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)

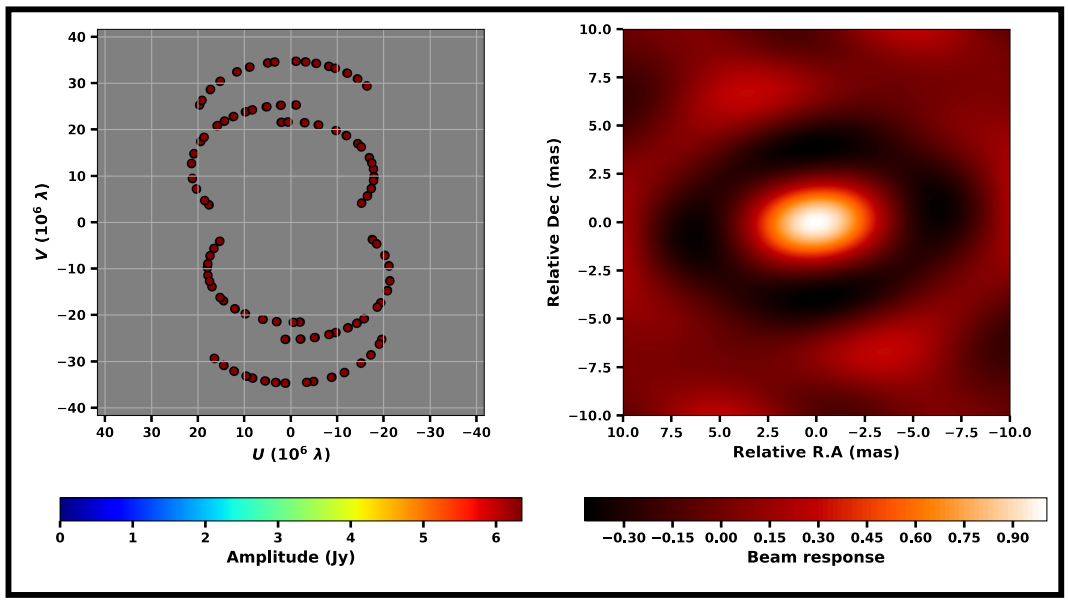
dynesty (pip install dynesty)

Intro

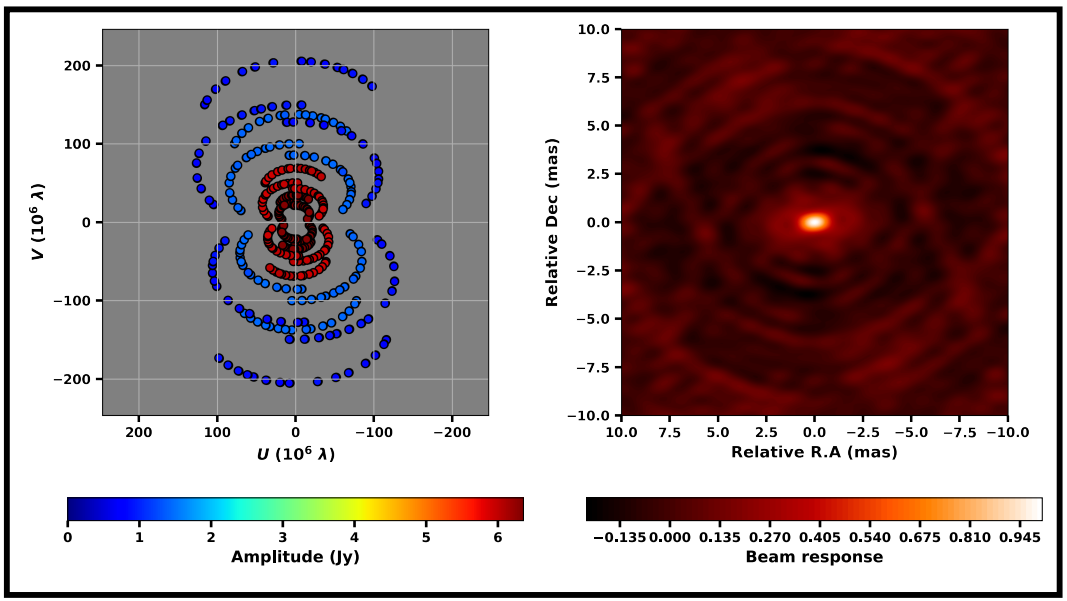
* Project code: p20st02j, p20st02k

** 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

GaMVAs follows Bayesian approach via nested sampling

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

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

$$\underbrace{p(\theta \mid x, H)}_{\text{probability}} = \frac{\overbrace{p(x \mid \theta, H)}^{\text{likelihood}}\ \overbrace{p(\theta \mid H)}^{\text{prior distribution}}}{\underbrace{p(x \mid H)}_{\substack{\text{evidence} \\ \text{(or, marginal likelihood)}}}}$$

x: observed data
θ: parameters
H: model

Implemetation: DYNESTY (documentation, for more details, see J S. Speagle 2020 (ADS))

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

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

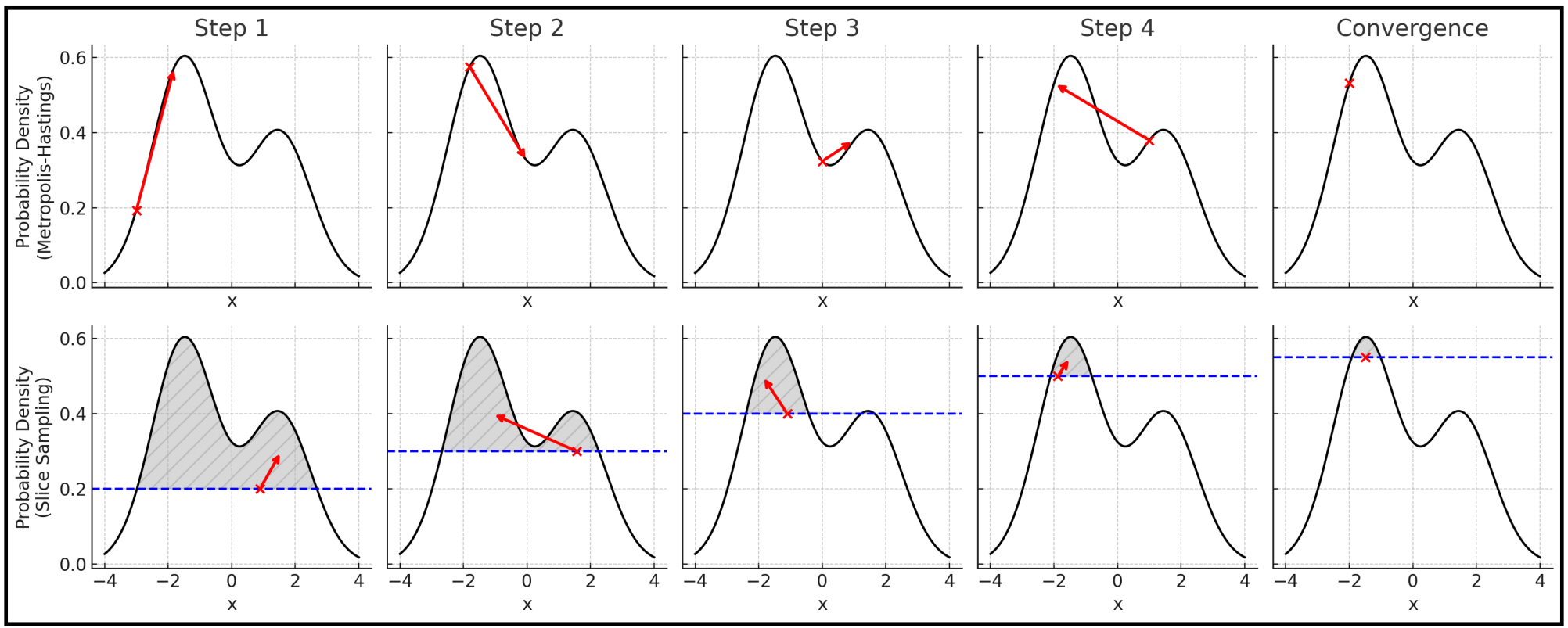
ˆ **(hat)** denotes remaining quantities

Intro // Methodologies

'Slice sampling' vs. 'random walk'

Nested sampling Metropolis-Hastings (MCMC)

* An example for 1-D problem

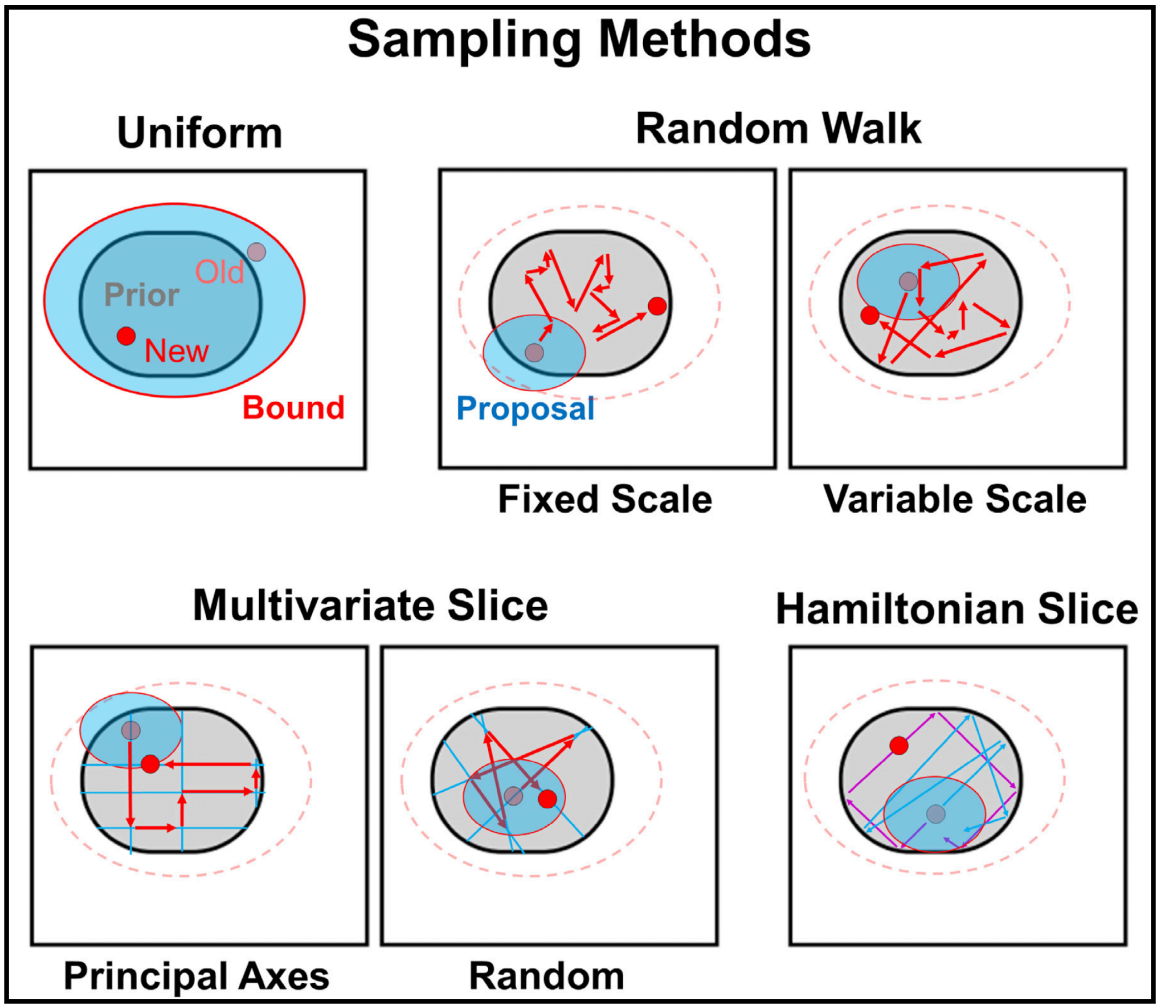


** Blue dashed line denotes auxiliary variable (a threshold)

'Slice sampling' vs. 'random walk'

Nested sampling Metropolis-Hastings (MCMC)

* An example for 2-D problem



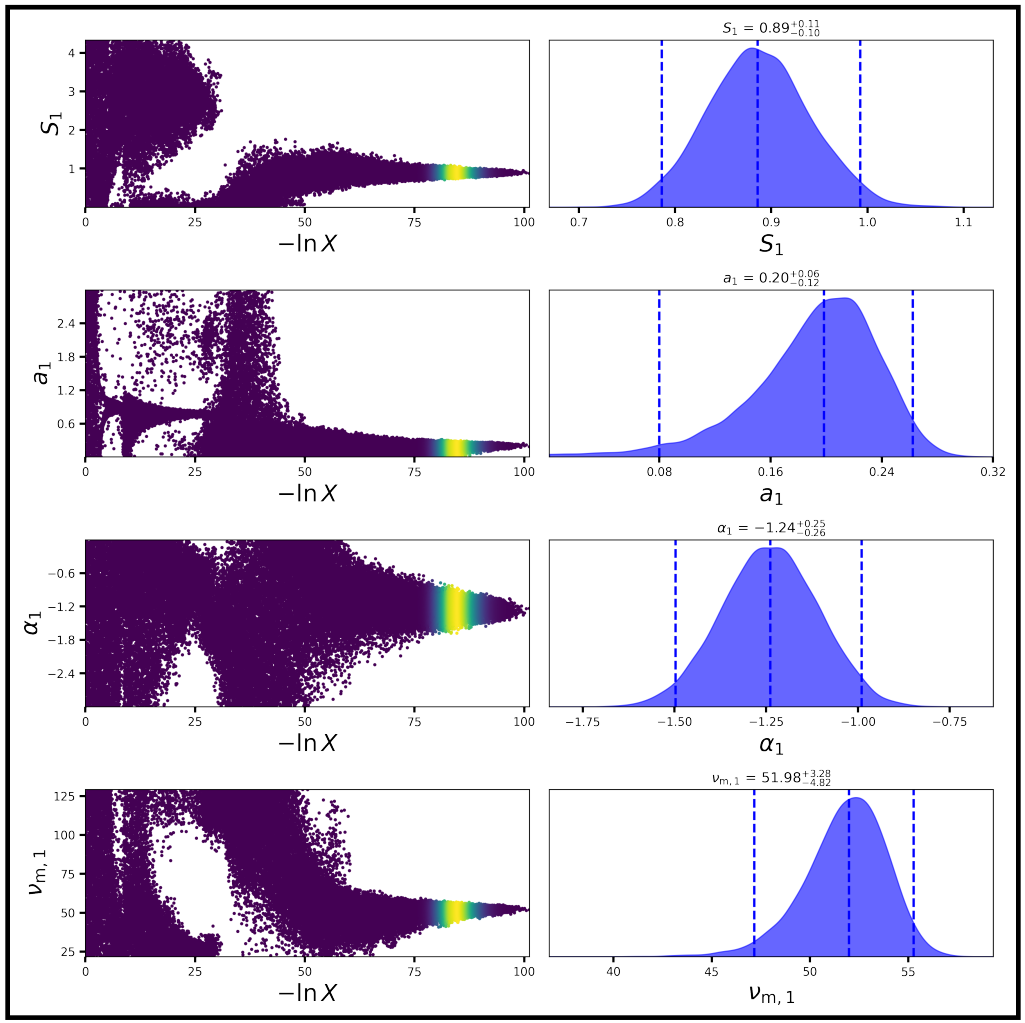
Speagle 2020

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: $$\mathscr{V}(u,v) = \iint I(l,m)\, e^{-2\pi i(ul+vm)}\, dl\, dm$$

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

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

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)

$$S(\nu; S_\mathrm{m}, \nu_\mathrm{m}, \alpha) = S_\mathrm{m} \left(\frac{\nu}{\nu_\mathrm{m}}\right)^{2.5} \times \frac{1-\exp(-\tau_\mathrm{m}(\nu/\nu_\mathrm{m})^{\alpha-2.5})}{1-e^{-\tau_\mathrm{m}}}$$

$\theta = (S,\ a,\ l_0,\ m_0,\ \alpha,\ \nu_\mathrm{m})$

α : spectral index

ν_m : turnover frequency [GHz]

ν : observing frequency [GHz]

3. Multi-frequency Gaussian :

$$\mathscr{V}_m(u_\nu, v_\nu, \nu; \theta) = S(\nu; S_\mathrm{m}, \nu_\mathrm{m}, \alpha) \times e^{-2\pi a^2(u_\nu^2+v_\nu^2)} e^{2\pi i(u_\nu l_0 + v_\nu m_0)}$$

Intro // objective function

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

$$\mathrm{BIC} = k\ln(n) - 2\ln(\mathscr{L})$$

k : degree of freedom

n : the number of data points

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

$$\mathscr{L} = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(-\frac{1}{2}\left(\frac{\mathrm{model}_i - \mathrm{data}_i}{\sigma_i}\right)^2\right)$$

σ_i : uncertainty of i^{th} data

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

Intro // caveat

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

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

→ original objective function should be negative log-likelihood (NLL)

Our objective function (intended for NLL) is $\mathrm{BIC} = k \ln(n) + \sum_{i=1}^{n} \left[\left(\frac{\mathrm{model}_i - \mathrm{data}_i}{\sigma_i} \right)^2 + \ln(2\pi\sigma_i^2) \right]$

by assuming that this form not affect the results significantly

Example run // f24sl02b

** Please see example_run.py for detailed attribute setting*

```
import gamvas as gv

# Load uv-fits files (map range: 22 mas)
uvf1 = gv.load.open_fits(path=path_uvf, file=file1, mrng=22 * gv.mas)
uvf2 = gv.load.open_fits(path=path_uvf, file=file2, mrng=22 * gv.mas)
uvf3 = gv.load.open_fits(path=path_uvf, file=file3, mrng=22 * gv.mas)
uvf4 = gv.load.open_fits(path=path_uvf, file=file4, mrng=22 * gv.mas)

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

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

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

# set data terms and weights
ftype = ['amp', 'clamp', 'clphs']
fwght = [0.1, 1, 1]    # low-weight to visibility amplitude

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

Example run // f24sl02b

* Please see example_run.py for detailed attribute setting

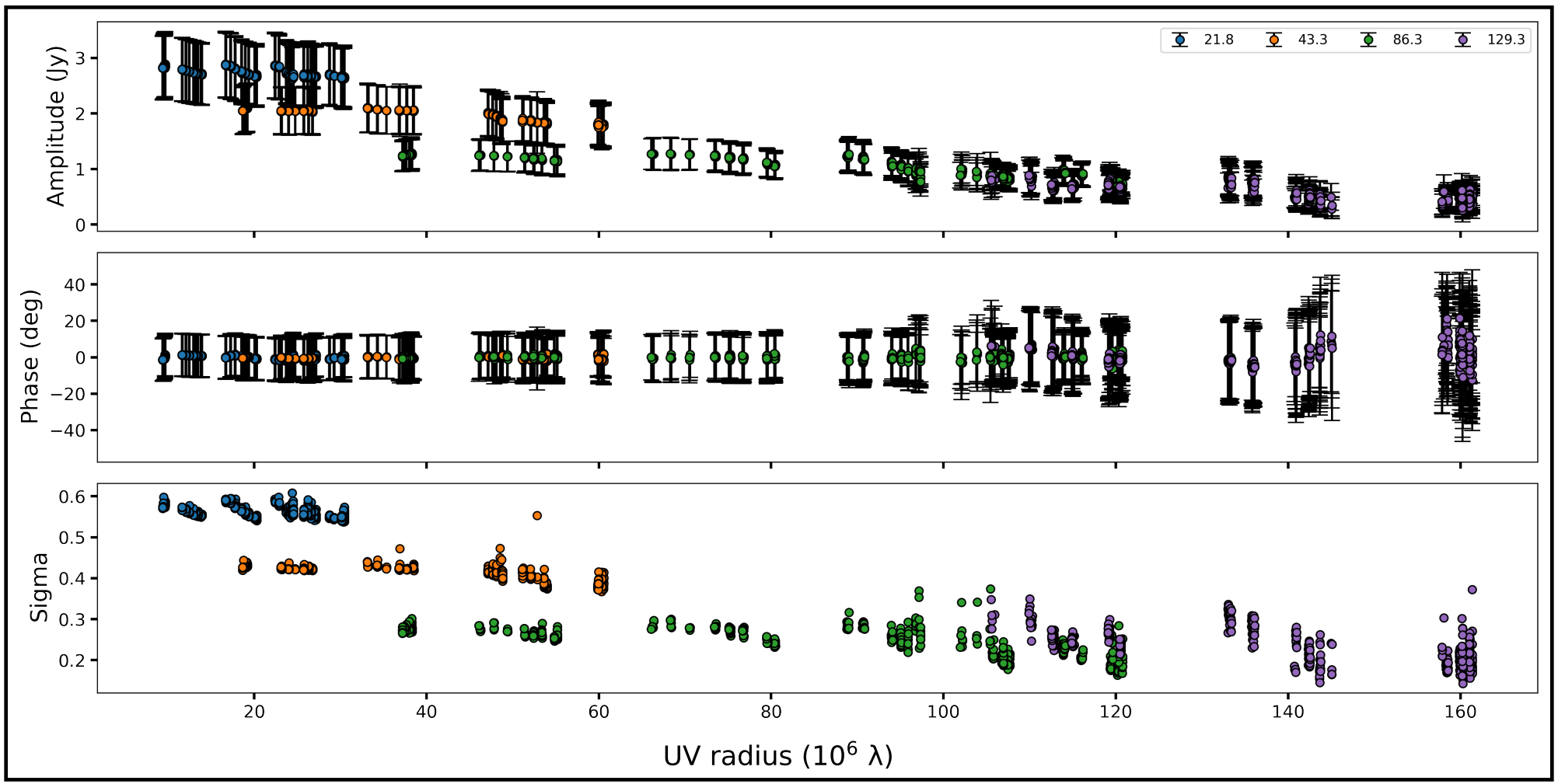
```
# apply systematics to data terms using median absolute deviation (MAD)
uvf.apply_systematics(binning=uvave, types=['vis', 'clamp', 'clphs'])

# flag uv-visibility having signal-to-noise ratio (SNR) lower than f
uvf.flag_uvvis(type='snr', value=f)
```

The followings are optional (example)

```
# flag uv-visibility data having sigma higher than f
uvf.flag_uvvis(type='sigma', value=f)

# flag uv-visibility data by the number of antenna
uvf.flag_uvvis(type='nant', value=i)

# flag uv-visibility data from a specific antenna(s) (ant1 & ant2)
uvf.flag_uvvis(type='ant', value=[‘ant1’, 'ant2'])

# flag uv-visibility data by uv-radius between [uvr1, uvr2] / mega-lambda
uvf.flag_uvvis(type='uvr', value=[uvr1, uvr2], unit='m')

# increase visibility sigma by f%, by adding a fractional value of visibility amplitude
uvf.add_error_fraction(fraction=0.0f)

# increase visibility sigma by a factor of f to the current value
uvf.add_error_factor(factor=f)
```

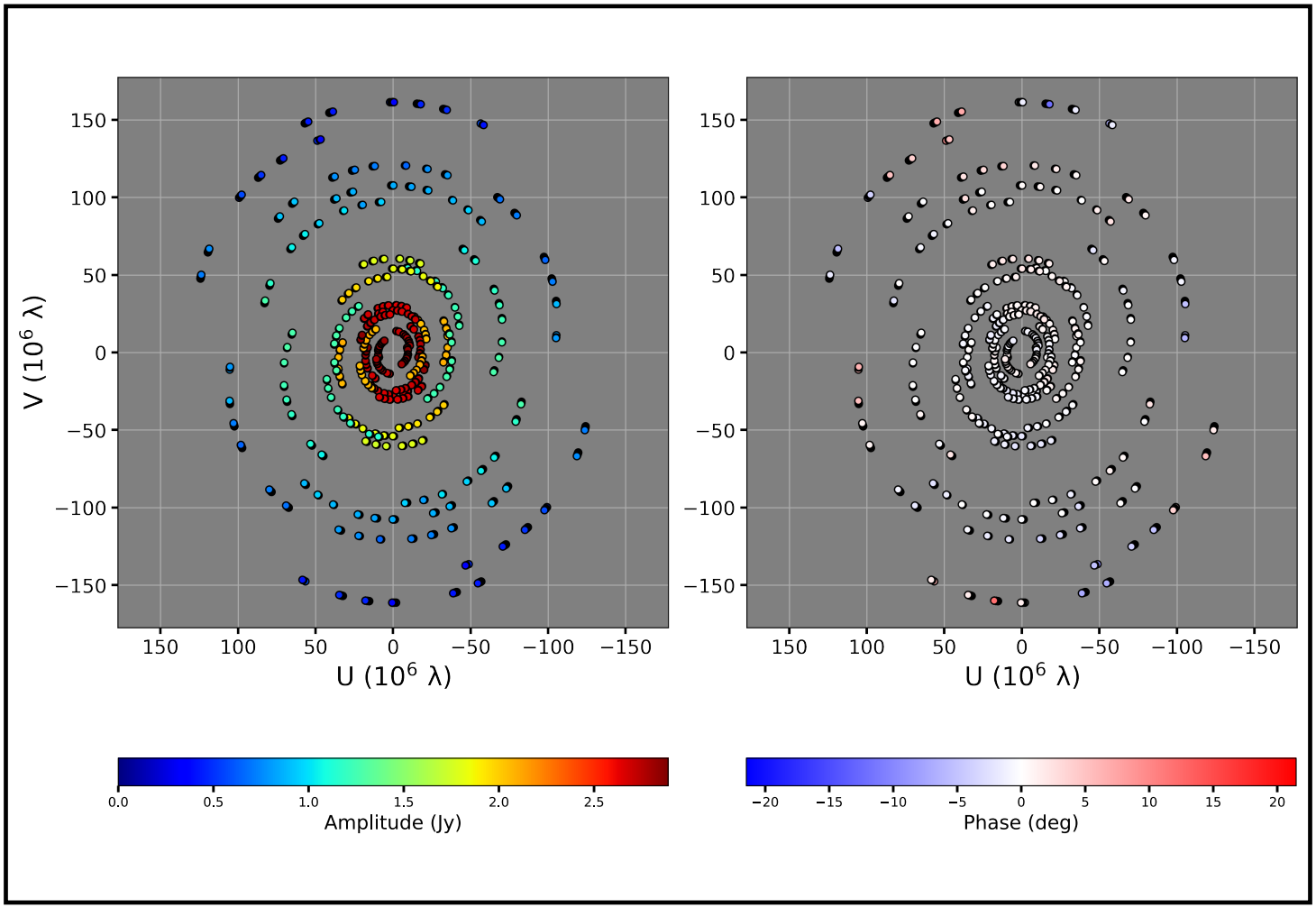
Example run // f24sl02b

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



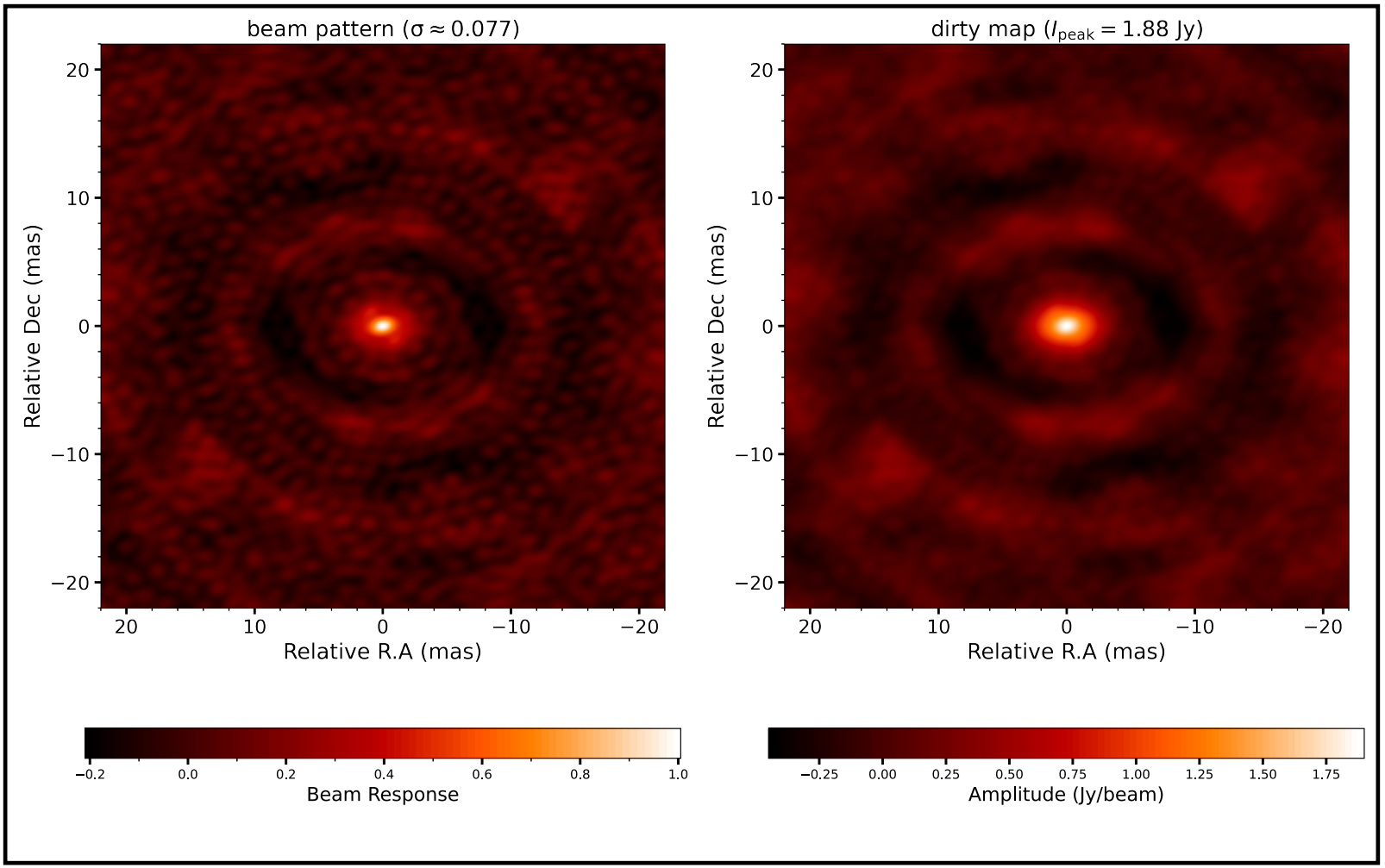
Example run // f24sl02b

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



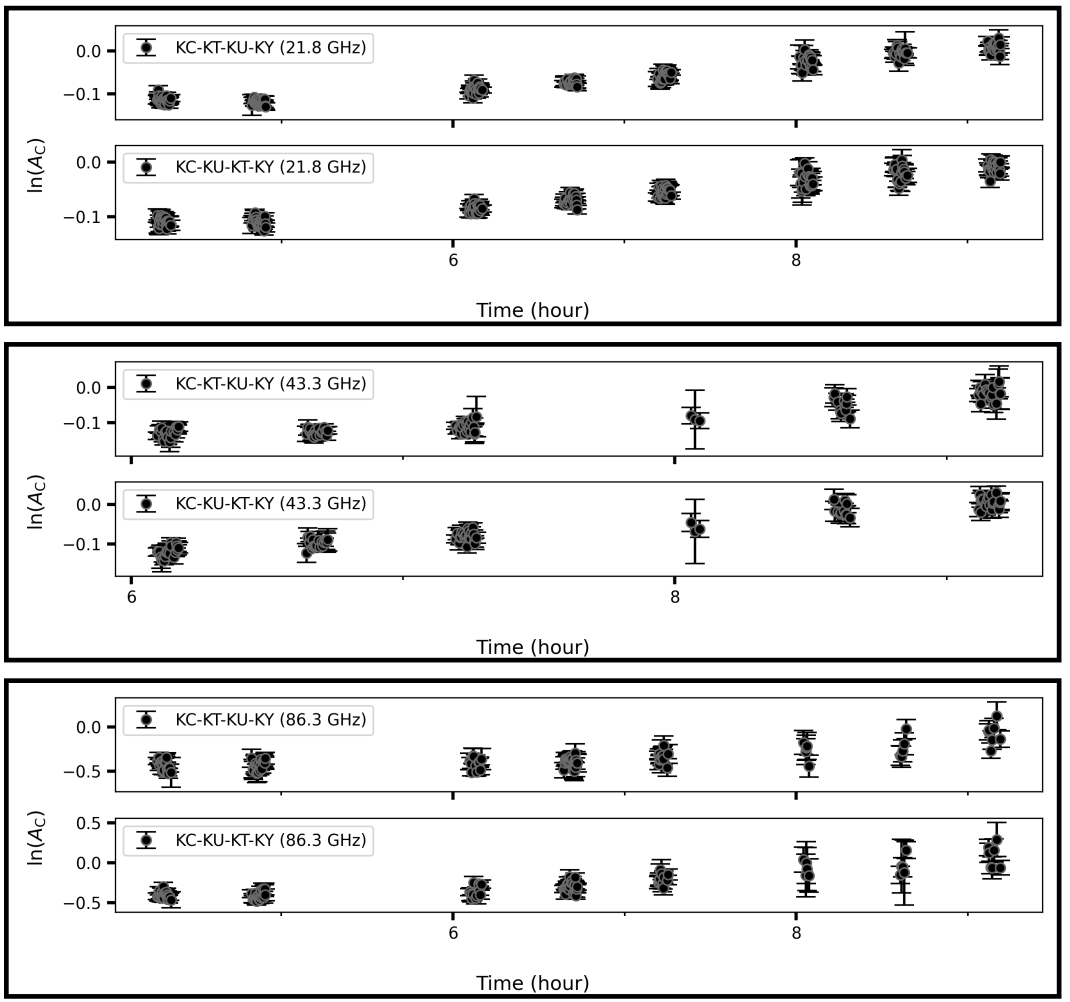
Example run // f24sl02b

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



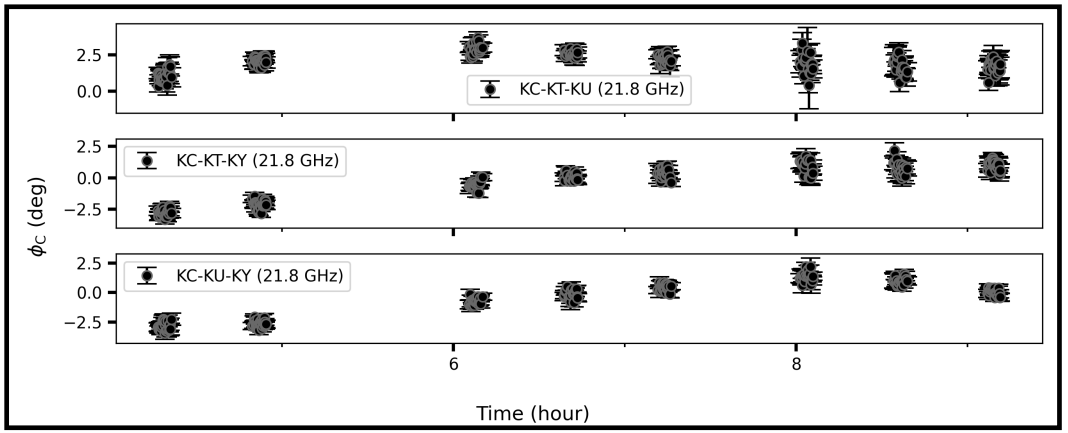
Example run // f24sl02b

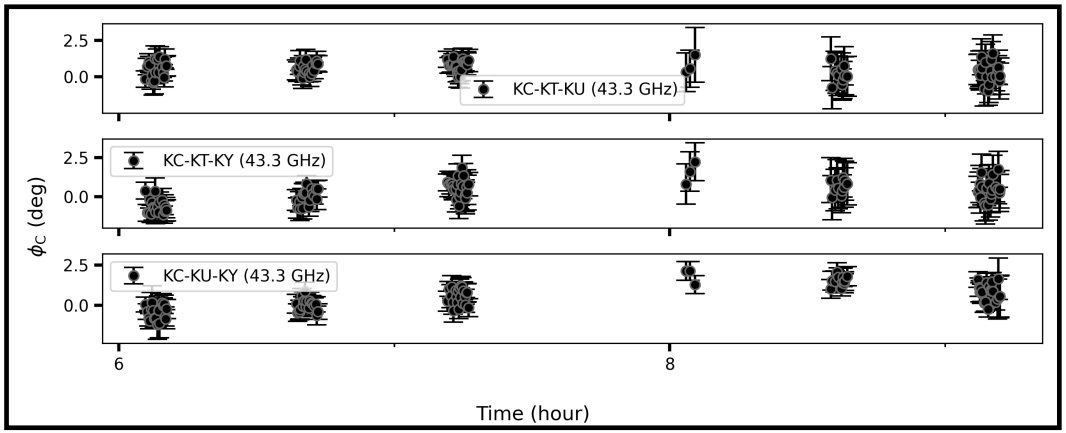
```
# check visibility data
uvall.ploter.draw_closure(type='clamp', plotimg=True) # save_path=, save_name=)
```

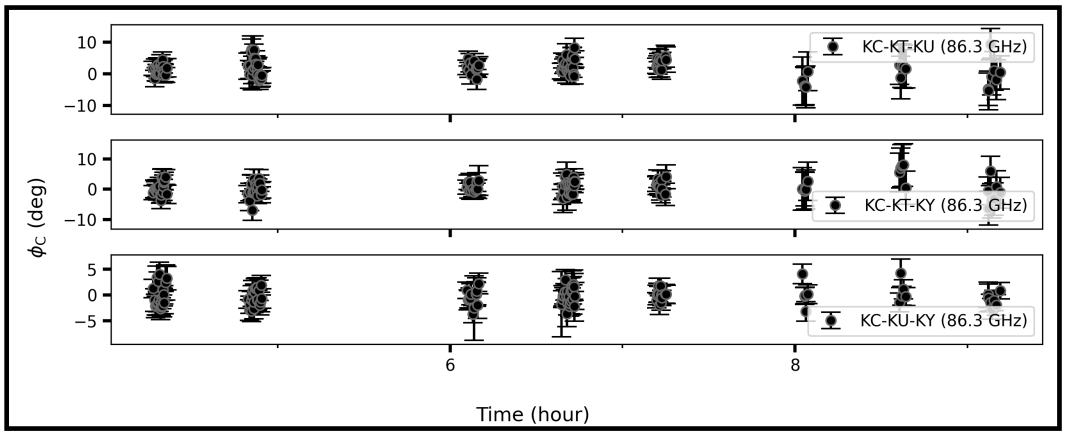


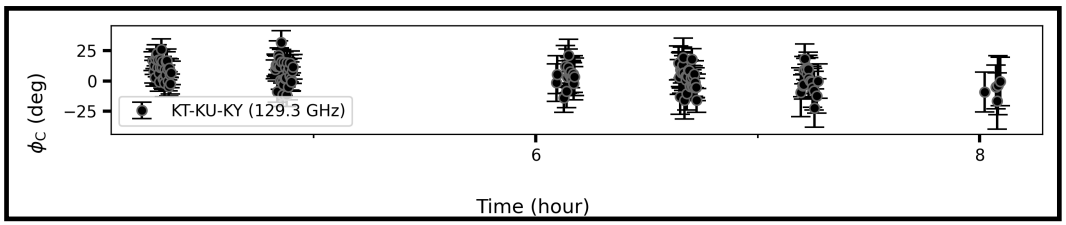
Example run // f24sl02b

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









Example run // f24sl02b

** Please see example_run.py for detailed attribute setting*

```
# run model-fitting
mfu = gv.modeling.modeling(
    uvfs=uvfs, select='ll', uvw='u', sampler='slice', spectrum='ssa',
    ftype=ftype, fwght=fwght, doampcal=True, dophscal=True, ncpu=ncpu,
    …
)
mfu.run()
```

```
### Running parameters ###
# Map range: 22.0 (mas)
# B_min: 1.298 (mas)
# B_maj: 1.951 (mas)
# B_pa: 100.888 (deg)
# Fit-spec: 'ssa'
# Fit-type: ['amp', 'clamp', 'clphs']
# Fit-wght: [0.1, 1, 1]
# G.model: 'gaussian'
# Number of complex visibility: 2628
# Number of closure amplitude: 750
# Number of closure phase: 1251
# uv-average time: 10 (sec)
# Selected polarization: 'LL'
# Number of active CPU cores: 8/8


# Maximum baseline flux : 2.889 Jy

# Running... (Pol MF.LL, MaxN_model=10, sampler='slice') // ! relative position
# Fit-parameters : {'amp': 0.1, 'clamp': 1, 'clphs': 1}
45866it [1:39:01,  6.15it/s, batch: 0 | bound: 554 | nc: 1133 | ncall: 50726094 | eff(%):  0.090 | loglstar:   -inf < 10340.190 <    inf | logz: 10247.186 +/-  0.428 | dlogz: 23.364 >  0.010]
```

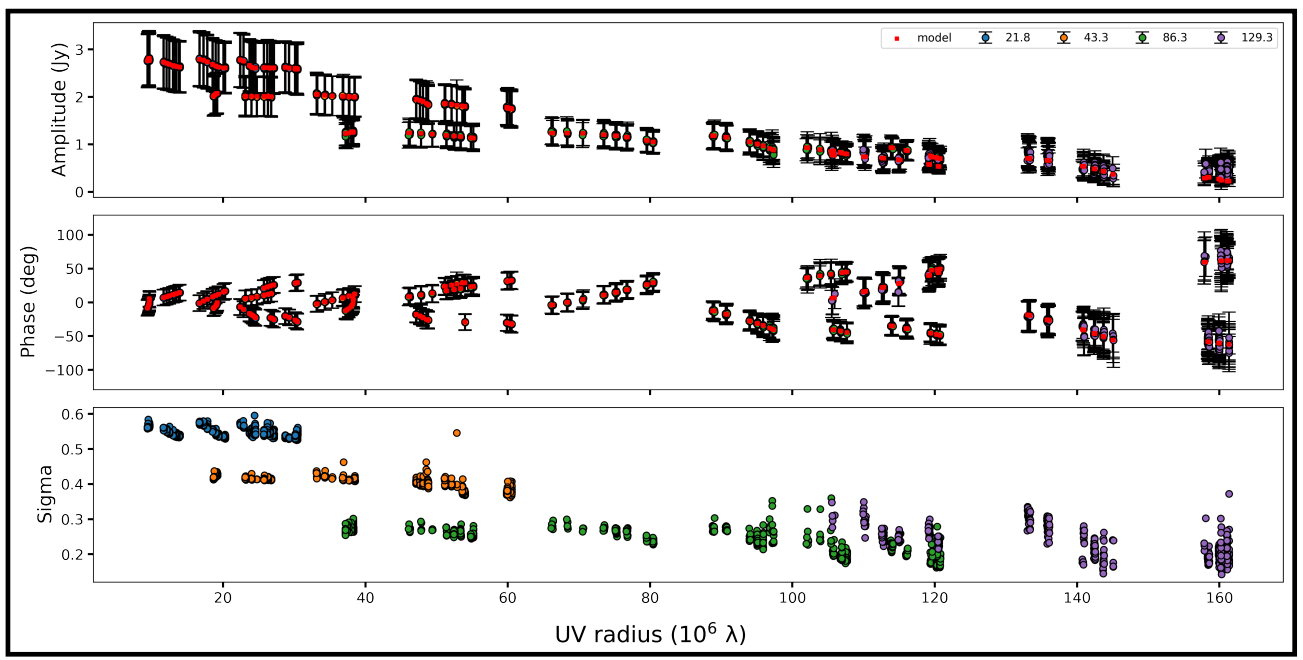
until remaining evidence
meets this criteria

Example run // f24sl02b

** Please see example_run.py for detailed attribute setting*

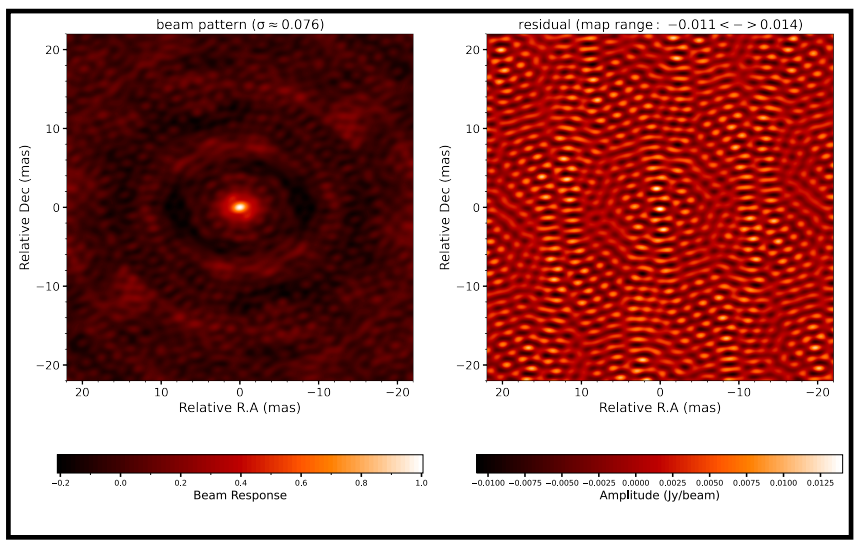
```
# run model-fitting
mfu = gv.modeling.modeling(
    uvfs=uvfs, select='ll', uvw='u', sampler='slice', spectrum='ssa',
    ftype=ftype, fwght=fwght, doampcal=True, dophscal=True, ncpu=ncpu,
    …
)
mfu.run()
```

radplot



beam parttern

residual map



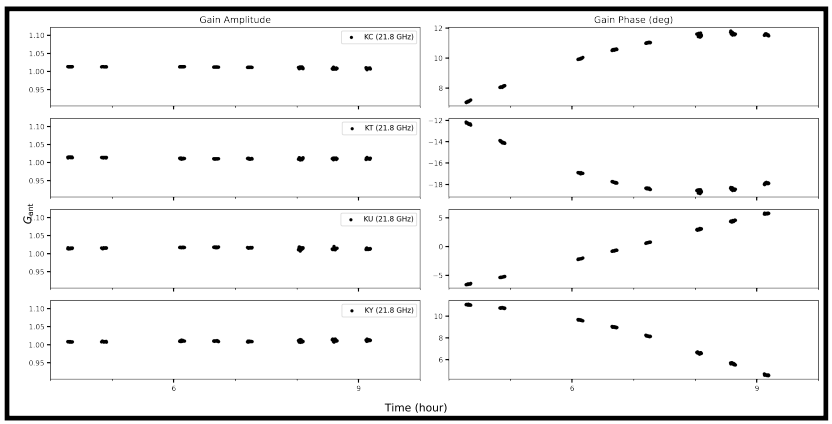
Example run // f24sl02b

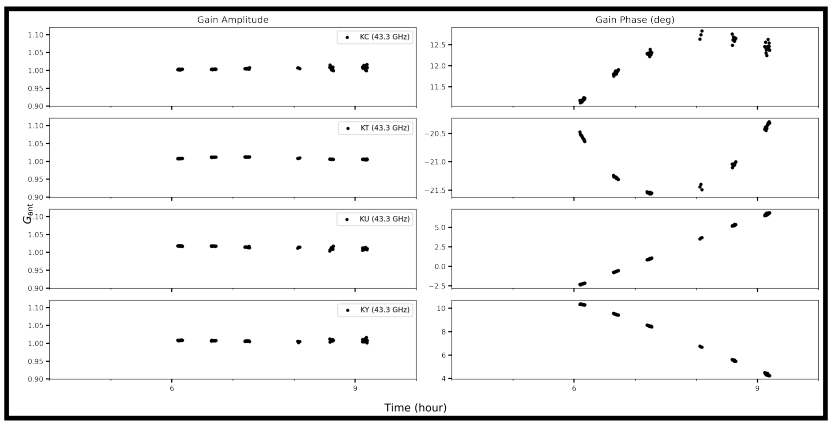
** Please see example_run.py for detailed attribute setting*

```
# run model-fitting
mfu = gv.modeling.modeling(
    uvfs=uvfs, select='ll', uvw='u', sampler='slice', spectrum='ssa',
    ftype=ftype, fwght=fwght, doampcal=True, dophscal=True, ncpu=ncpu,
    …
)
mfu.run()
```

complex gain
after self-calibration

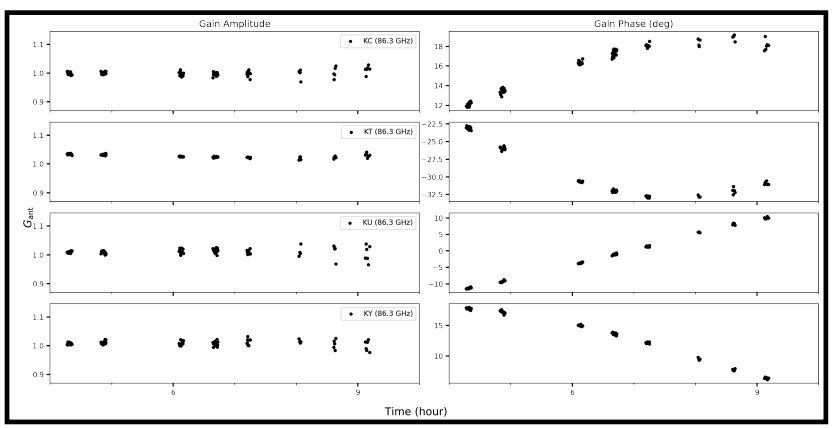
22 GHz

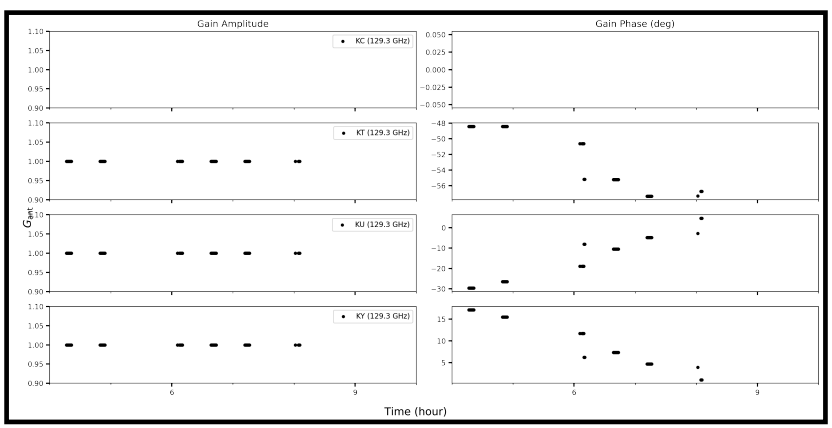




43 GHz

86 GHz





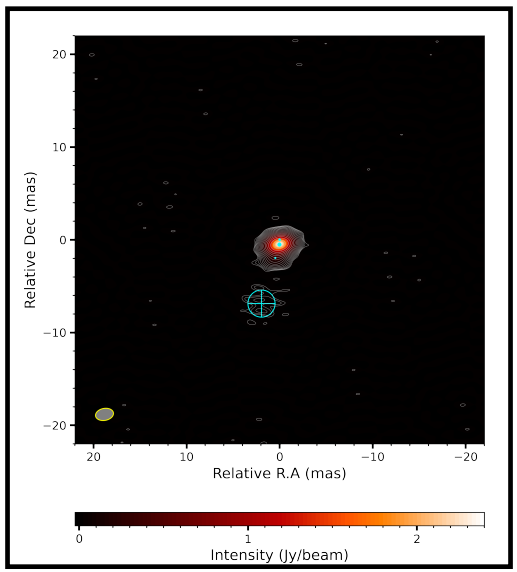
129 GHz

Example run // f24sl02b

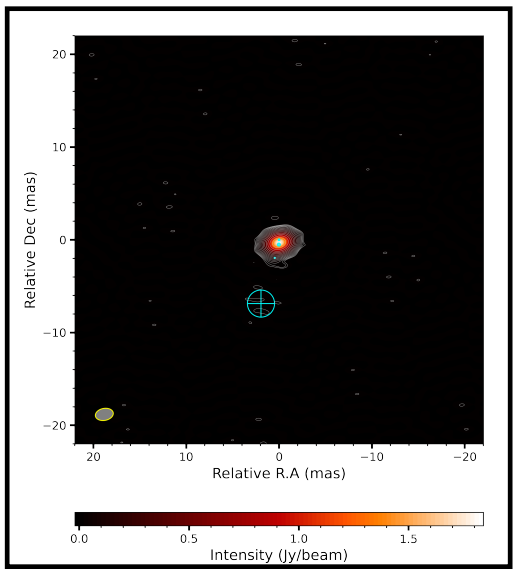
** Please see example_run.py for detailed attribute setting*

```
# run model-fitting
mfu = gv.modeling.modeling(
    uvfs=uvfs, select='ll', uvw='u', sampler='slice', spectrum='ssa',
    ftype=ftype, fwght=fwght, doampcal=True, dophscal=True, ncpu=ncpu,
    …
)
mfu.run()
```

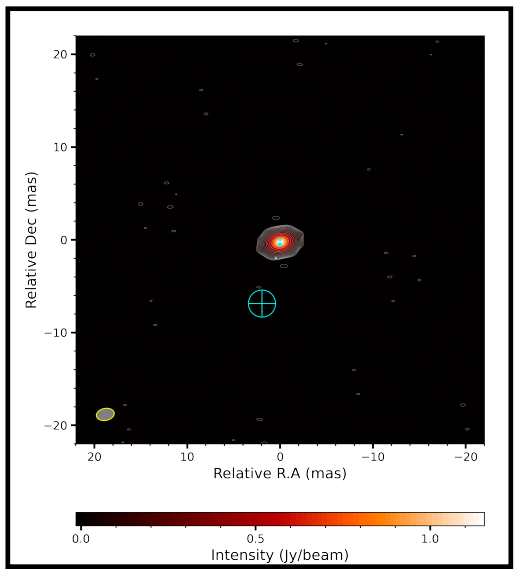
22 GHz



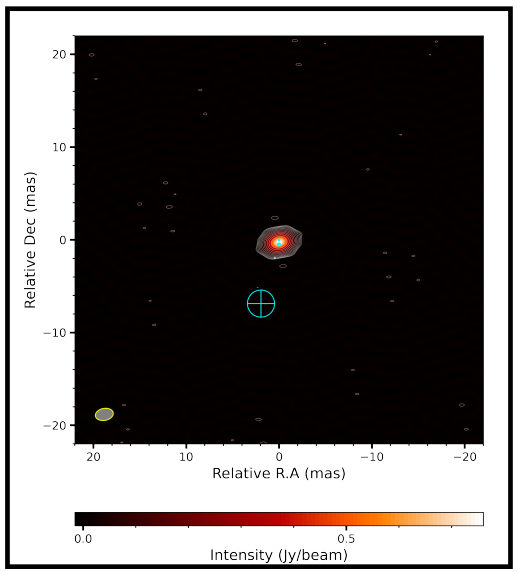
43 GHz



86 GHz



129 GHz

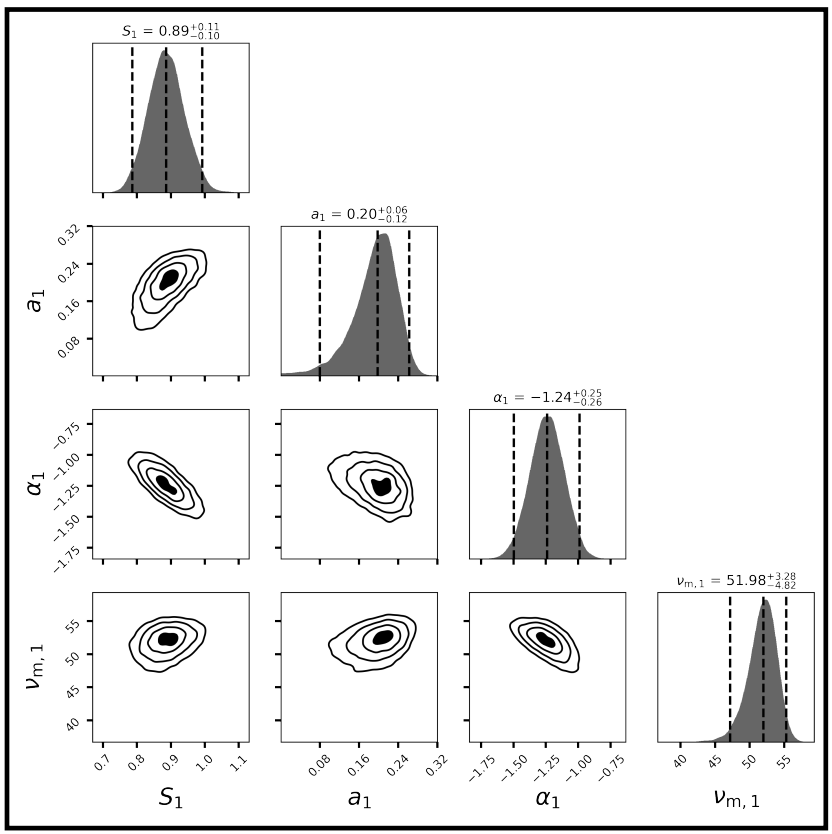


Example run // f24sl02b

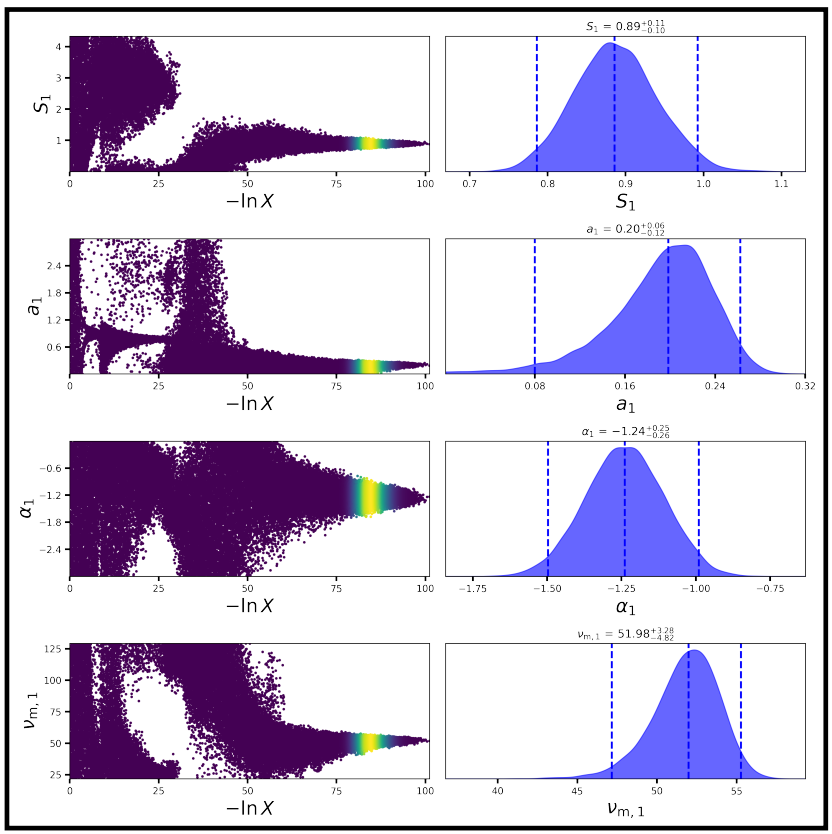
** Please see example_run.py for detailed attribute setting*

```
# run model-fitting
mfu = gv.modeling.modeling(
    uvfs=uvfs, select='ll', uvw='u', sampler='slice', spectrum='ssa',
    ftype=ftype, fwght=fwght, doampcal=True, dophscal=True, ncpu=ncpu,
    …
)
mfu.run()
```

corner plot of model #1

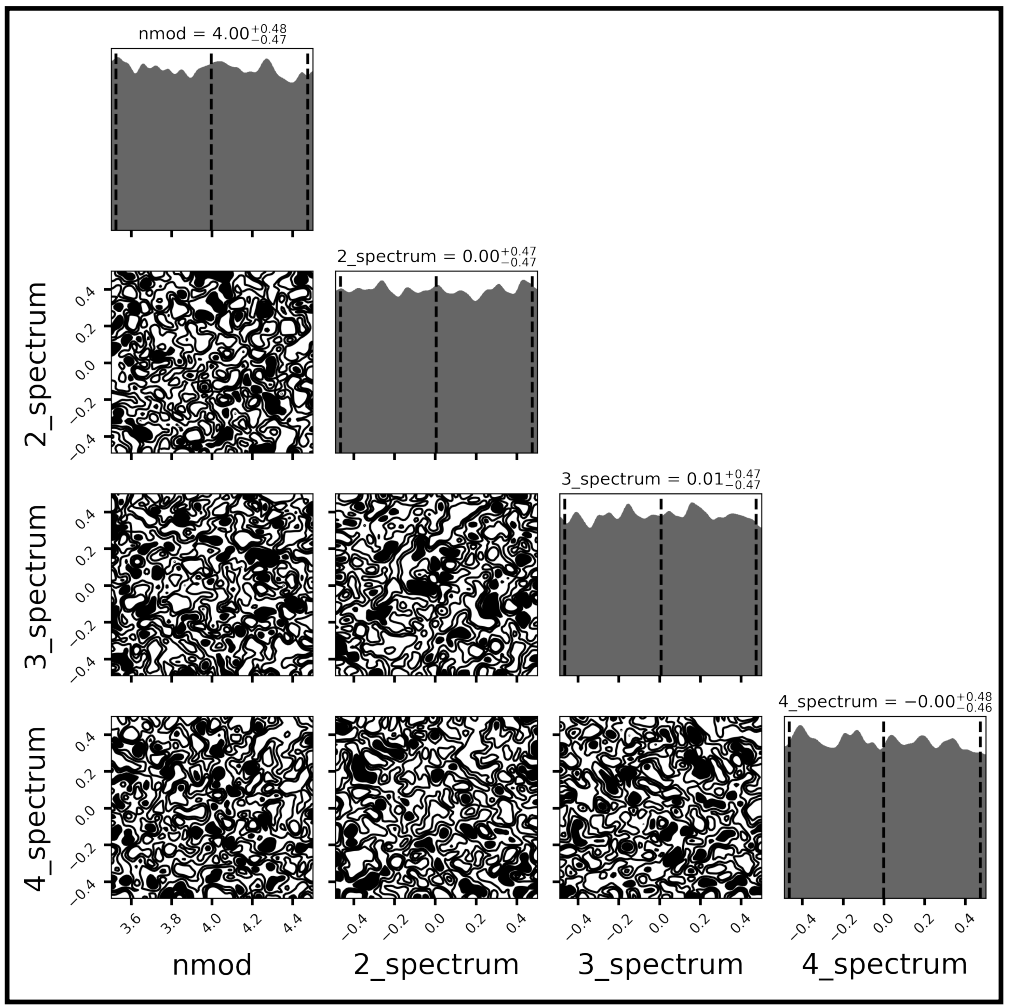


trace plot of model #2

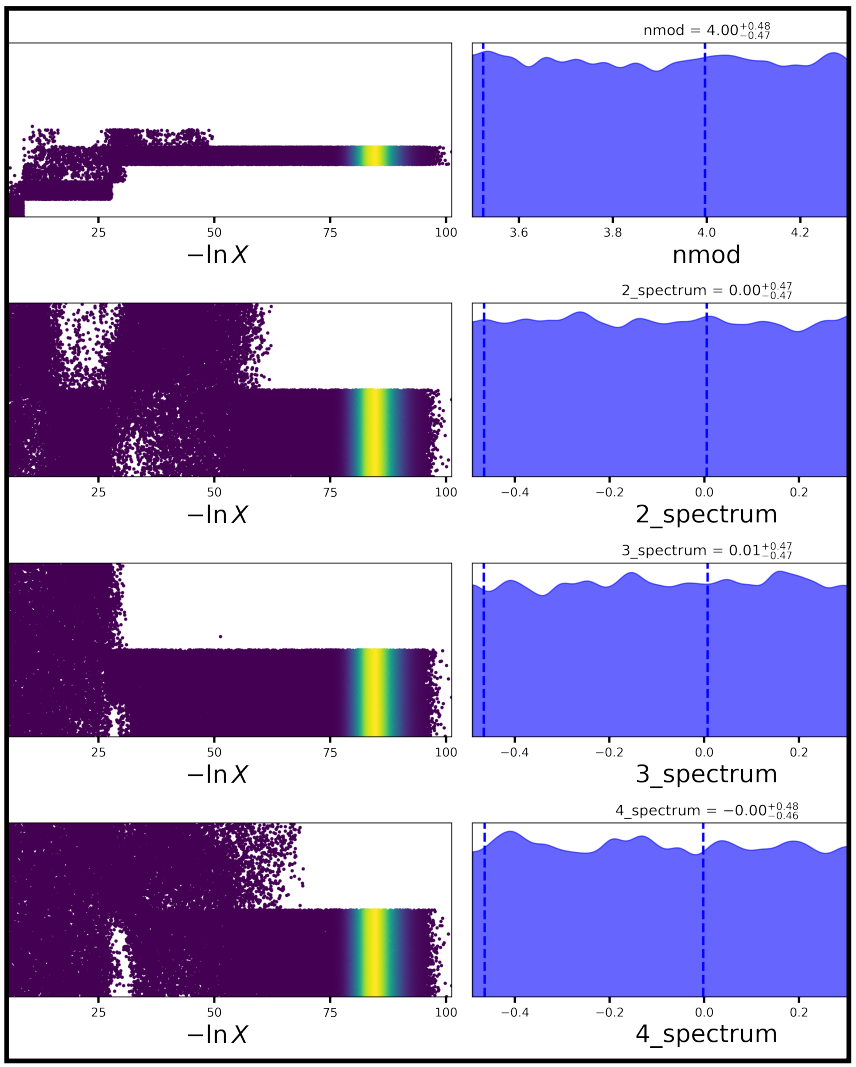


Example run // f24sl02b

corner plot of boolean-spectrum values

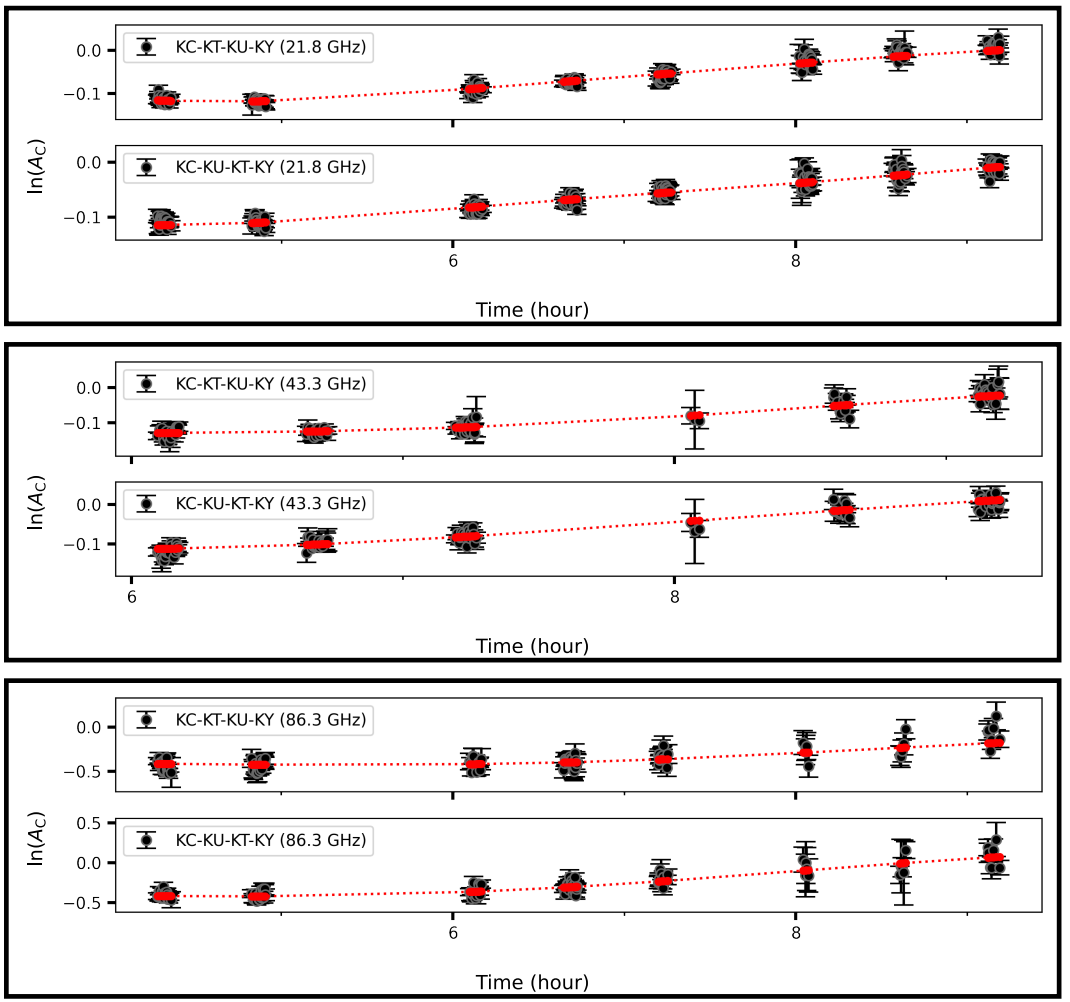


trace plot of boolean-spectrum values



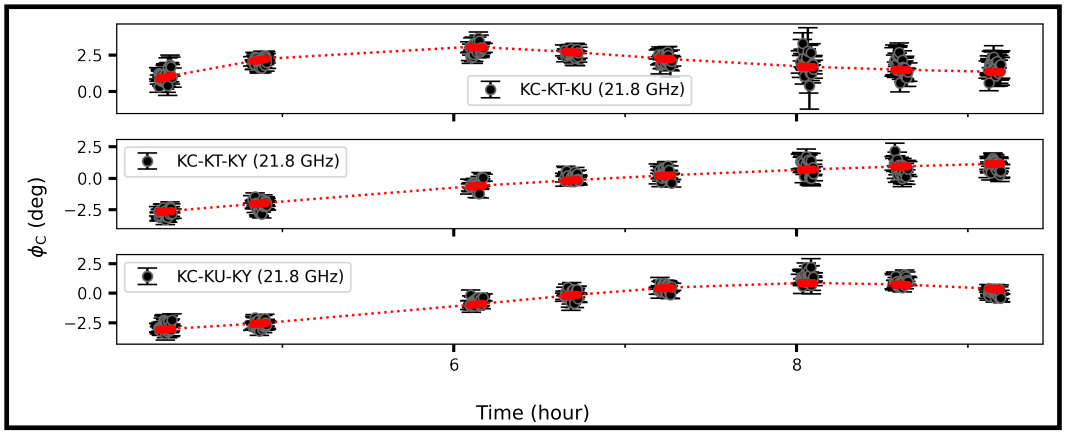
Example run // f24sl02b

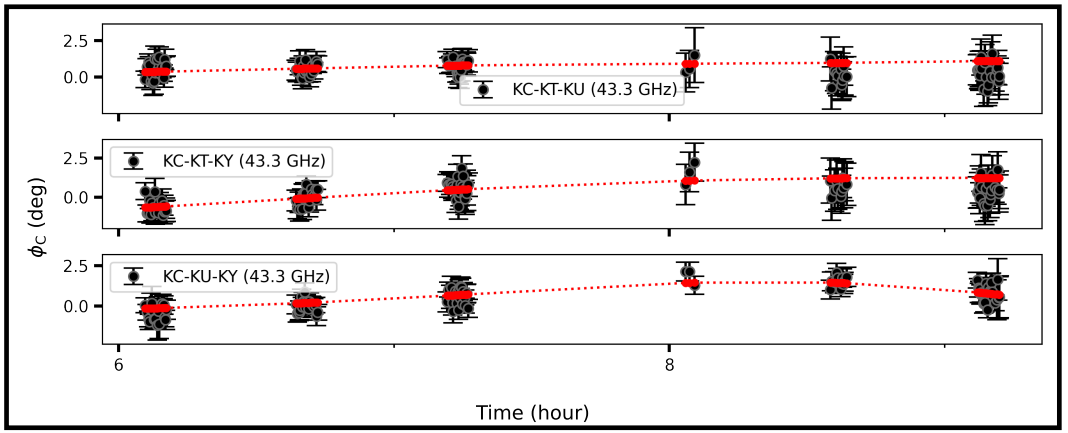
logarithmic closure amplitude

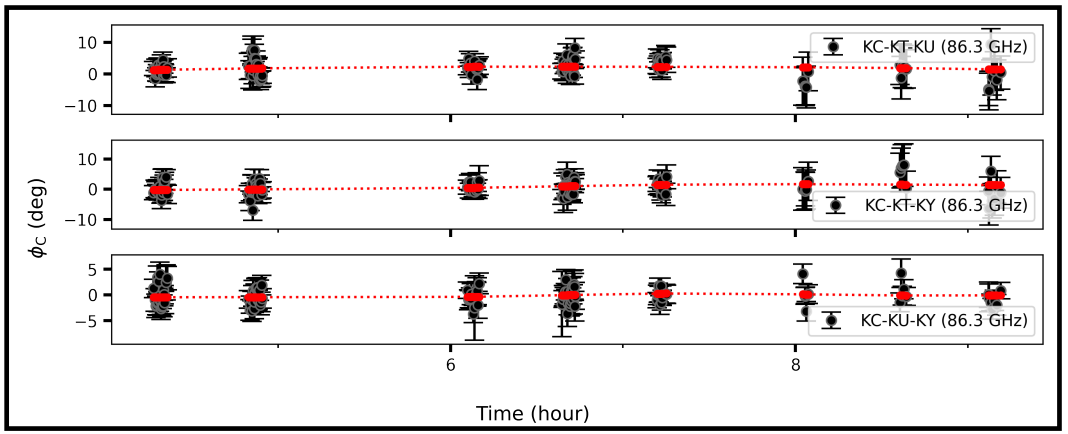


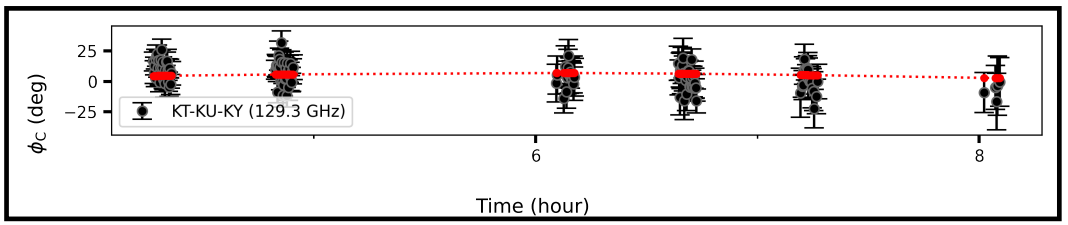
Example run // f24sl02b

closure phase









Statistics // n25hj01a

Obs. date: 2025 02-10 06:00 – 02-11 06:00 (24h)

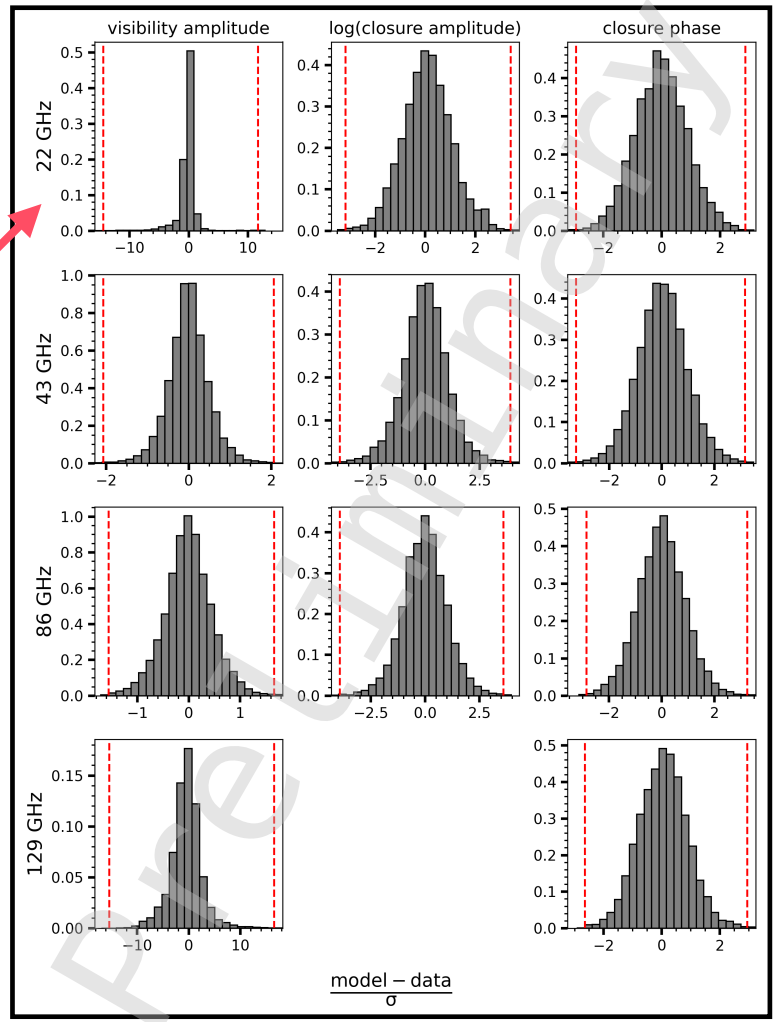
Bands: 22 / 43/ 86 / 129 GHz

Sources: 21 AGNs

0235+164	1226+023 (3C 273)
0316+413 (3C 84)	1228+126 (M 87)
0415+379 (3C 111)	1253−055 (3C 279)
0420−014	1510−089
0430+052 (3C 120)	1553+113
0446+112	1641+399 (3C 345)
0735+178	1652+398 (Mrk 501)
0851+202 (OJ 287)	1749+096
0954+658	1807+698 (3C 371)
1156+295 (Ton 599)	2200+420 (BL Lac)
	2251+158 (3C 454.3)

Broad distribution in vis. amplitude caused by systematics during the observation

* red vertical lines: 99% level



Normalized residual distribution obtained from all modeling results, except for 3C 84

Compare with uv-radius-limited VLBA // n25hj01a

Obs. date:
  EKVN: 2025-02-10 (22 – 129 GHz)
  VLBA: 2025-02-16 (43 GHz)