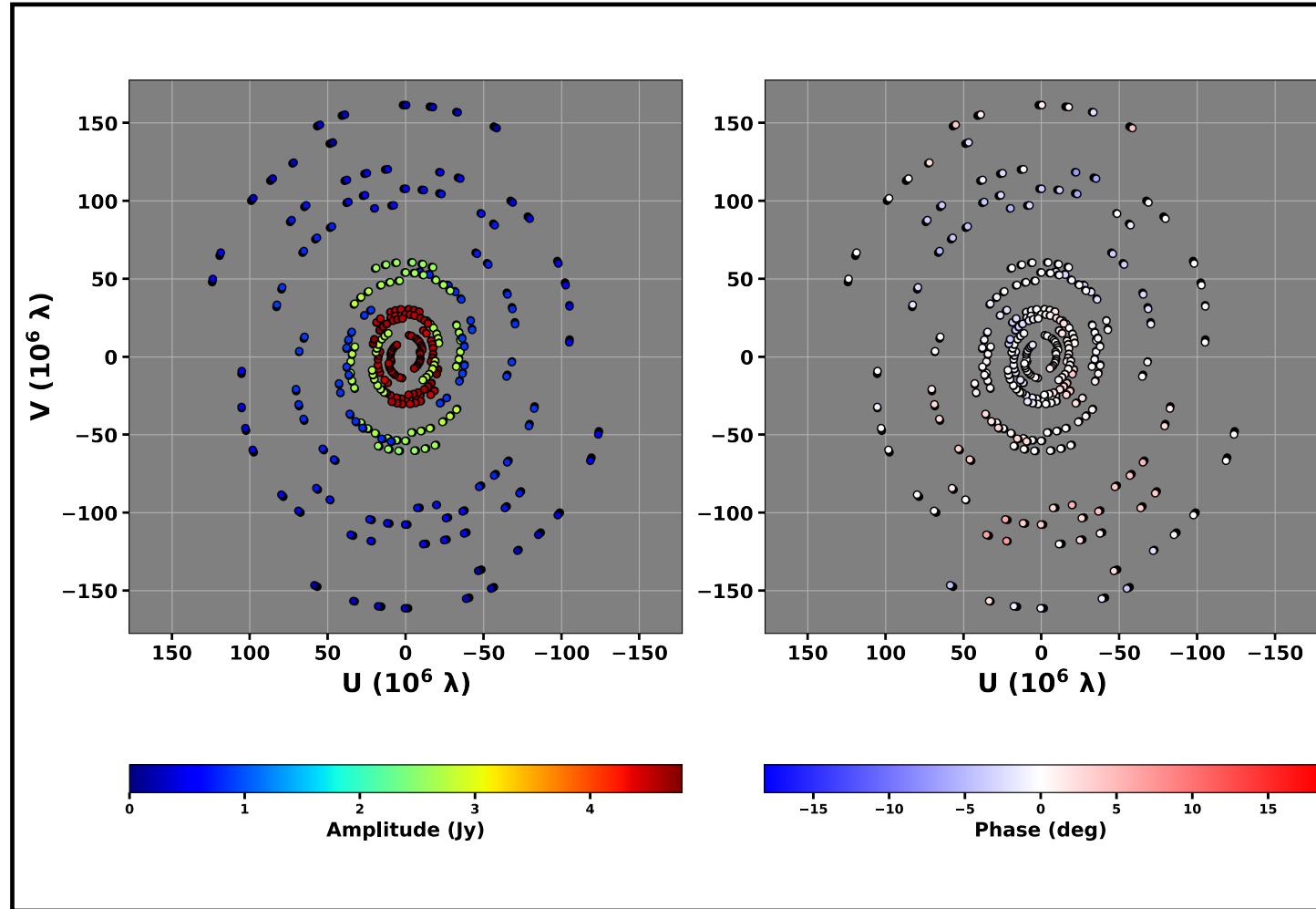
Example run // f24sl02b

```
# check visibility data  
uvall.ploter.draw_radplot(uvall, plotimg=True) # save_path=, save_name=)
```



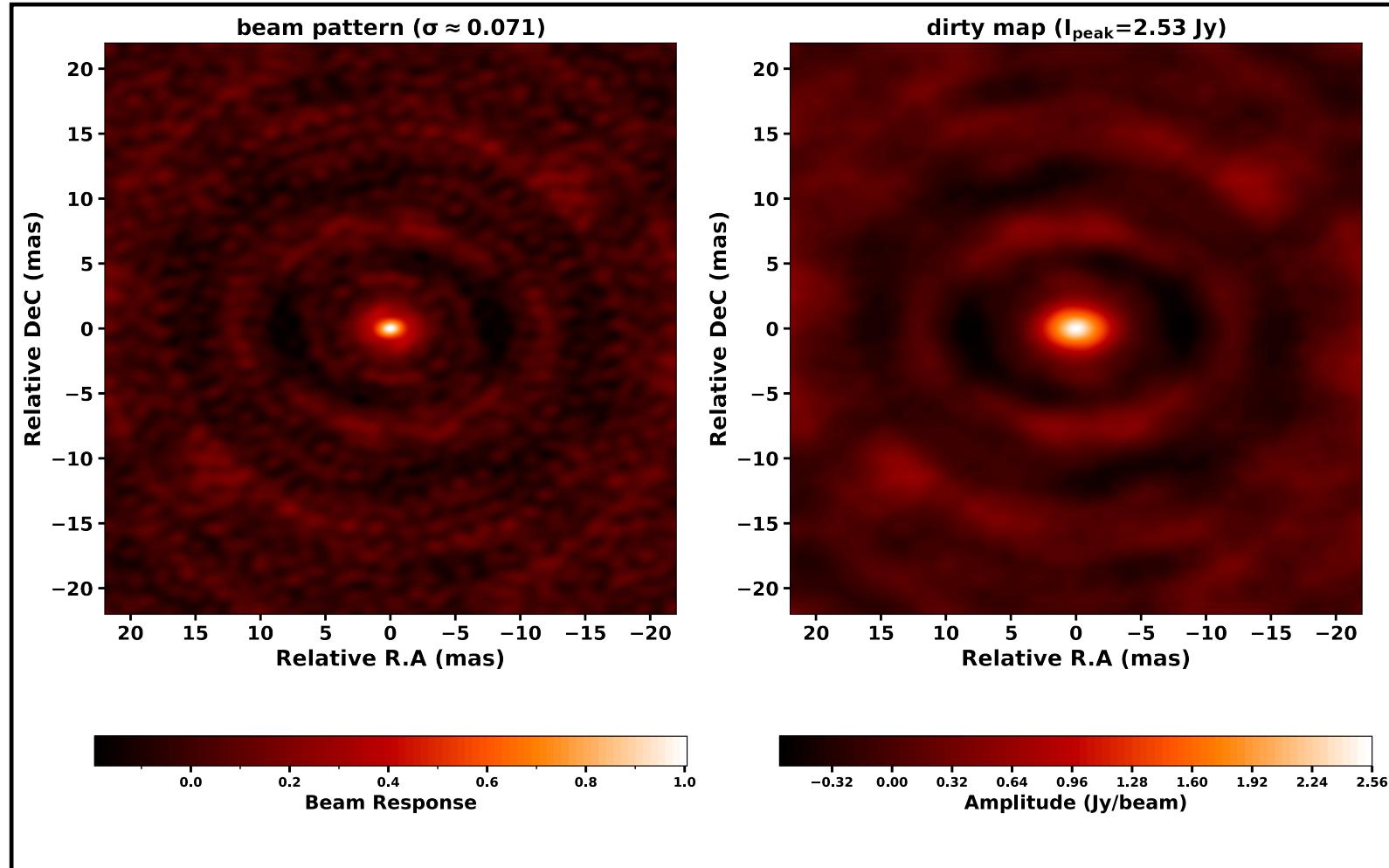
Example run // f24sl02b

```
# check visibility data  
uvall.ploter.draw_uvcover(uvall, plotimg=True) # save_path=, save_name=)
```



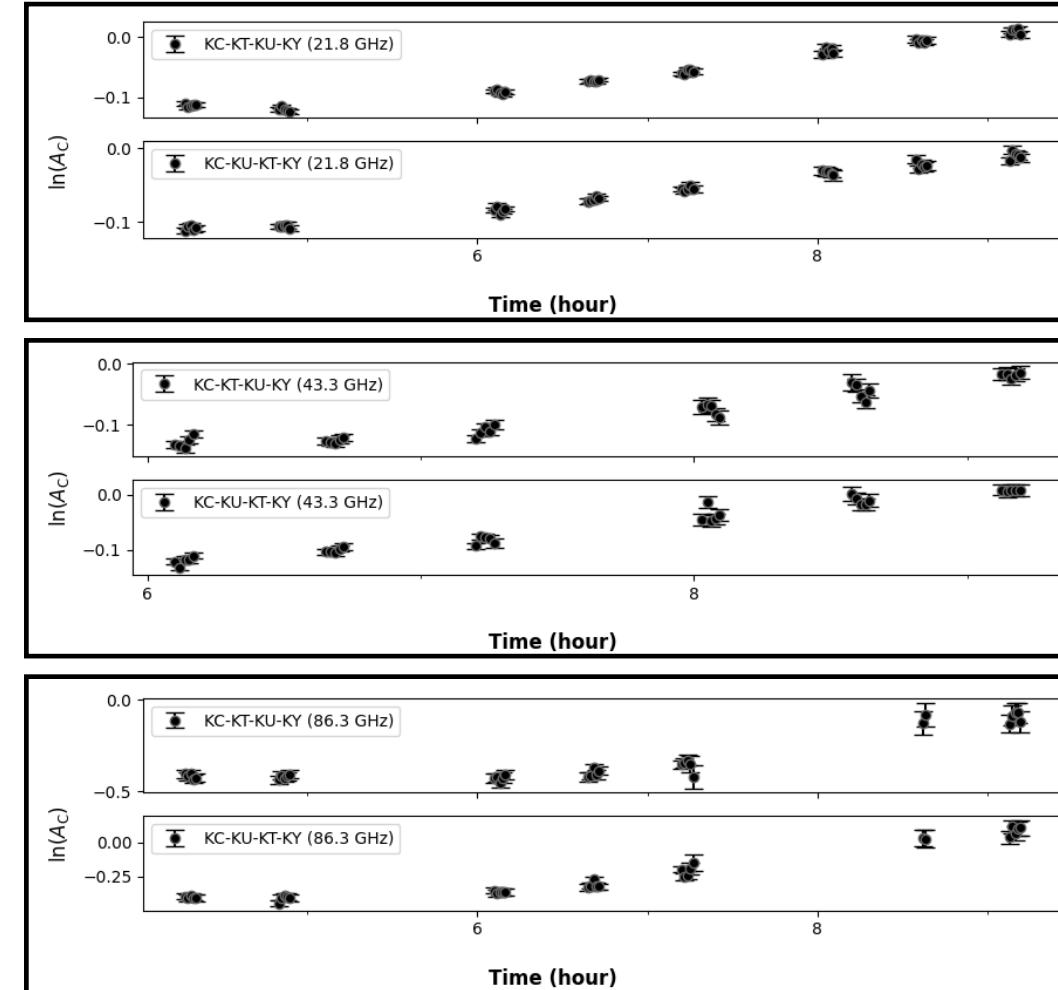
Example run // f24sl02b

```
# check visibility data  
uvall.ploter.draw_dirtymap(uvall, plotimg=True) # save_path=, save_name=)
```



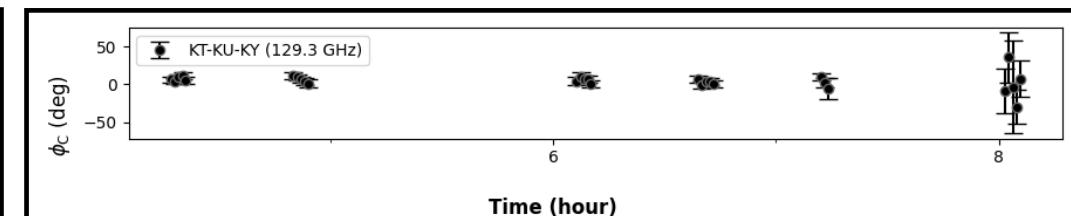
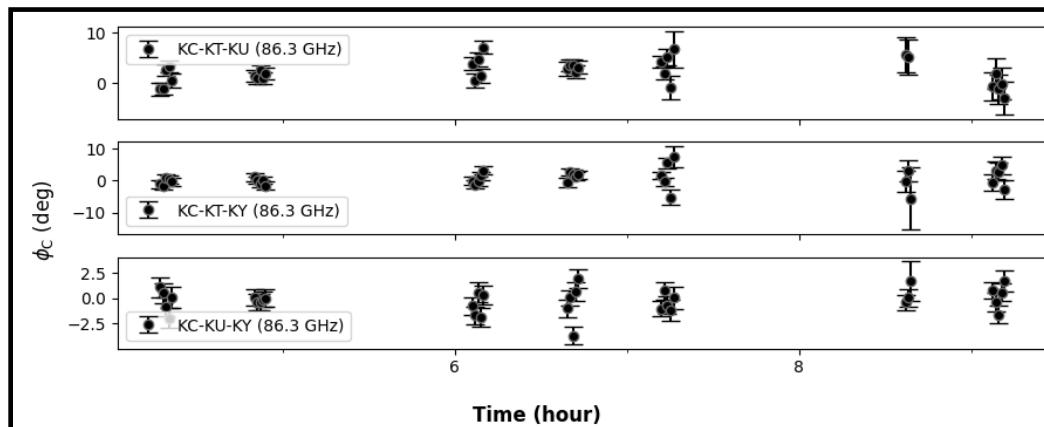
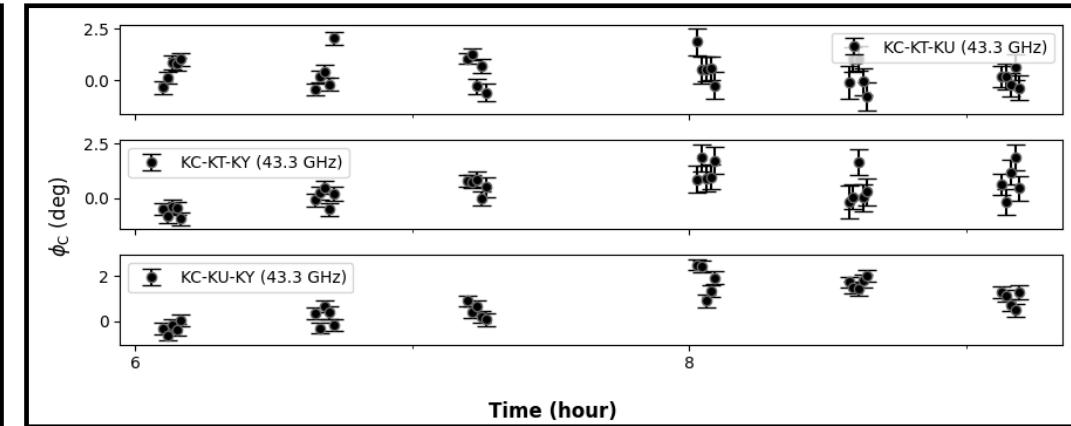
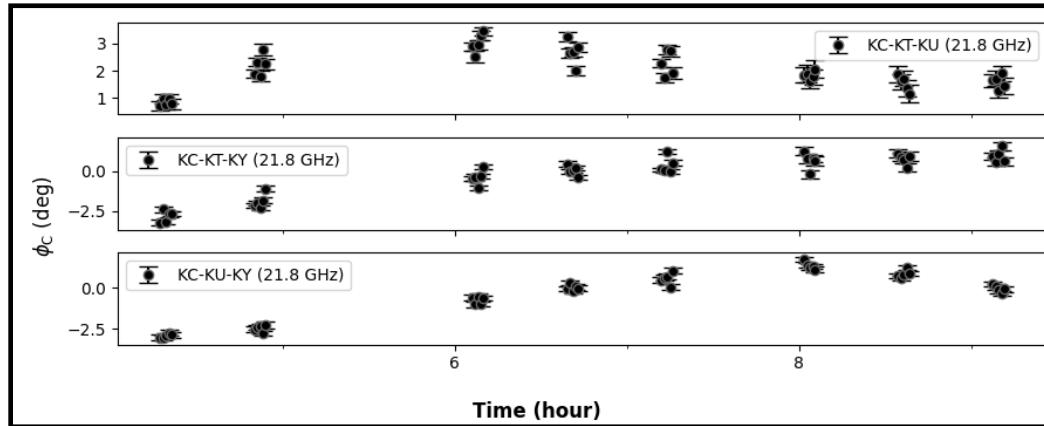
Example run // f24sl02b

```
# check visibility data  
uvall.ploter.draw_closure(type='clamp', plotimg=True) # save_path=, save_name=)
```



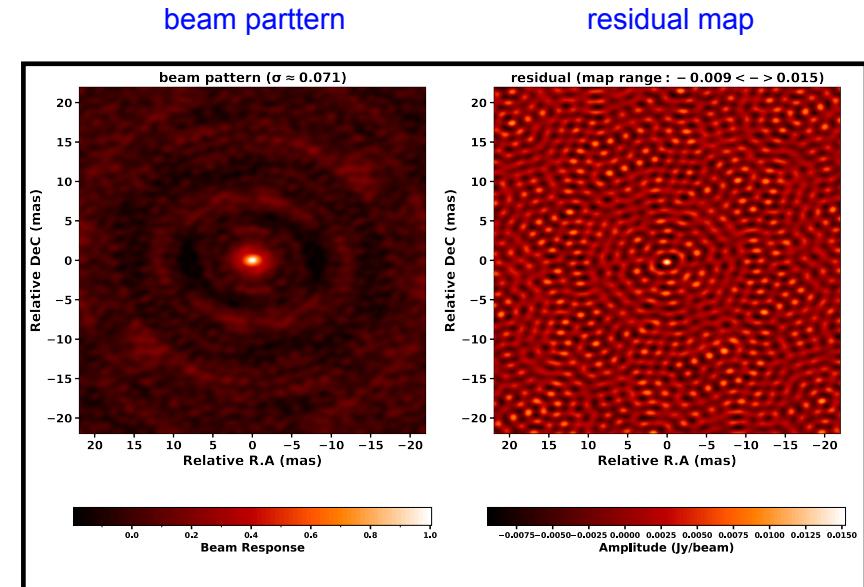
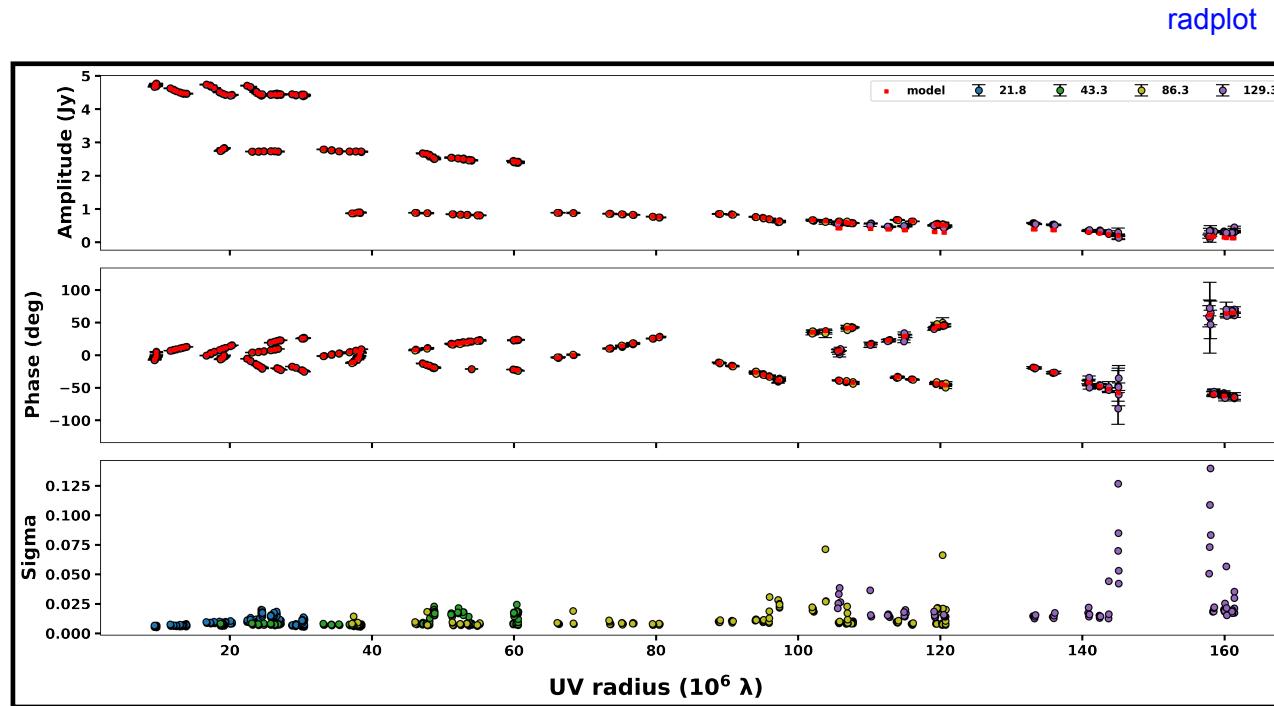
Example run // f24sl02b

```
# check visibility data  
uvall.ploter.draw_closure(type='clphs', plotimg=True) # save_path=, save_name=)
```



Example run // f24sl02b

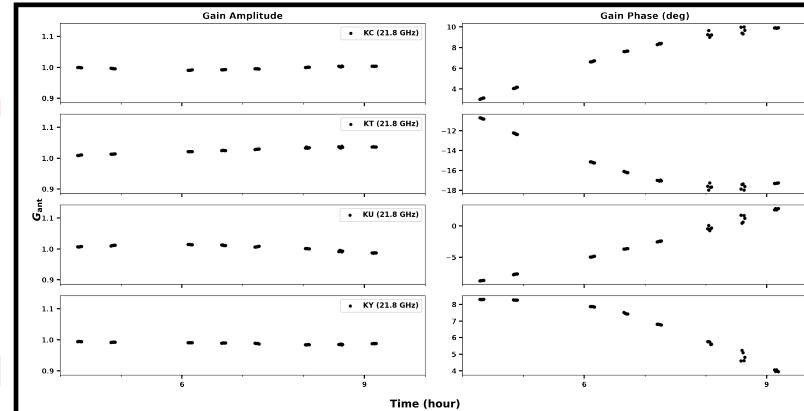
```
# run model-fitting * Please see example_run.py for detailed attribute setting
mfu = gv.modeling.modeling(
    uvfs=uvfs, select='ll', sampler='slice', bound='multi',
    ftype=ftype, fwght=fwght, doampcal=True, dophscal=True, ncpu=ncpu,
    ...
)
mfu.run()
```



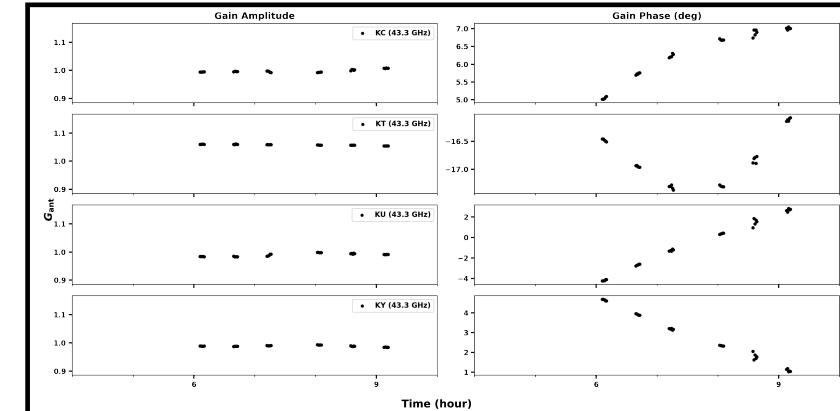
Example run // f24sl02b

```
# run model-fitting                                * Please see example_run.py for detailed attribute setting !
mfu = gv.modeling.modeling(
    uvfs=uvfs, select='ll', sampler='slice', bound='multi',
    ftype=ftype, fwght=fwght, doampcal=True, dophscal=True, ncpu=ncpu,
    ...
)
mfu.run()
```

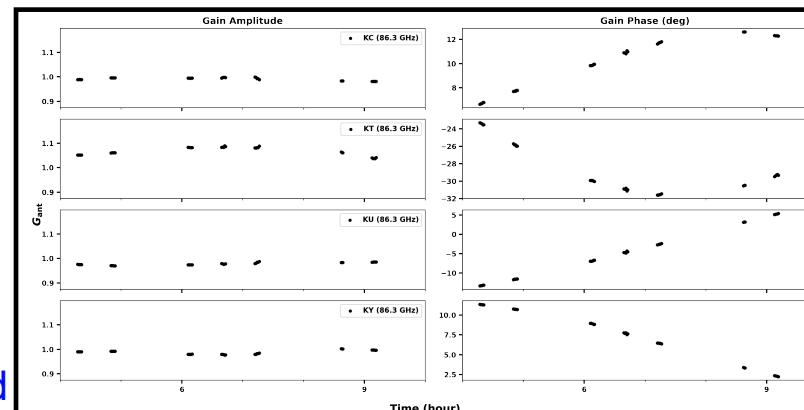
complex gain
after self-calibration



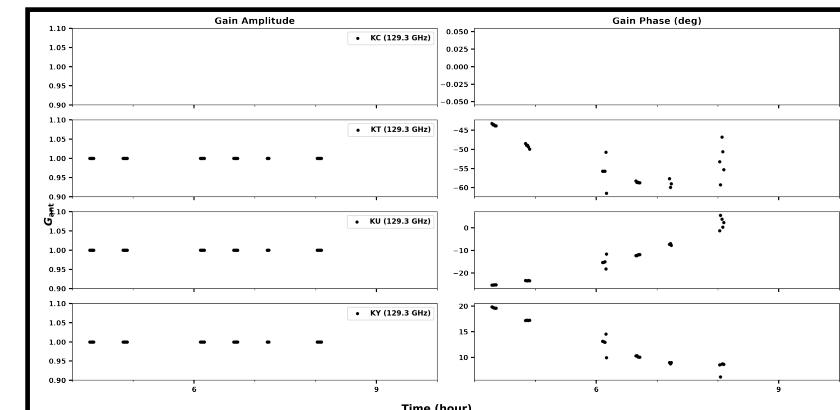
K-band



Q-band



W-band

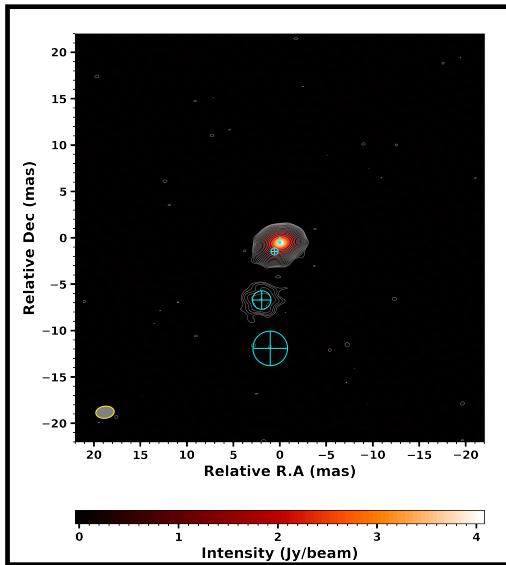


D-band

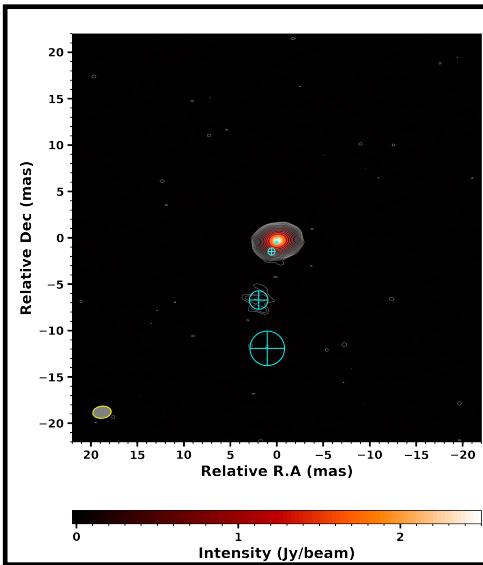
Example run // f24sl02b

```
# run model-fitting
mfu = gv.modeling.modeling(
    uvfs=uvfs, select='ll', sampler='slice', bound='multi',
    ftype=ftype, fwght=fwght, doampcal=True, dophscal=True, ncpu=ncpu,
    ...
)
mfu.run()
```

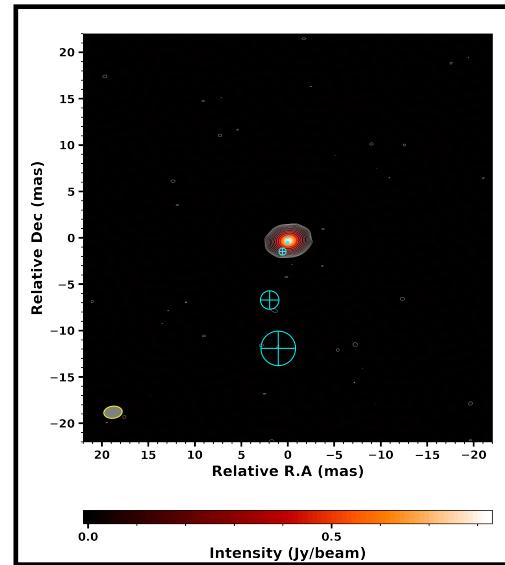
K-band



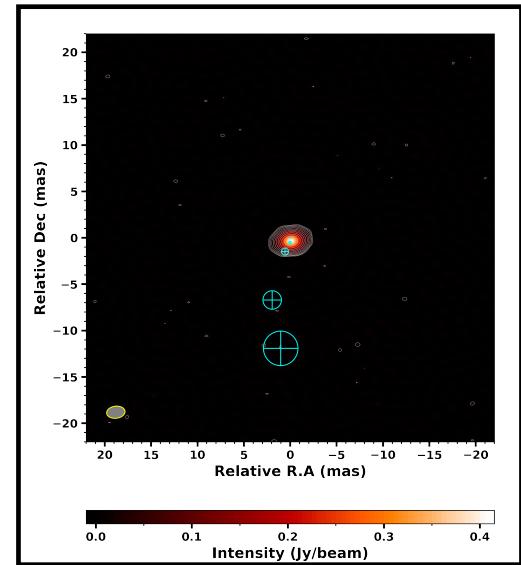
Q-band



W-band



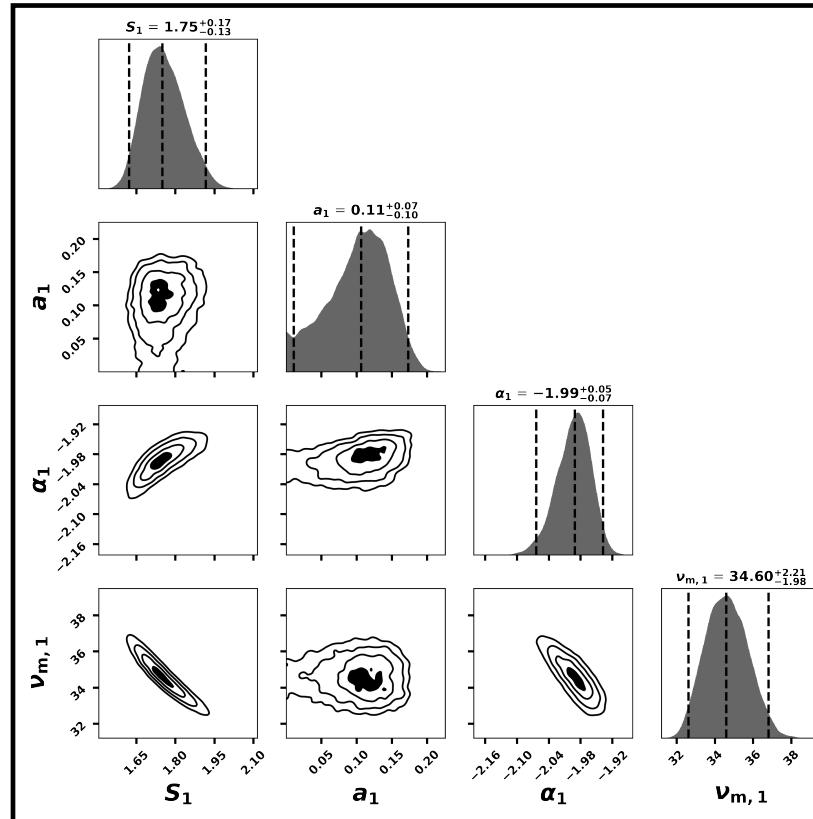
D-band



Example run // f24sl02b

```
# run model-fitting
mfu = gv.modeling.modeling(
    uvfs=uvfs, select='ll', sampler='slice', bound='multi',
    ftype=ftype, fwght=fwght, doampcal=True, dophscal=True, ncpu=ncpu,
    ...
)
mfu.run()
```

corner plot of model #1



trace plot of model #2

