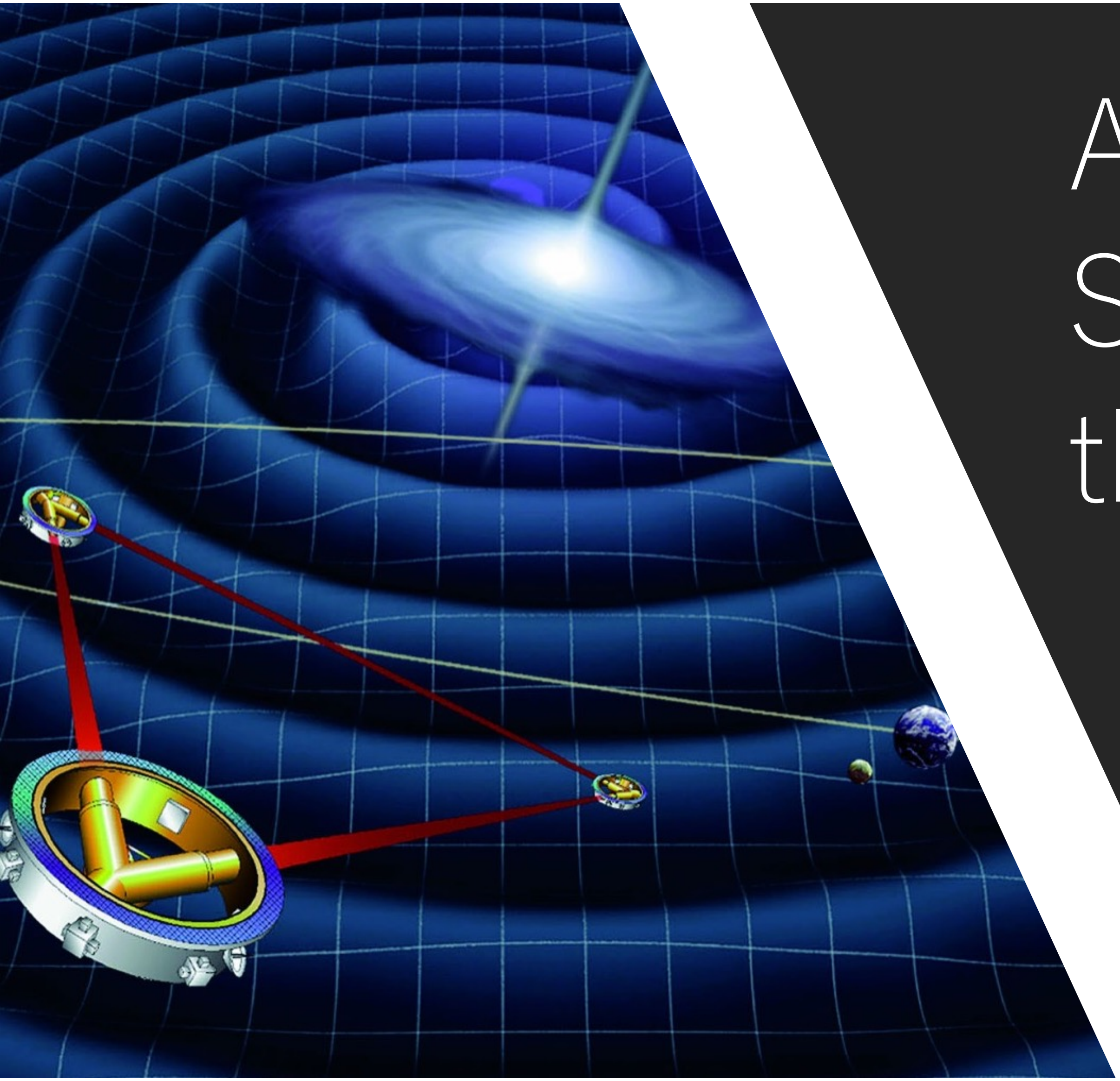


Accelerating SGWBinner code with the JAX framework

Jun'ya Kume (UNIPD, INFN, RESCEU)

In collaboration with

Marco Peloso (UNIPD, INFN),

Mauro Pieroni (CERN),

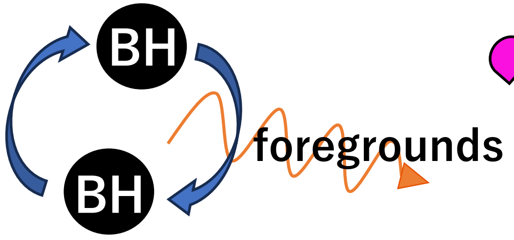
Angelo Ricciardone (U. Pisa, INFN)

Contents

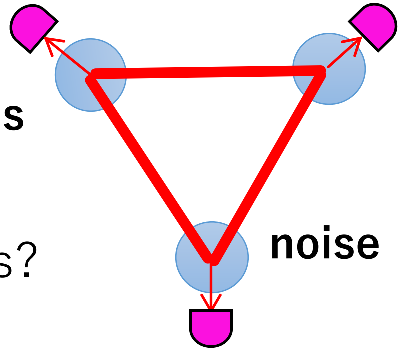
- Signal reconstruction with SGWBinner
- How can we use JAX?
- New results with the accelerated code
- Summary & Discussion

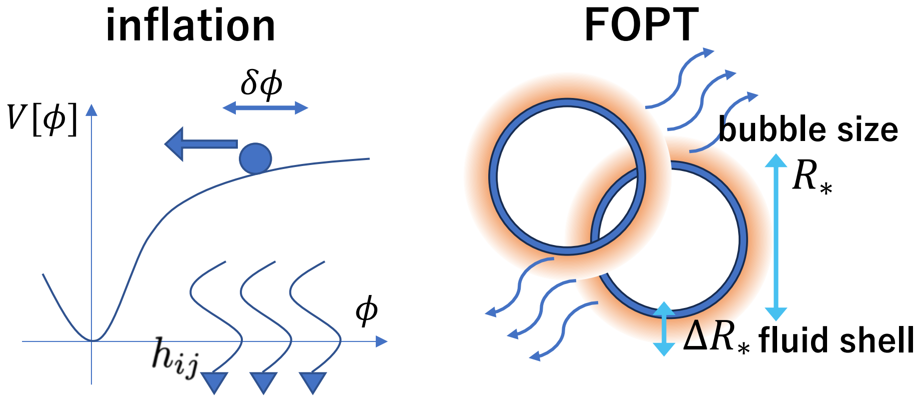
Signal reconstruction with SGWBinner

- A handy tool to test your model! (Caprini+ 2019, Flauger+ 2021)



-Can LISA detect signals?
constrain models?

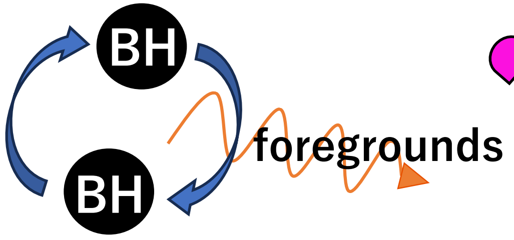


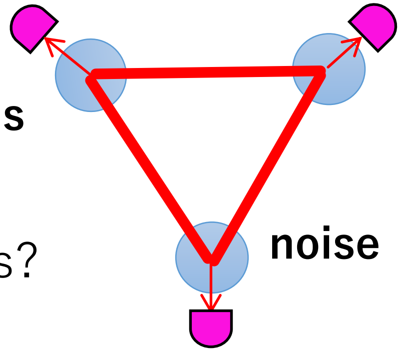


expected SGWB sources

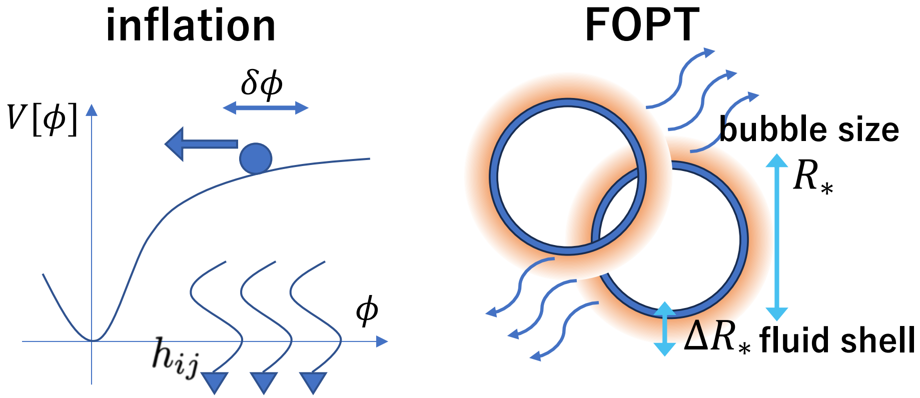
Signal reconstruction with SGWBinner

- A handy tool to test your model! (Caprini+ 2019, Flauger+ 2021)





-Can LISA detect signals?
constrain models?



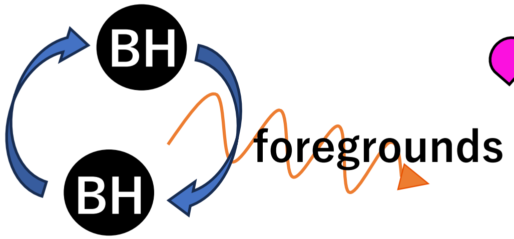
expected SGWB sources

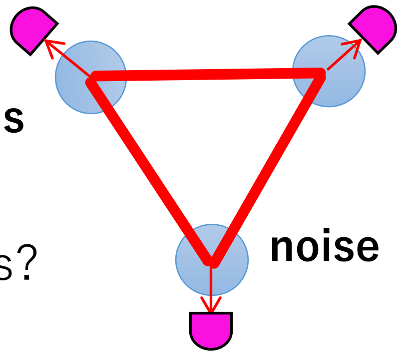
- simulate LISA TDI data stream with signal & foregrounds
- semi-analytic forecast on "binned" signal reconstruction
- signal reconstruction with MC sampling (binned/template)

On your laptop!

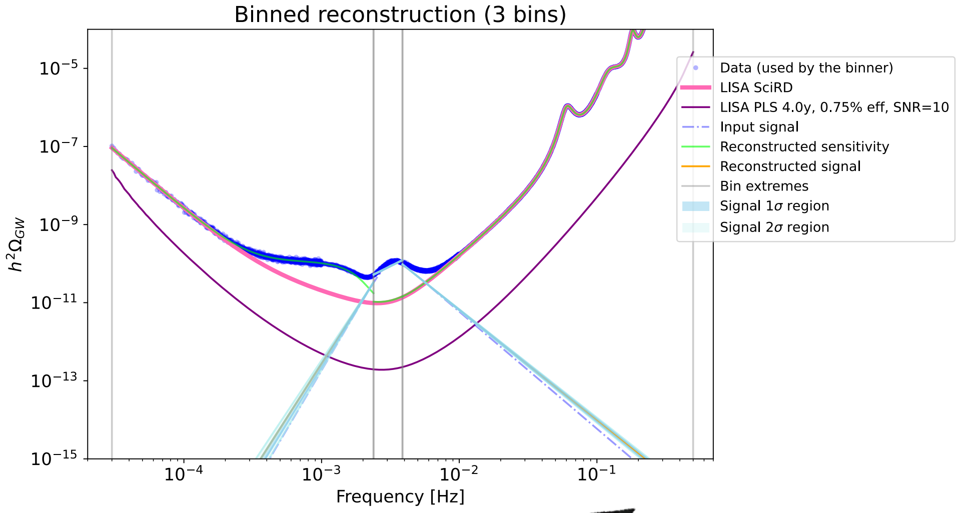
Signal reconstruction with SGWBinner

- A handy tool to test your model! (Caprini+ 2019, Flauger+ 2021)





-Can LISA detect signals?
constrain models?



- simulate LISA TDI data stream with signal & foregrounds
- semi-analytic forecast on "binned" signal reconstruction
- signal reconstruction with MC sampling (binned/template)

On your laptop!

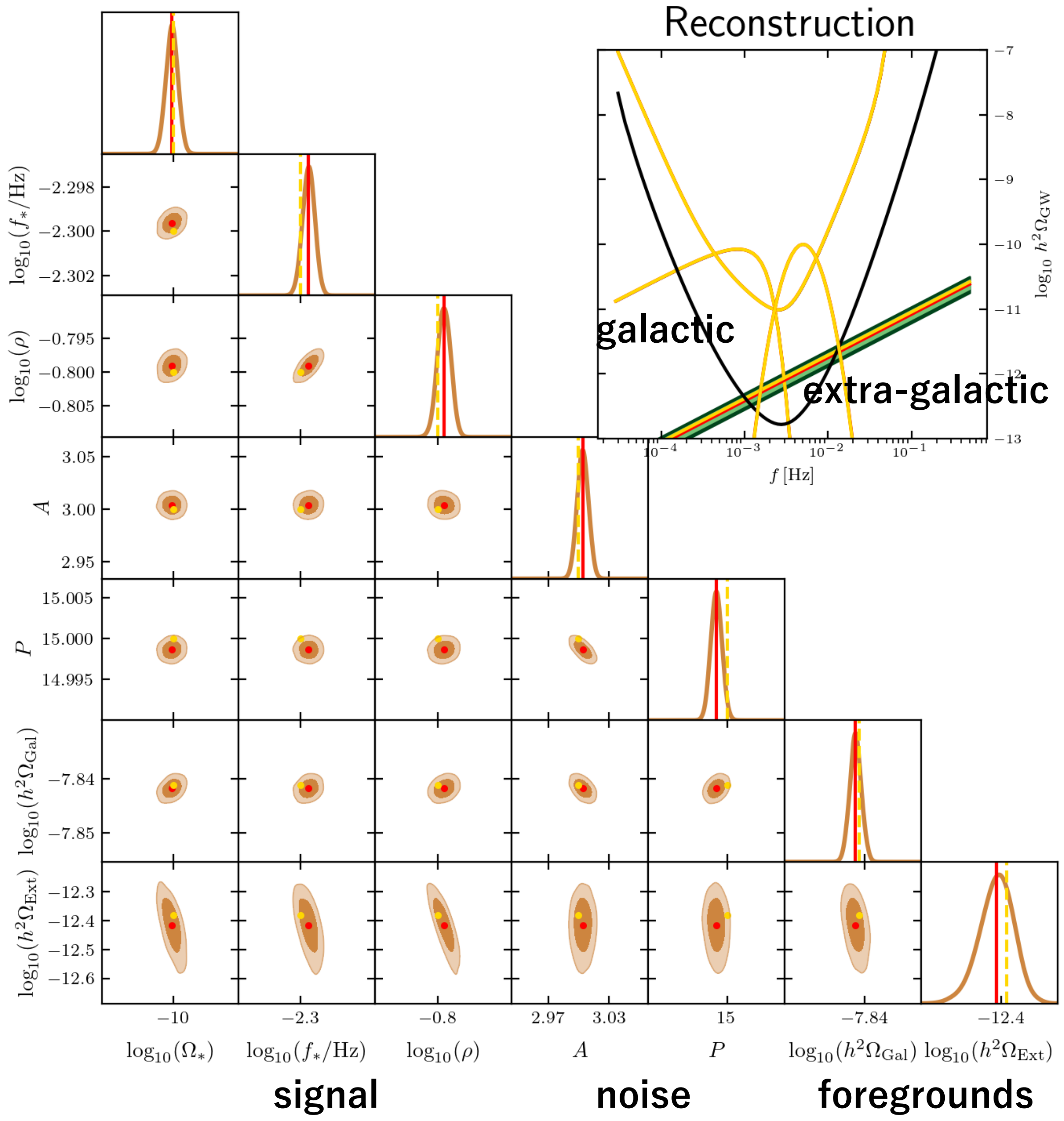
- Signal reconstruction by MC sampling

Interfacing **Cobaya** (Torrado & Lewis)

for Bayesian analysis in cosmology

total posterior for all bins and all channel

→ more accurate prediction! But...



- Signal reconstruction by MC sampling

Interfacing **Cobaya** (Torrado & Lewis)

for Bayesian analysis in cosmology

total posterior for all bins and all channel

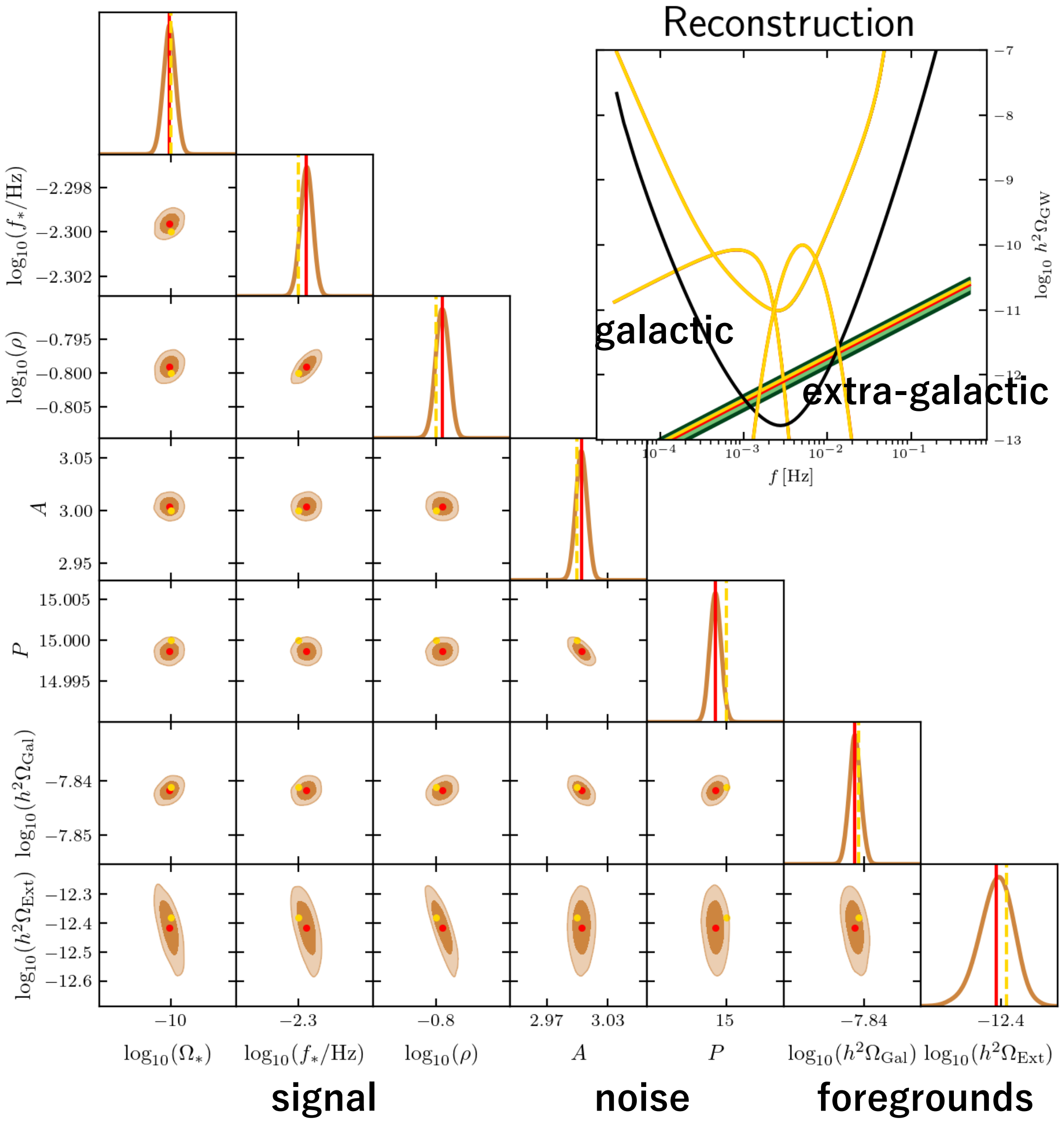
→ more accurate prediction! But...

The most time-consuming part of Binner

running

over night...

- Can we further accelerate this code? ⌛



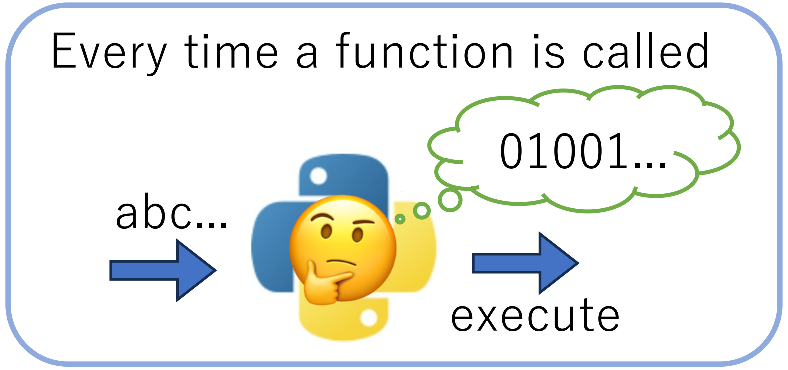
Contents

How can we use JAX?

- What's JAX? Why JAX?

(Bradbury+ 2018)

"high-performance computing"
& "large-scale ML"
→ linear algebra with huge arrays



flexible but slower...

Appreciable features:

- **Just-In-Time compile** provided by XLA compiler

  code optimization targeted on CPU, GPU & TPU

- `jax.numpy` & `jax.scipy` libraries for XLA

  easy conversion of the existing code!

How can we use JAX?

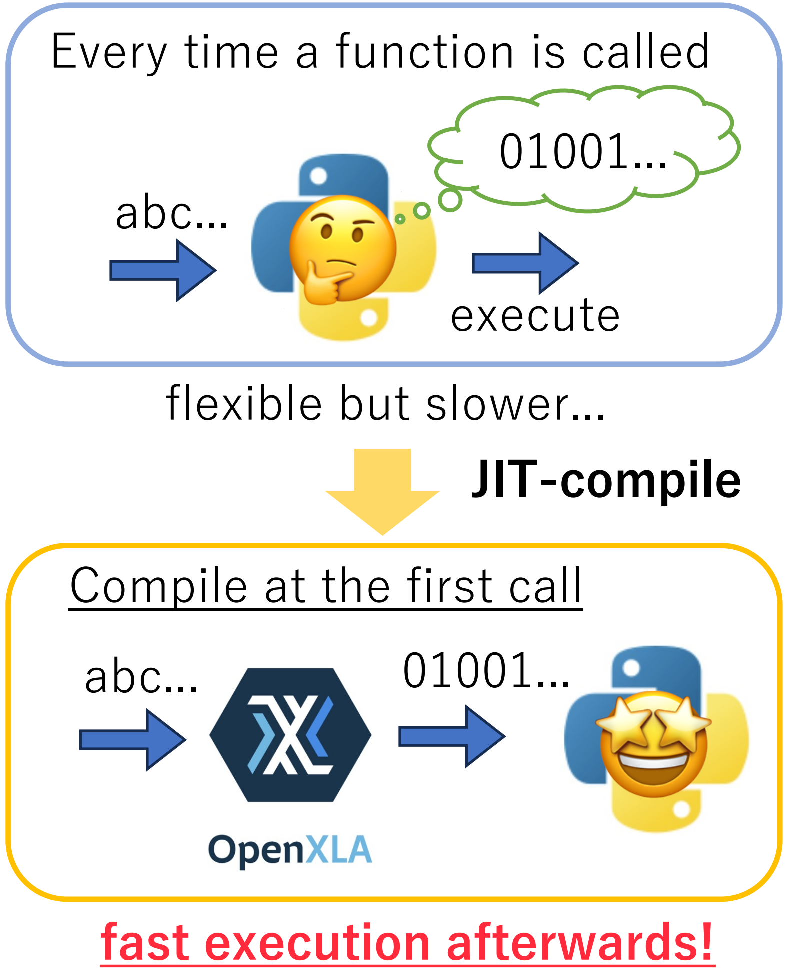
- What's JAX? Why JAX?

(Bradbury+ 2018)

"high-performance computing"
& "large-scale ML"
→ linear algebra with huge arrays

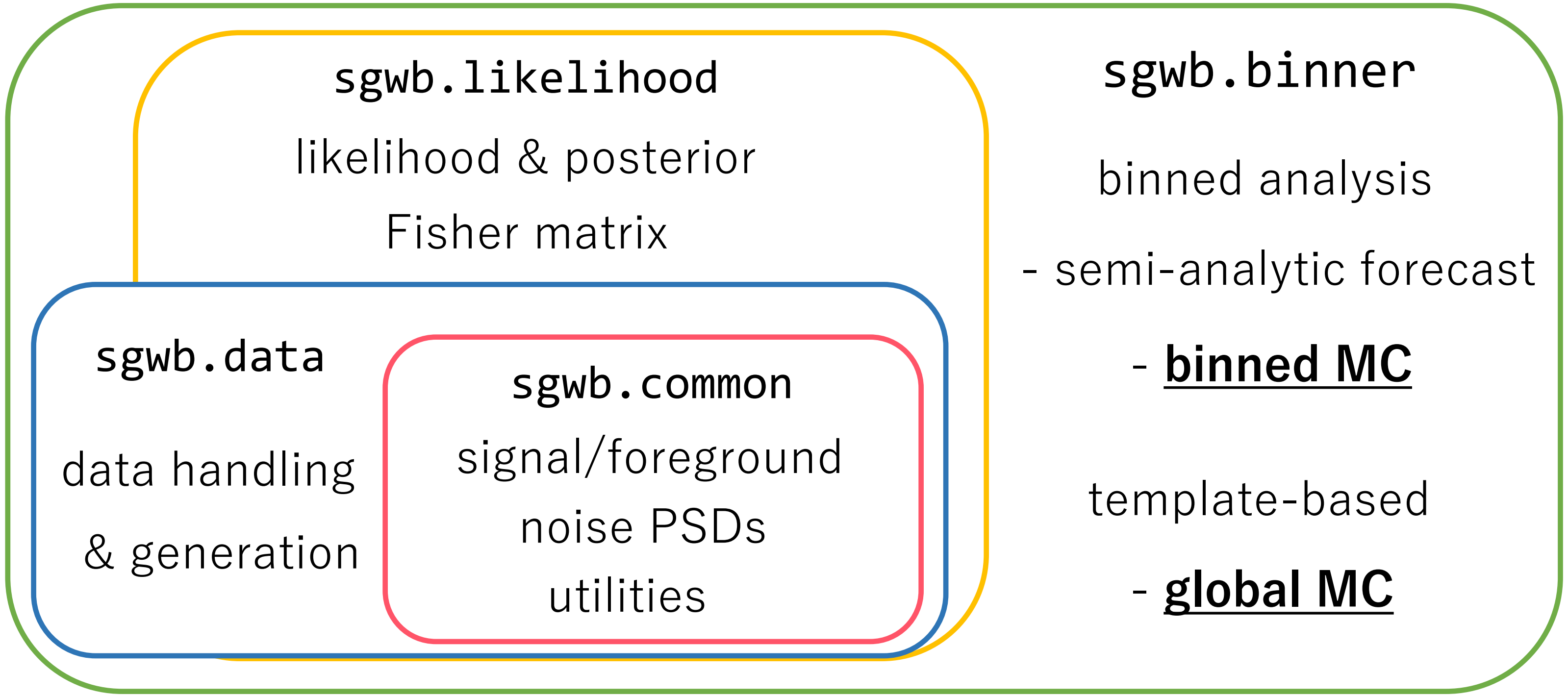
Appreciable features:

- **Just-In-Time compile** provided by XLA compiler
  code optimization targeted on CPU, GPU & TPU
- `jax.numpy` & `jax.scipy` libraries for XLA
  easy conversion of the existing code!



- Embedding JAX into SGWBinner

Schematics of SGWBinner code



working with coarse-grained data $\bar{D}_{ij}(f_k)$ (ij → TDI ch.)

to compute likelihood $\mathcal{L}(\bar{D}_{ij}(f_k)|\vec{\theta},\vec{n})$, posterior & Fisher matrix...

- What to do?

Make use of **`jax.jit`**

at <u>likelihood computation</u>

$$\mathcal{L}(\bar{D}_{ij}(f_k)|\vec{\theta},\vec{n})$$

➡ **<u>accelerate MC!</u>**

※**NumPy** fast enough for

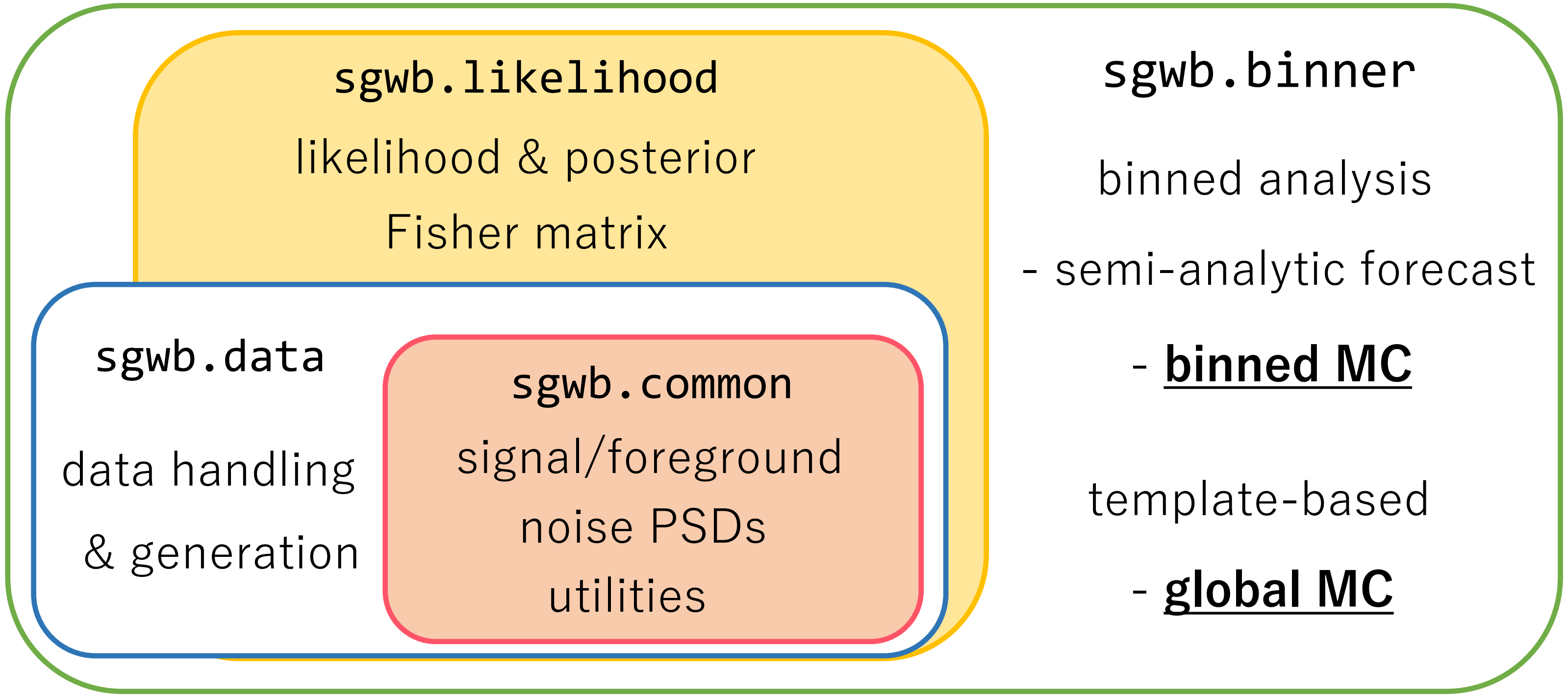
semi-analytic forecast

→ keep the other parts

NumPy-based

- Embedding JAX into SGWBinner

Schematics of SGWBinner code



working with coarse-grained data $\overline{D}_{ij}(f_k)$ (ij → TDI ch.)

to compute likelihood $\mathcal{L}(\overline{D}_{ij}(f_k)|\vec{\theta},\vec{n})$, posterior & Fisher matrix...

- What to do?

Make use of `jax.jit`

at <u>likelihood computation</u>

$$\mathcal{L}(\overline{D}_{ij}(f_k)|\vec{\theta},\vec{n})$$

➡ **<u>accelerate MC!</u>**

※**NumPy** fast enough for

semi-analytic forecast

→ keep the other parts

NumPy-based

- Accelerating the LISA likelihood

sgwb.likelihood

$\mathcal{L}(\bar{D}_{ij}(f_k)|\vec{\theta},\vec{n})$

`likelihood_jax/statistical_tools_jax`

: a new class to JIT-compile this function

noise

`LISA_noise_jax`

signal & foregrounds

`templates`

sgwb.common/jax

→ power spectrum computed with `jax.numpy(scipy)`

mostly `numpy` → `jax.numpy`

`scipy` → `jax.scipy`

with a care on traceability

(see JAX documentation)

This JAXed class is called

at final MC/global MC

in `sgwb.binner`

- Accelerating the LISA likelihood

$\mathcal{L}(\bar{D}_{ij}(f_k)|\vec{\theta},\vec{n})$

sgwb.likelihood

`likelihood_jax/statistical_tools_jax`

: a new class to JIT-compile this function

noise → `LISA_noise_jax`

signal & foregrounds → `templates`

sgwb.common/jax

→ power spectrum computed with `jax.numpy(scipy)`

mostly `numpy` → `jax.numpy`

`scipy` → `jax.scipy`

with a care on traceability

(see JAX documentation)

This JAXed class is called at final MC/global MC in `sgwb.binner`

Ex.) lognormal_bump: $h^2\Omega_{gw}(f) = \Omega_* \exp(-[\log_{10}(f/f_*)/\rho]^2\ )$

```
[model] Setting measured speeds (per sec): {LISA: 350.0}
```

speed measurement at **Cobaya**

```
[model] Setting measured speeds (per sec): {LISA: 4120.0}
```

10 times faster!

#loose gain if powers of arrays are involved in a complex way. But still 2-3 times faster.

Contents

New results with the accelerated code

- MC sampling with more noise parameters

i)foreground shape parameters:

2 amplitudes $(\Omega_{Gal}, \Omega_{Ext})$

➡ **8 parameters** (2 + 6 for shape)

$$h^2\Omega_{GW}^{Gal}(f) \sim f^{n_{Gal}}\left[1+\tanh\left(\frac{f_{knee}-f}{f_2}\right)\right]e^{-\left(\frac{f}{f_1}\right)^{\alpha}} h^2\Omega_{Gal}$$

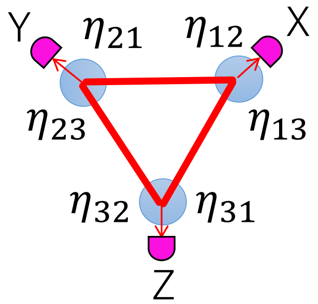
$$h^2\Omega_{GW}^{Ext}(f) \sim f^{n_{Ext}}\ h^2\Omega_{Ext}$$

ii)unequal noise level (Hartwig+ 2023)

2 noise amplitudes (A, P)

(equal noise: $A_{ij} = A$, $P_{ij} = P$)

➡ **6 * 2 parameters**



Acceleration noise A_{ij}

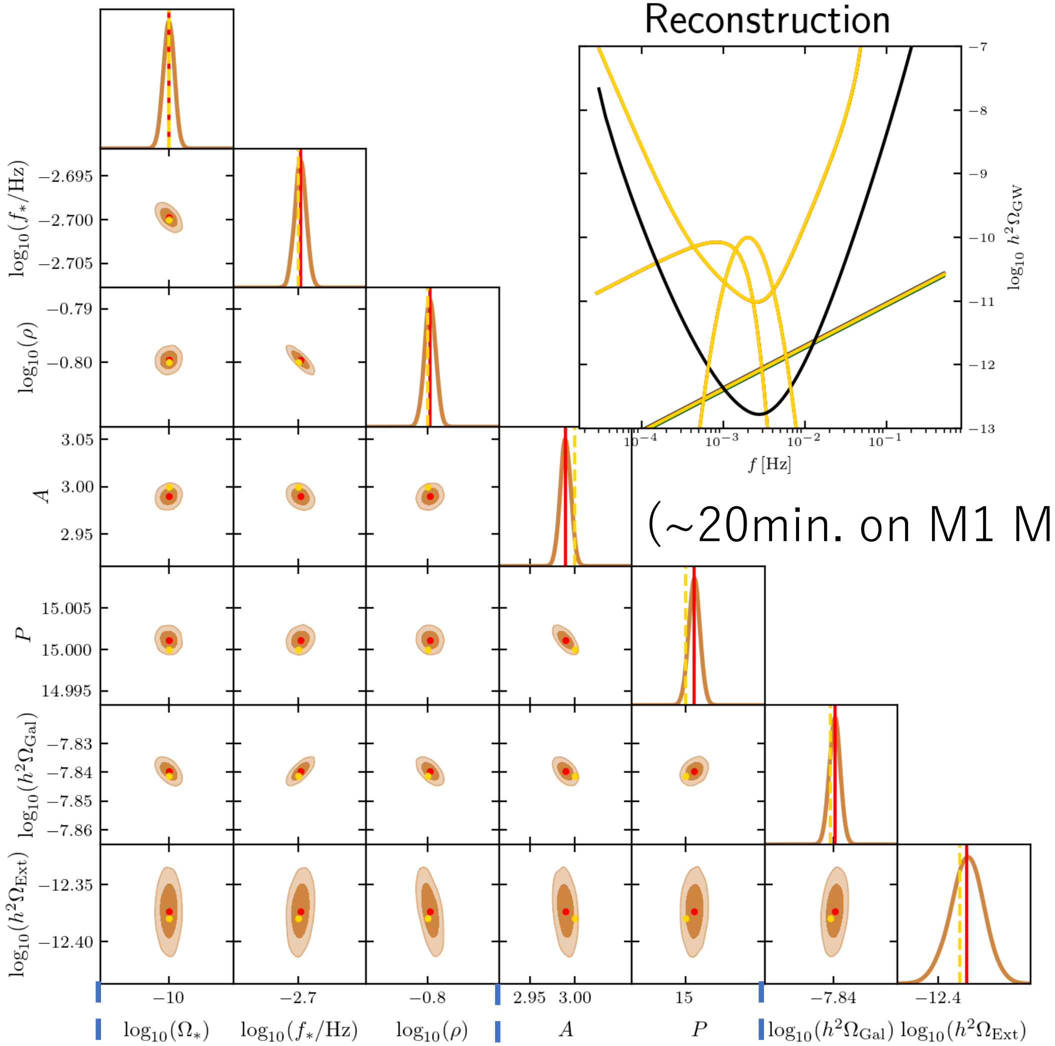
Opto-Mechanical-System P_{ij}

defined at each link η_{ij}

※AET becomes non-diagonal for uneq. noise

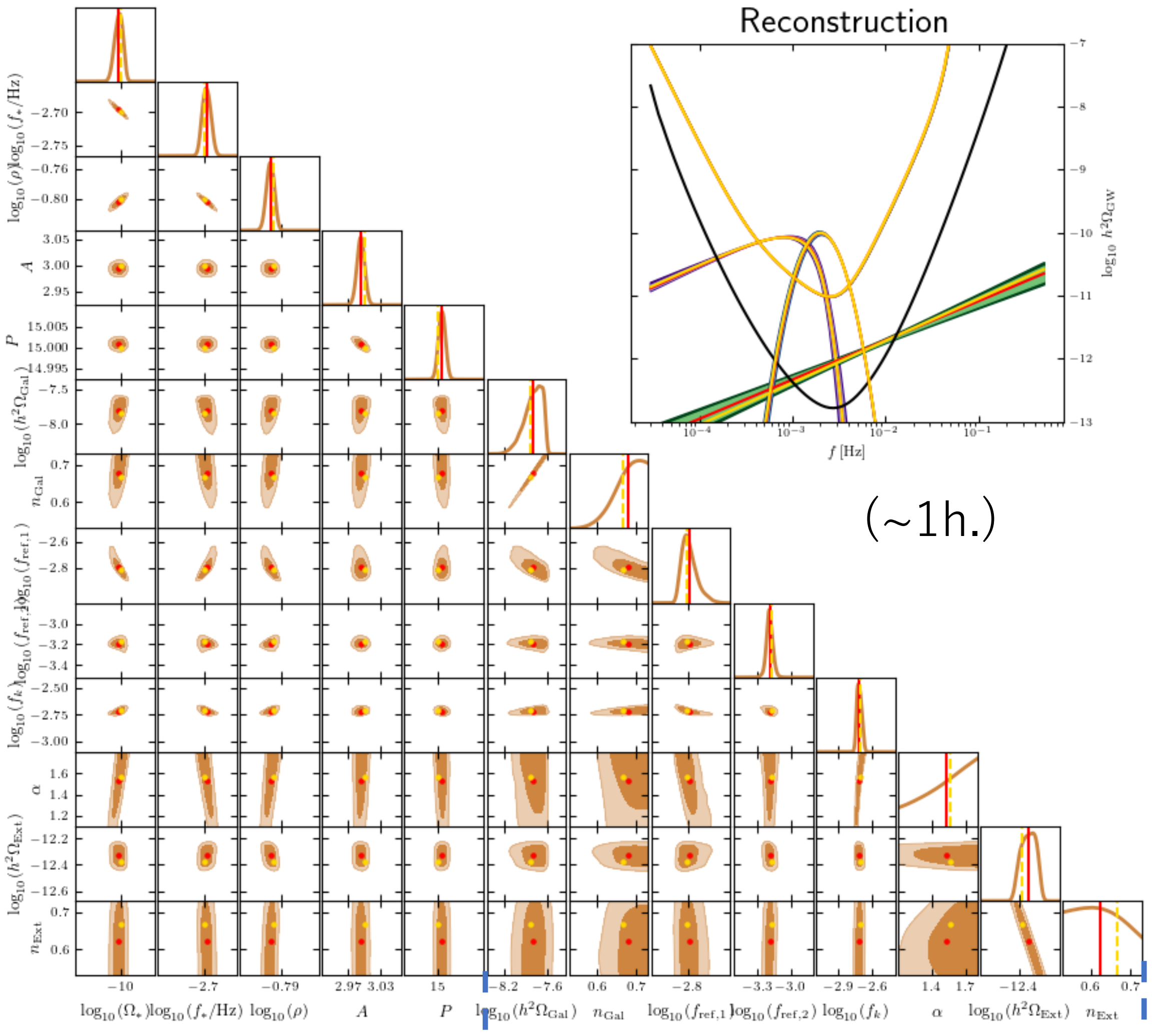
- How does the constraint depend on the assumptions we made so far?

- Foregrounds with more free parameters



(~20min. on M1 Mac)

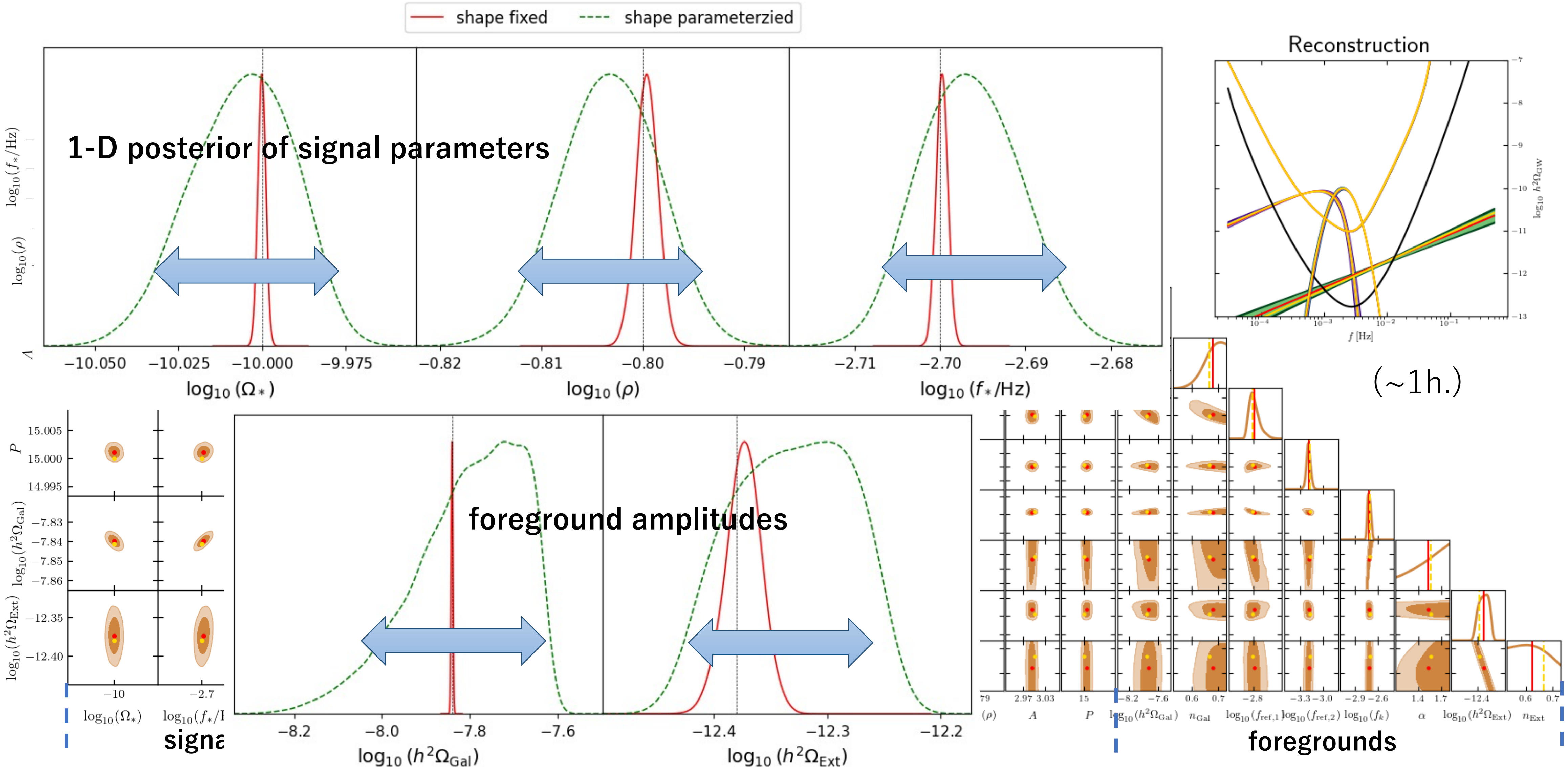
signal | **noise** | **foregrounds**



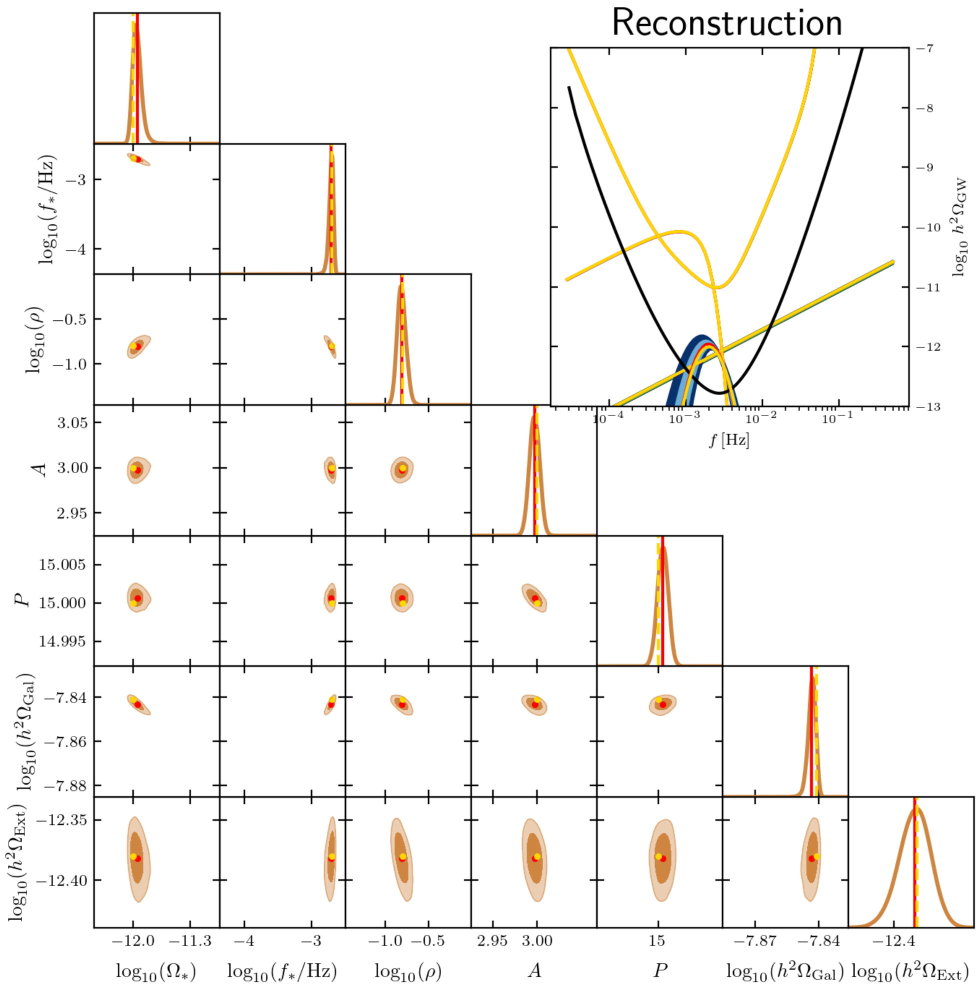
(~1h.)

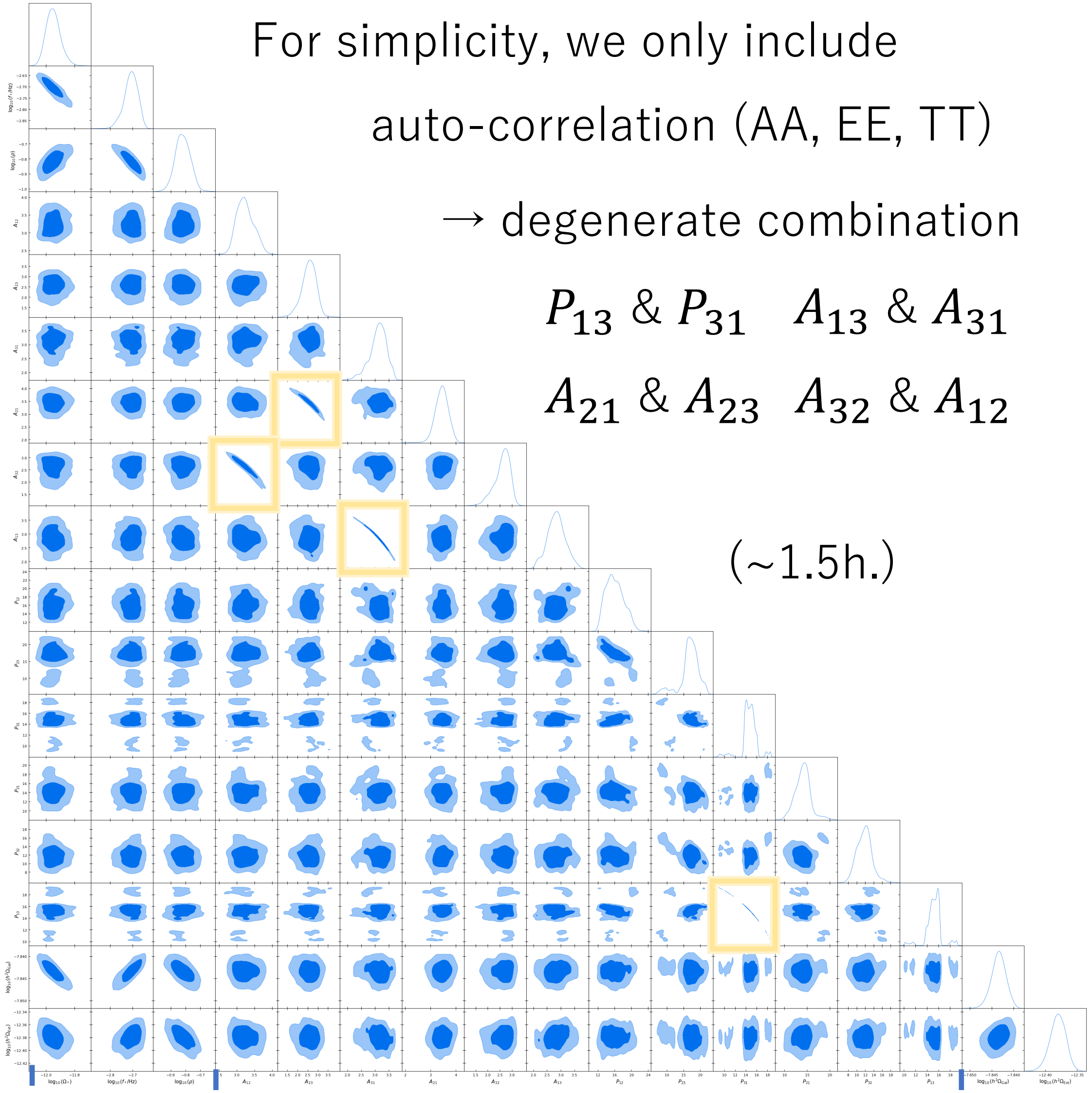
foregrounds

- Foregrounds with more free parameters



- Unequal noise level





For simplicity, we only include auto-correlation (AA, EE, TT)

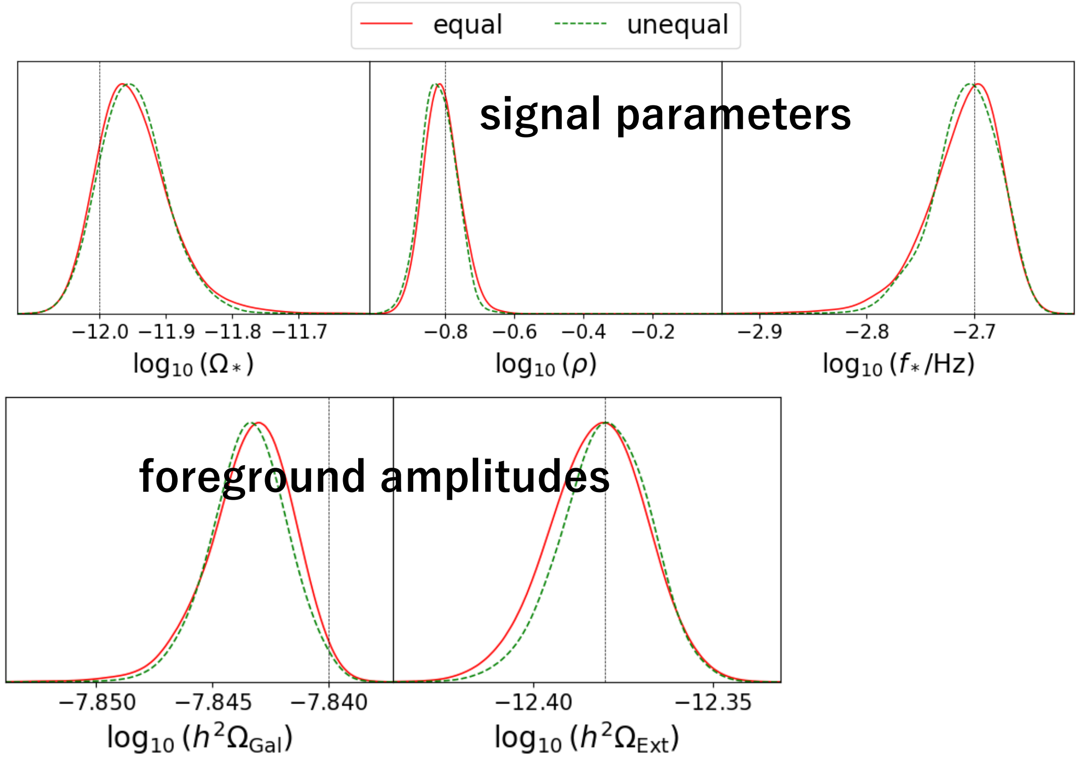
→ degenerate combination

P_{13} & P_{31} A_{13} & A_{31}

A_{21} & A_{23} A_{32} & A_{12}

(~1.5h.)

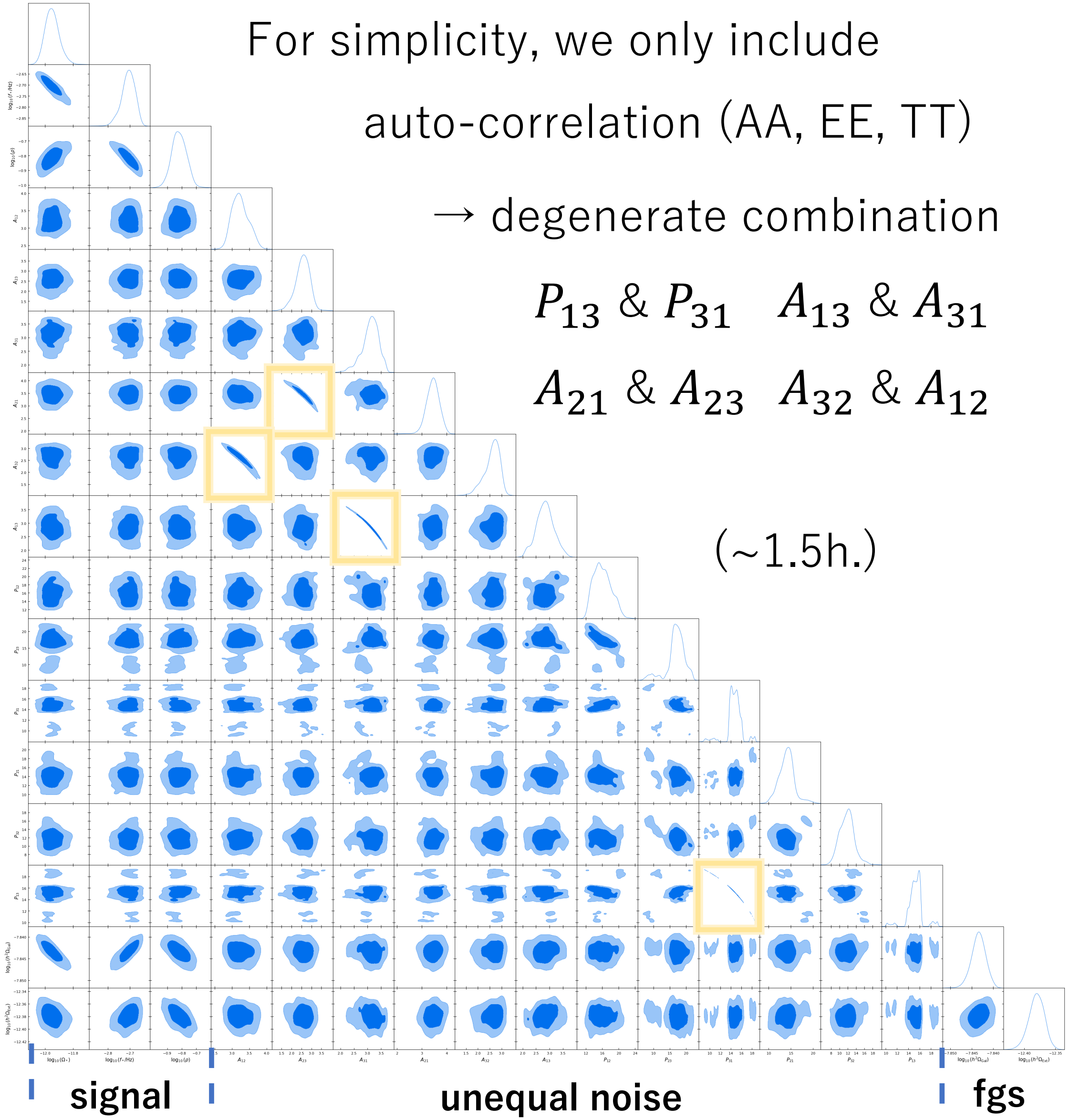
- Unequal noise level



The error sizes of signal parameters

& fg amplitudes don't change much!

(result consistent with Hartwig+ 2023)



For simplicity, we only include

auto-correlation (AA, EE, TT)

→ degenerate combination

P_{13} & P_{31} A_{13} & A_{31}

A_{21} & A_{23} A_{32} & A_{12}

(~1.5h.)

Contents

- ➢Signal reconstruction with SGWBinner
- ➢How can we use JAX?
- ➢New results with the accelerated code
- ➢Summary & Discussion

Summary

✓`JAX` provides an easy way to accelerate your code

- `jax.jit` speeds up repeated operations
- easy conversion of `numpy`-based code with `jax.numpy` & `jax.scipy`

✓Accelerating likelihood → faster MC sampling

- **a few to 10 times faster!!** (depending on templates)

✓Bump signal reconstruction with more noise parameters

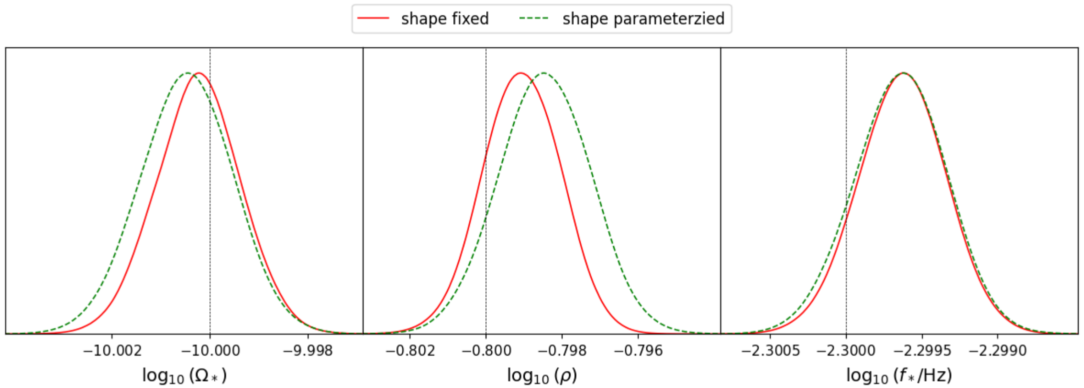
- unequal noise does not affect signal parameter estimate
- foreground shape assumption does when the signal overlaps

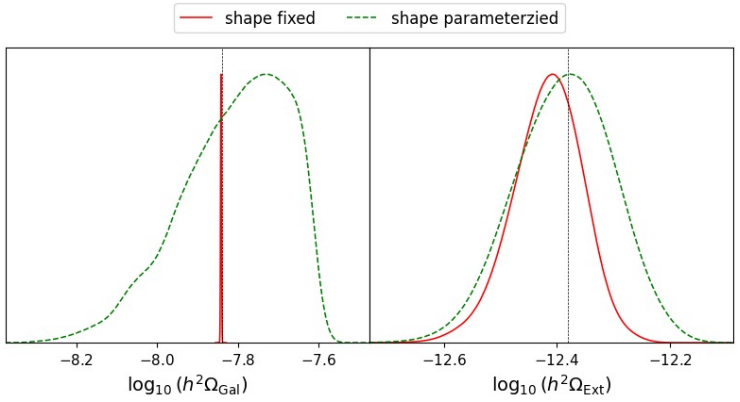
Discussion

- How to speed up your model?
  - analytic: write codes with `jax.numpy` & `jax.scipy` ※not all functions implemented...
  - numeric: interpolation with `scipy` & un-jax signal part...?

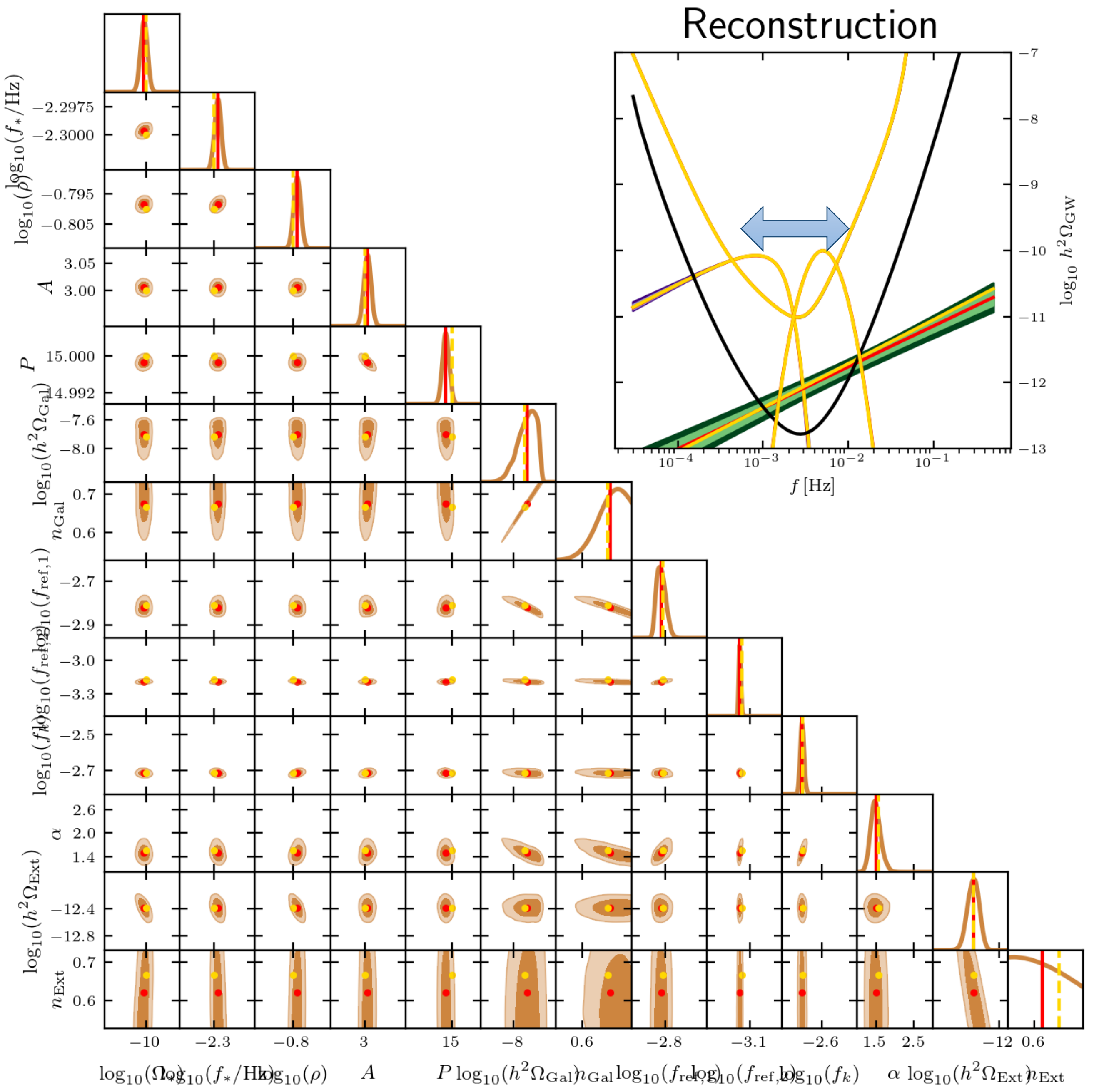
- Other JAX features to be utilized
  - automatic differentiation
    → an easy way to get Fisher information (`jax.hessian`)
  - running on GPU?
    → useful for larger arrays. anisotropy search/circular polarization

- less degenerate signal with more fg parameters



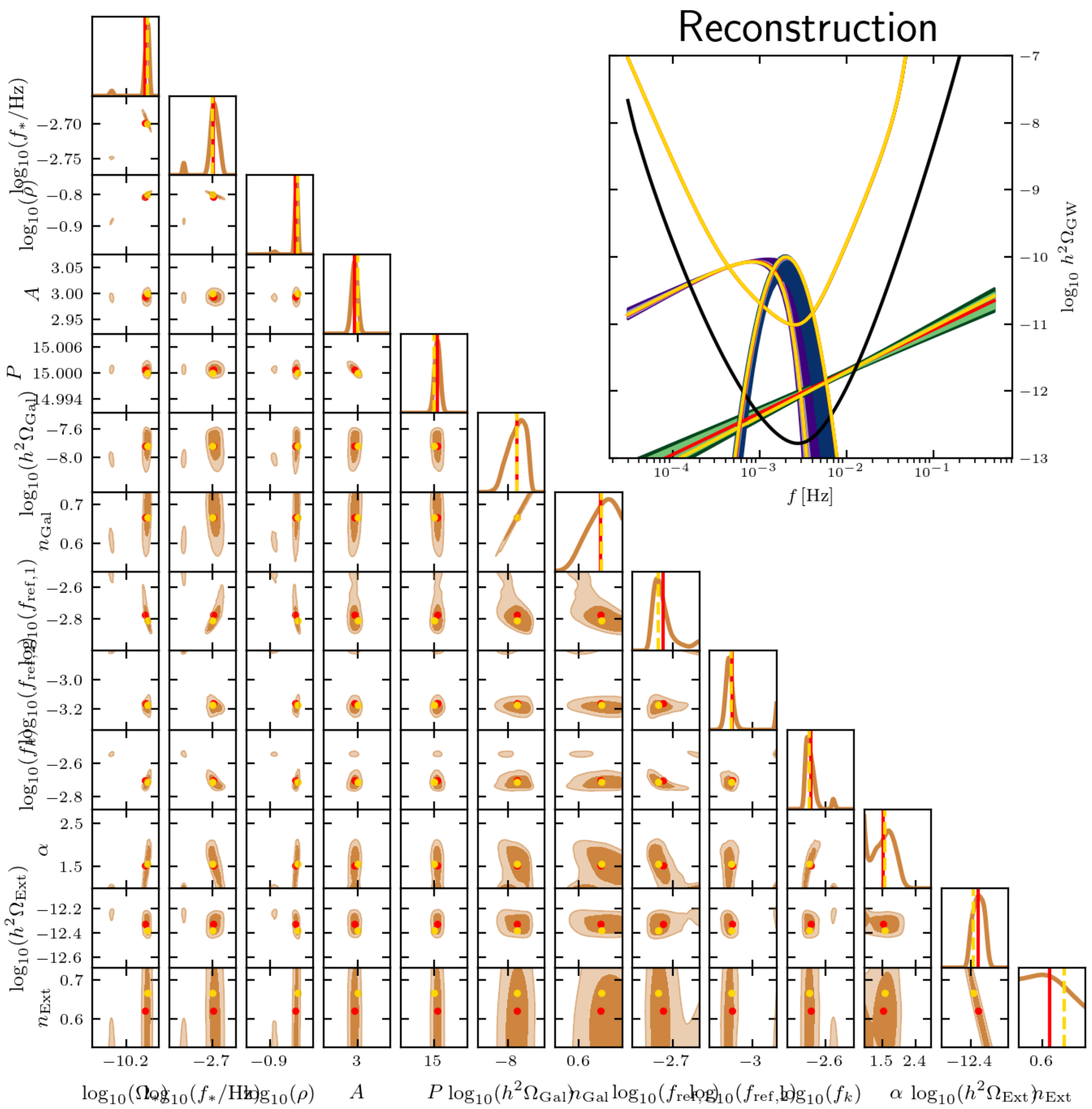


only galactic amplitude affected much

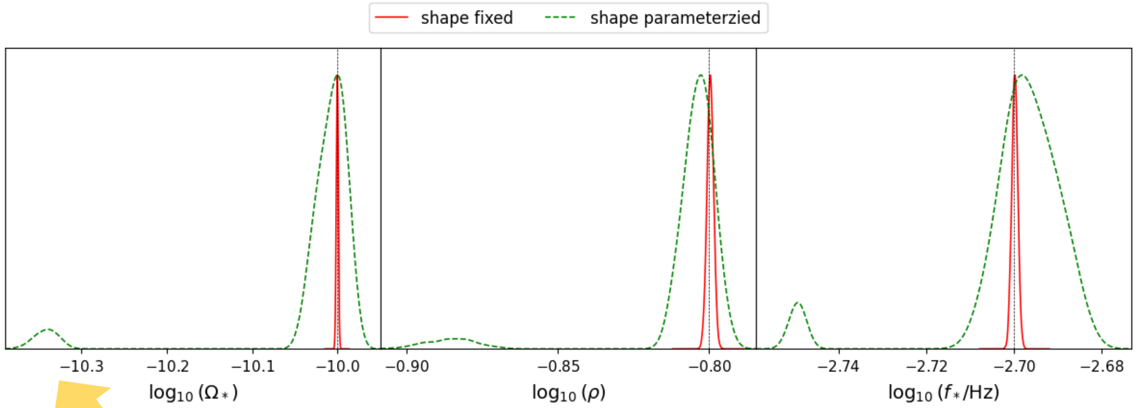


setting for pypolychord

```
{ "nlive" : 1000,
  "num_repeats" : 4d,
  "confidence_for_unbounded": 0.999,
  "precision_criterion": 1e-4 }
```



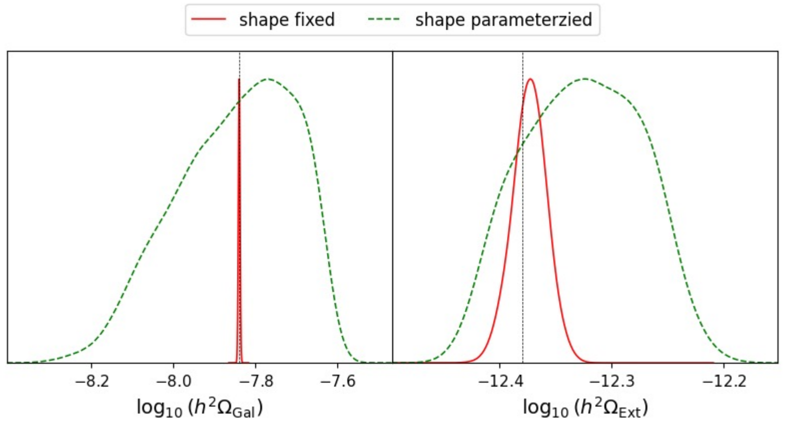
・Change prior of α in degenerate case:



new branch appears

→ fg has longer tail at high freq.

with suppressed signal amplitude



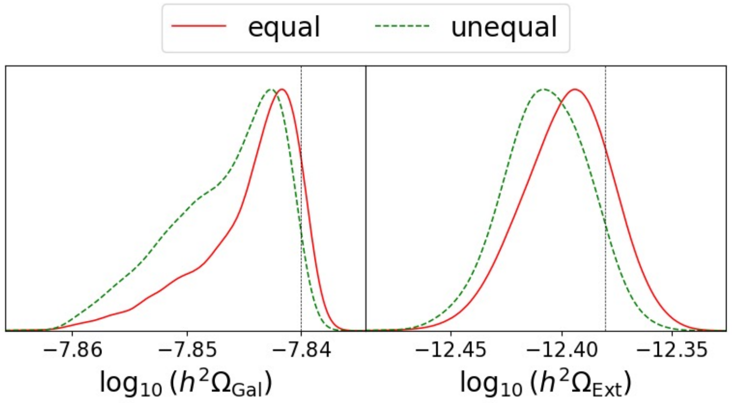
no difference in fg amplitudes

・unequal noise data with weak signal

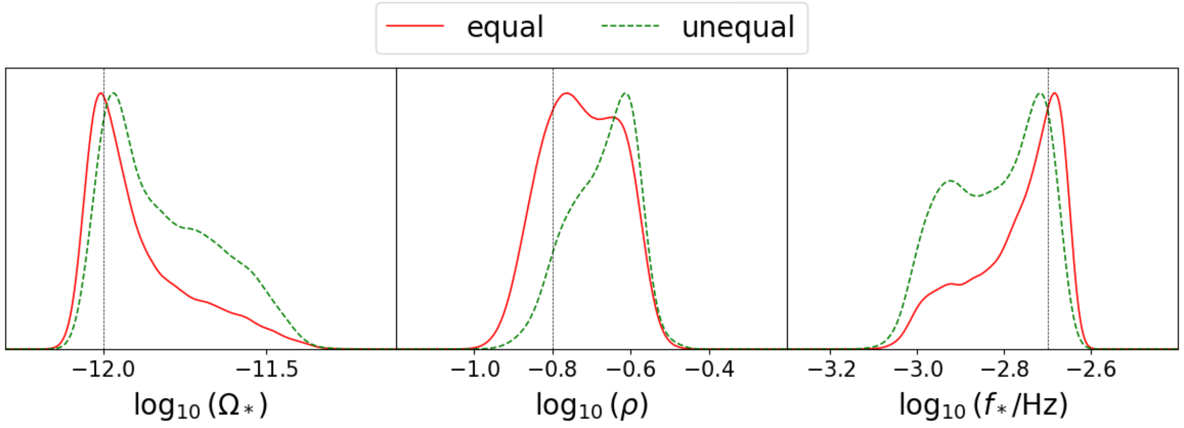
$$A_{ij} = \{3.61, 3.02, 2.87, 3.43, 2.65, 3.45\}$$

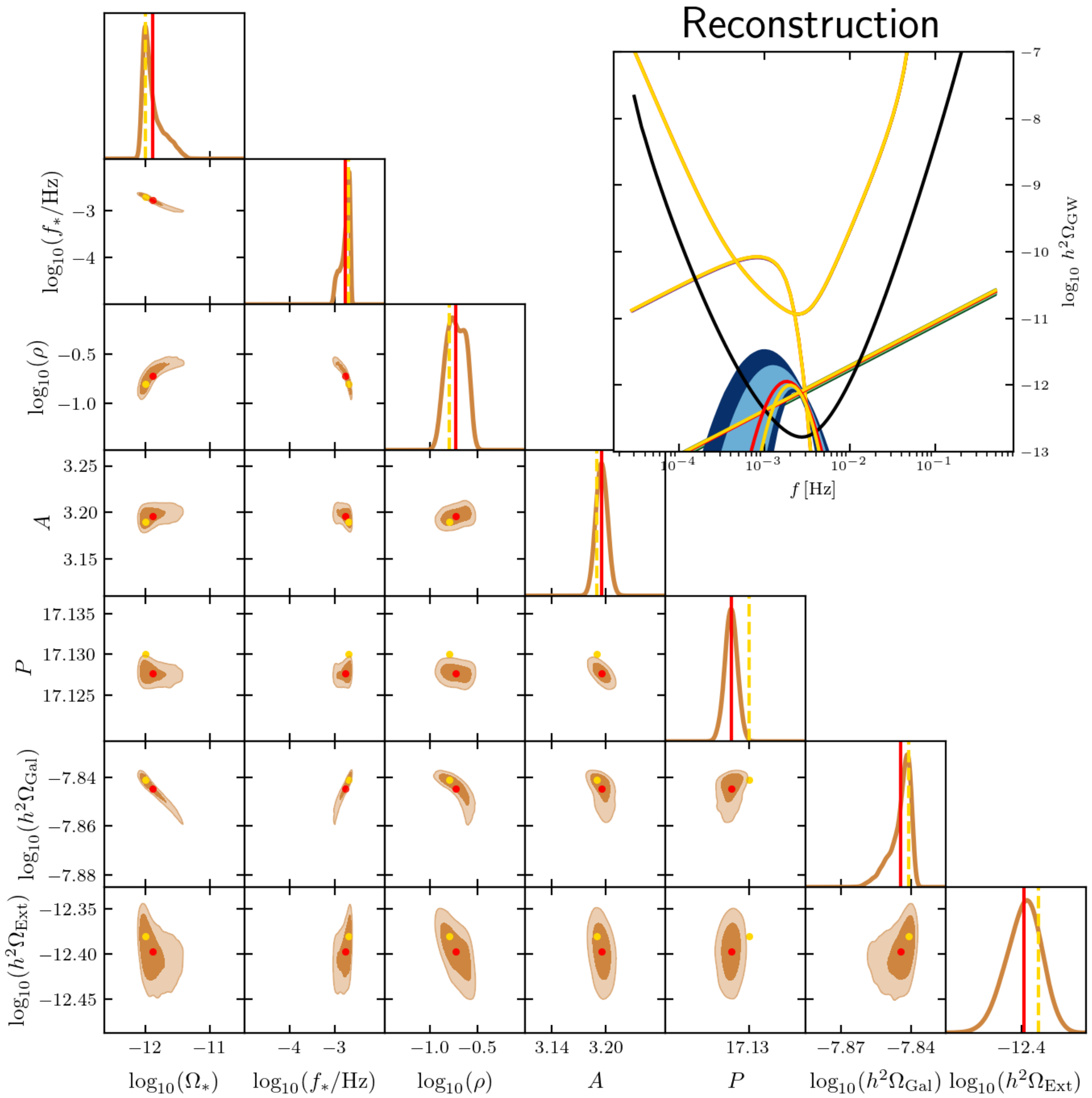
$$P_{ij} = \{14.00, 16.93, 9.43, 21.55, 17.04, 20.83\}$$

→ fit with equal & unequal noise spectrum



Again, the size of error does not change much.





Auto-correlation with unequal noise $P_{(ij)}^{\mathrm{acc/IMS}} \equiv P_{ij}^{\mathrm{acc/IMS}} + P_{ji}^{\mathrm{acc/IMS}}$. $P_{ij}^{acc}(f) \propto A_{ij}^2$, $P_{ij}^{IMS}(f) \propto P_{ij}^2$

$$N_{\mathrm{AA}}(f) = \left\{4\left[(P_{21}^{\mathrm{acc}} + P_{23}^{\mathrm{acc}} + P_{(31)}^{\mathrm{acc}}) + 2P_{(31)}^{\mathrm{acc}}\cos\left(\frac{2\pi fL}{c}\right) + (P_{12}^{\mathrm{acc}} + P_{32}^{\mathrm{acc}} + P_{(31)}^{\mathrm{acc}})\cos^2\left(\frac{2\pi fL}{c}\right)\right]\right.$$
$$\left. + \left[(P_{(12)}^{\mathrm{IMS}} + P_{(23)}^{\mathrm{IMS}} + 2P_{(31)}^{\mathrm{IMS}}) + 2P_{(31)}^{\mathrm{IMS}}\cos\left(\frac{2\pi fL}{c}\right)\right]\right\} \times 2\sin^2\left(\frac{2\pi fL}{c}\right) ,$$

$$N_{\mathrm{EE}}(f) = \left\{4\left[(4P_{12}^{\mathrm{acc}} + P_{21}^{\mathrm{acc}} + P_{23}^{\mathrm{acc}} + 4P_{32}^{\mathrm{acc}} + P_{(31)}^{\mathrm{acc}}) + 2(2P_{(12)}^{\mathrm{acc}} + 2P_{(23)}^{\mathrm{acc}} - P_{(31)}^{\mathrm{acc}})\cos\left(\frac{2\pi fL}{c}\right)\right.\right.$$
$$\left. +(P_{12}^{\mathrm{acc}} + 4P_{21}^{\mathrm{acc}} + P_{32}^{\mathrm{acc}} + 4P_{23}^{\mathrm{acc}} + P_{(31)}^{\mathrm{acc}})\cos^2\left(\frac{2\pi fL}{c}\right)\right]$$
$$\left. +(5P_{(12)}^{\mathrm{IMS}} + 5P_{(23)}^{\mathrm{IMS}} + 2P_{(31)}^{\mathrm{IMS}}) + 2(2P_{(12)}^{\mathrm{IMS}} + 2P_{(23)}^{\mathrm{IMS}} - P_{(31)}^{\mathrm{IMS}})\cos\left(\frac{2\pi fL}{c}\right)\right\} \times \frac{2}{3}\sin^2\left(\frac{2\pi fL}{c}\right)$$

$$N_{\mathrm{TT}}(f) = \left\{2(P_{(12)}^{\mathrm{acc}} + P_{(23)}^{\mathrm{acc}} + P_{(31)}^{\mathrm{acc}})\left[1 - \cos\left(\frac{2\pi fL}{c}\right)\right]^2\right.$$
$$\left. +(P_{(12)}^{\mathrm{IMS}} + P_{(23)}^{\mathrm{IMS}} + P_{(31)}^{\mathrm{IMS}})\left[1 - \cos\left(\frac{2\pi fL}{c}\right)\right]\right\} \times \frac{8}{3}\sin^2\left(\frac{2\pi fL}{c}\right)$$

P_{13} & P_{31}, A_{13} & A_{31}, A_{21} & A_{23}, A_{32} & A_{12} : appear in the same way