Towards GPU-accelerated multimessenger inference of neutron star mergers and dense matter physics

Thibeau Wouters

Utrecht University

Nikhef

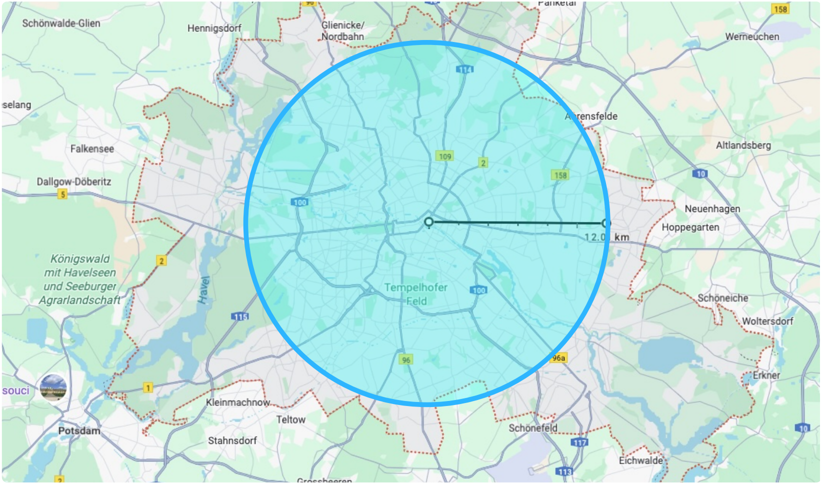
Table of Contents

Contents

Contents

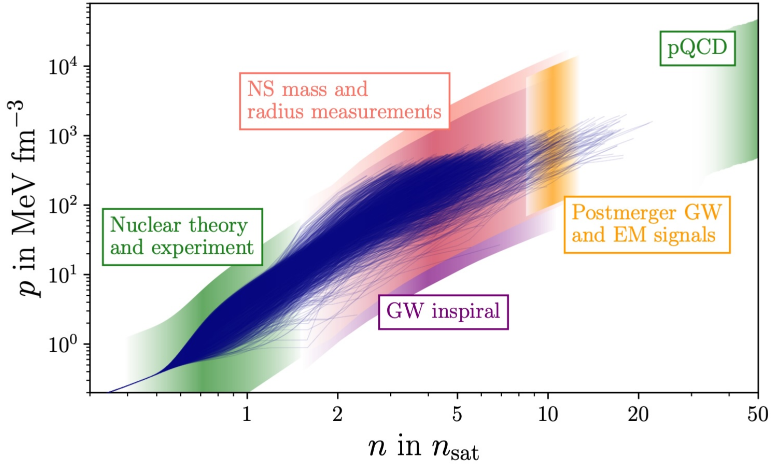
Neutron stars

- Neutron stars: supernova remnants, densest matter in the universe
- $m \sim 1.2 - 2.3 M_{\odot}$, $R \sim 10 - 13$ km



Equation of state

Neutron stars uniquely probe the equation of state (EOS) of dense nuclear matter [1]



Contents

Multimessenger astrophysics

Observe neutron star mergers through gravitational waves and electromagnetic radiation: GW170817 [2, 3]

TODO: Snapshots

Contents

Future GW detectors: Einstein Telescope

Einstein Telescope: Third-generation ground-based GW detector [4, 5]

Current software cannot handle the ET data analysis problem [6]

Future GW detectors: Einstein Telescope

Einstein Telescope: Third-generation ground-based GW detector [4, 5]

- Increased sensitivity
  - Louder signals
  - $10^5 - 10^6$ binary black hole mergers/year (now: $\sim 200/10$ years)
  - $10^4 - 10^5$ binary neutron star mergers/year (now: $2/10$ years)
  - $10^2 - 10^3$ multimessenger events/year (now: $1/10$ years)

Current software cannot handle the ET data analysis problem [6]

Future GW detectors: Einstein Telescope

Einstein Telescope: Third-generation ground-based GW detector [4, 5]

- Increased sensitivity
  - Louder signals
  - $10^5 - 10^6$ binary black hole mergers/year (now: $\sim 200/10$ years)
  - $10^4 - 10^5$ binary neutron star mergers/year (now: 2/10 years)
  - $10^2 - 10^3$ multimessenger events/year (now: 1/10 years)
- Observe from 5 Hz (now: 20 Hz)
  - Longer signals: binary neutron star: 2 hours vs 2 minutes
  - Signals will overlap

Current software cannot handle the ET data analysis problem [6]

My research focus: why – how – what

Why?

To make inference of multimessenger astrophysics scalable

- Prepare for future detectors
- Understand systematic effects through simulated data

My research focus: why – how – what

Why?

To make inference of multimessenger astrophysics scalable

- Prepare for future detectors
- Understand systematic effects through simulated data

How?

Without compromises to flexibility and accuracy

- Accelerate with GPU hardware, differentiable programming
- Use machine learning to assist inference, not replace it

My research focus: why – how – what

Why?

To make inference of multimessenger astrophysics scalable

- Prepare for future detectors
- Understand systematic effects through simulated data

How?

Without compromises to flexibility and accuracy

- Accelerate with GPU hardware, differentiable programming
- Use machine learning to assist inference, not replace it

What?

GPU-accelerated Bayesian joint inference framework of neutron star mergers and dense matter physics

Contents

Contents

Parameter estimation: Bayesian inference

How do we "measure" source parameters θ for data d?

Bayesian inference:

$$\text{posterior} = p(\theta|d,\mathcal{M}) = \frac{p(d|\theta,\mathcal{M})p(\theta|\mathcal{M})}{p(d|\mathcal{M})} = \frac{\text{likelihood} \times \text{prior}}{\text{evidence}}$$

Ingredients:

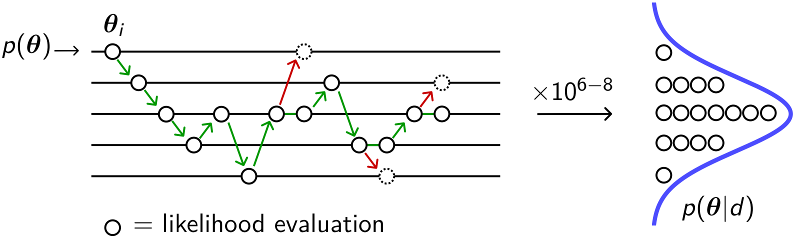
- Prior: specified by users
- Likelihood: often costly
- Posterior: intractable, needs stochastic samplers
- Evidence: model selection

Parameter estimation: Markov chain Monte Carlo

How do we sample the posterior? Markov chain Monte Carlo

- N chains $\boldsymbol{\theta}_i$ explore posterior in parallel
- Evolve chains to new position: proposal
- Compute likelihood $\rightarrow$ accept/reject

Alteratives: nested sampling, sequential Monte Carlo



Computational aspects

$$\text{Total runtime} \approx N_{\text{likelihood}} \times \tau_{\text{likelihood}}$$

Computational aspects

$$\text{Total runtime} \approx N_{\text{likelihood}} \times \tau_{\text{likelihood}}$$

- $N_{\text{likelihood}}$: total number of likelihood evaluations
  - Dimensionality of $\boldsymbol{\theta}$
  - Signal-to-noise ratio
  - Multimodality/shape posterior
  - Efficiency of sampler (proposals)

Computational aspects

$$\text{Total runtime} \approx N_{\text{likelihood}} \times \tau_{\text{likelihood}}$$

- $N_{\text{likelihood}}$: total number of likelihood evaluations
  - Dimensionality of $\boldsymbol{\theta}$
  - Signal-to-noise ratio
  - Multimodality/shape posterior
  - Efficiency of sampler (proposals)

- $\tau_{\text{likelihood}}$: speed of a single likelihood evaluation
  - Complexity model/likelihood function
  - Parallelization of likelihood evaluations
  - Hardware

Computational aspects

$$\text{Total runtime} \approx N_{\text{likelihood}} \times \tau_{\text{likelihood}}$$

- $N_{\text{likelihood}}$: total number of likelihood evaluations
    - Dimensionality of $\boldsymbol{\theta}$
    - Signal-to-noise ratio
    - Multimodality/shape posterior
    - **Efficiency of sampler (proposals)**
- $\tau_{\text{likelihood}}$: speed of a single likelihood evaluation
    - Complexity model/likelihood function
    - **Parallelization of likelihood evaluations**
    - **Hardware**

How can we optimize this?

Contents

jax "is a Python library for accelerator-oriented array computation and program transformation, designed for high-performance numerical computing and large-scale machine learning." [7]

In particular:

- Python: integrate into existing workflows
- Focus on arrays: widespread applications
- Numpy-like API: (`numpy` $\rightarrow$ `jax.numpy`)

JAX

jax "is a Python library for accelerator-oriented array computation and program transformation, designed for high-performance numerical computing and large-scale machine learning." [7]

In particular:

- Python: integrate into existing workflows
- Focus on arrays: widespread applications
- Numpy-like API: (`numpy` → `jax.numpy`)

Gamechangers:

- GPU accelerators
- Composable function transformations: `jit`, `grad`, `vmap` & `pmap`

JAX – Function transformations

- `jit`: Just-in-time compilation to XLA (CPU/GPU/TPU)

TODO: Some figure for autodiff

JAX – Function transformations

- `jit`: Just-in-time compilation to XLA (CPU/GPU/TPU)
- `grad`: Automatic differentiation: compute gradients with chain rule

TODO: Some figure for autodiff

JAX – Function transformations

- `jit`: Just-in-time compilation to XLA (CPU/GPU/TPU)
- `grad`: Automatic differentiation: compute gradients with chain rule
- `vmap`, `pmap`: Vectorization, batch processing, parallelization

TODO: Some figure for autodiff

JAX – Function transformations

- `jit`: Just-in-time compilation to XLA (CPU/GPU/TPU)
- `grad`: Automatic differentiation: compute gradients with chain rule
- `vmap`, `pmap`: Vectorization, batch processing, parallelization
- Composable, e.g.:
  - Higher-order derivatives: `jaxgrad(jaxgrad(f))`
  - Evolve N chains in parallel along gradient of the likelihood

$$\boldsymbol{\theta} \leftarrow \alpha * \texttt{vmap}\left(\texttt{jit}(\texttt{grad}(\texttt{logL}(\boldsymbol{\theta})))\right)$$

TODO: Some figure for autodiff

Computational aspects

$$\text{Total runtime} \approx N_{\text{likelihood}} \times \tau_{\text{likelihood}}$$

- $N_{\text{likelihood}}$: total number of likelihood evaluations
  - Dimensionality of $\boldsymbol{\theta}$
  - Signal-to-noise ratio
  - Multimodality/shape posterior
  - **Efficiency of sampler (proposals)**
- $\tau_{\text{likelihood}}$: speed of a single likelihood evaluation
  - Complexity model/likelihood function
  - **Parallelization of likelihood evaluations**
  - **Hardware**

How can we optimize this?

Contents

Normalizing flows

Q: Given samples $\{x\} \sim p(x)$, how can we get $p(x)$?

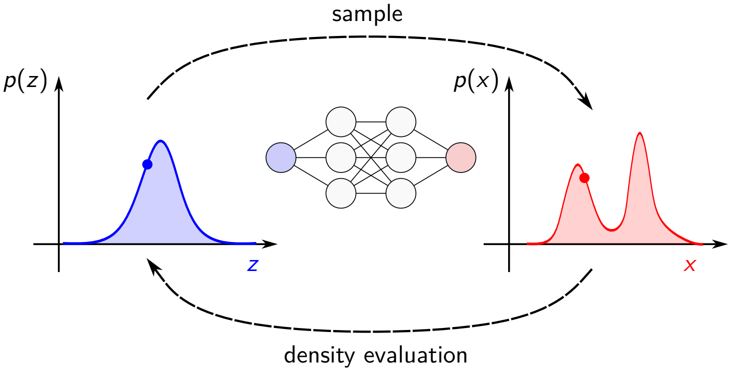
- Evaluate density at any point
- Generate new samples

Normalizing flows

Q: Given samples $\{x\} \sim p(x)$, how can we get $p(x)$?

- Evaluate density at any point
- Generate new samples

A: Normalizing flows: generative machine learning model, bijection between latent space (Gaussian) and data space

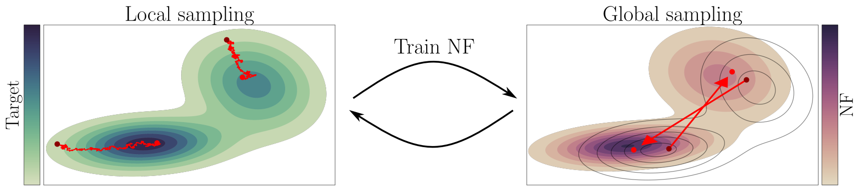


Contents

flowMC

flowMC [8, 9]: normalizing flow-enhanced MCMC

1. Run gradient-based sampler (local sampler)
2. Train normalizing flow on MCMC chains
3. Propose samples from the normalizing flow (global sampler)
4. Repeat



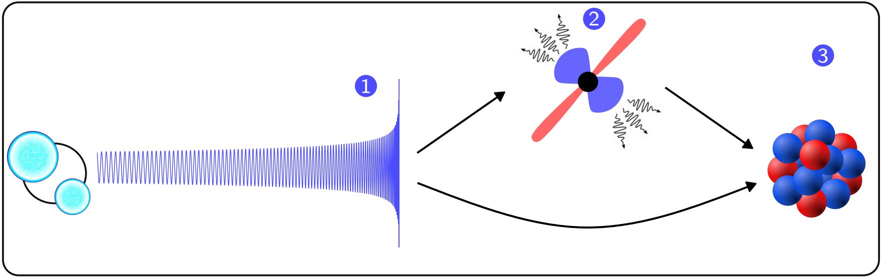
Contents

Overview

Analyzing a multi-messenger **binary neutron star** signal:

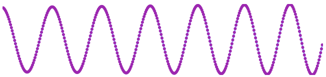
1. **Gravitational waves**
2. Electromagnetic counterparts
3. Nuclear equation of state

JAX + flowMC

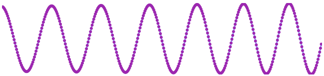


Gravitational waves: RIPPLE

- Waveforms on GPU: $\mathcal{O}(10^3)$ faster
- RIPPLE [10]: from LALSUITE to JAX



Gravitational waves: RIPPLE



- Waveforms on GPU: $\mathcal{O}(10^3)$ faster
- RIPPLE [10]: from LALSUITE to JAX
- Waveforms available in RIPPLE:
  - Binary black holes:
    - `IMRPhenomXAS`
    - `IMRPhenomD`
    - `IMRPhenomPv2`
  - Binary neutron stars:
    - `TaylorF2`
    - `IMRPhenomD_NRTidalv2`
    - `IMRPhenomPv_NRTidalv2` (by Nihar Gupte)
- Also see GWFAST [11] and SFTS [12]

Gravitational waves: Jim

Parameter estimation: Jim [13, 14]: ripple + flowMC

- Verified for binary neutron stars in current generation detectors [14]
- Matches bilby, passes *pp*-test

Gravitational waves: Jim

Parameter estimation: Jim [13, 14]: ripple + flowMC

- Verified for binary neutron stars in current generation detectors [14]
- Matches bilby, passes *pp*-test
- Runtime: from hours to minutes

Event	Waveform	Jim (1 GPU)	pBilby (480 cores)	RB-Bilby (24 cores)	ROQ-Bilby (24 cores)
GW170817	TF2	15.33 min	9.64 h	3.8 h	–
	NRTv2	25.59 min	10.99 h	4.11 h	1.65 h
GW190425	TF2	18.30 min	8.18 h	2.81 h	–
	NRTv2	21.20 min	4.91 h	2.42 h	0.97 h
Injection	TF2	24.76 min	–	–	–
	NRTv2	18.02 min	–	–	–

Gravitational waves: JIM

Parameter estimation: JIM [13, 14]: RIPPLE + FLOWMC

- Verified for binary neutron stars in current generation detectors [14]
- Matches BILBY, passes *pp*-test
- Runtime: from hours to minutes
- Up to $5-10\times$ more effective samples (normalizing flow)

	Effective sample size
JIM	4.1×10^4
PBILBY	6.5×10^3
RB-BILBY	5.8×10^3
ROQ-BILBY	7.4×10^3

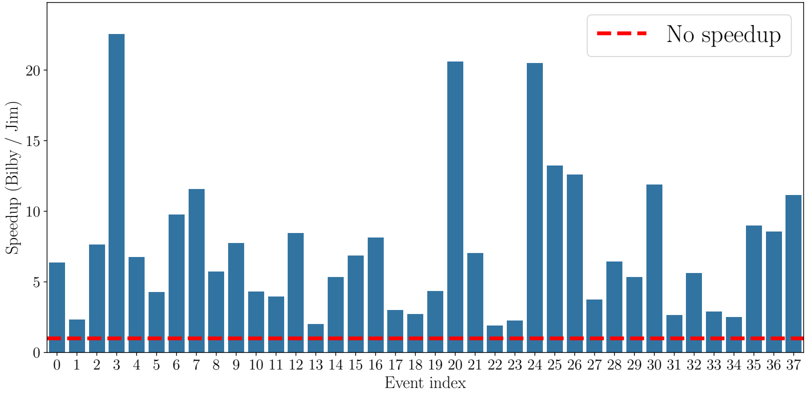
Gravitational waves: JIM

Parameter estimation: JIM [13, 14]: RIPPLE + FLOWMC

- Verified for binary neutron stars in current generation detectors [14]
- Matches BILBY, passes *pp*-test
- Runtime: from hours to minutes
- Up to $5 - 10\times$ more effective samples (normalizing flow)
- More cost-effective/energy-efficient

		kWh	CO_2 [10^3 kg]	Trees
JIM		34	11	0.55
PBILBY		4127	1354	67.68
RB-BILBY		80	26	1.32
ROQ-BILBY	sampling	32	10	0.52
	precompute	27	9	0.44

- Re-analyze GWTC-3 with `IMRPhenomPv2`
- JIM on A800 GPU: on average $\sim$ 7 times faster than BILBY on 16 CPUs, **without compromises**
- Work in progress!



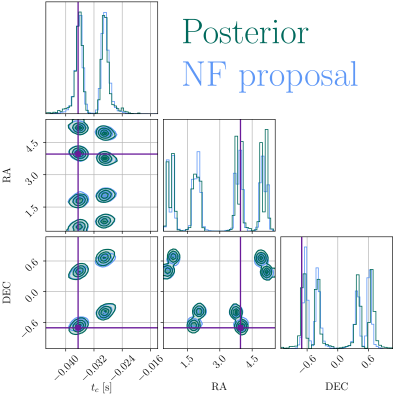
Einstein Telescope

Work in progress: showing proof-of-concepts!

Einstein Telescope

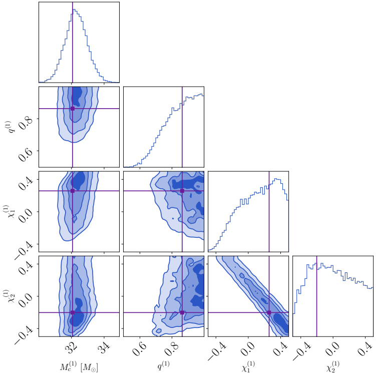
Work in progress: showing proof-of-concepts!

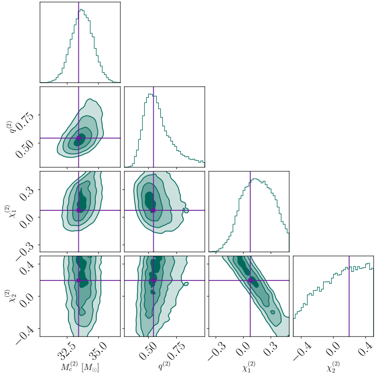
- ET posteriors are multimodal
- Normalizing flows help to jump between modes



Overlapping signals

- Assess scaling of JIM: BBH+BBH with LIGO-Virgo
  - 2 binary black hole mergers: 22 parameters
  - $M_c^{(1)} = 32 M_\odot$, $M_c^{(2)} = 33 M_\odot$, $\Delta t = 70$ ms
  - $\mathrm{SNR}^{(1)} = 25.76$, $\mathrm{SNR}^{(2)} = 25.24$
  - 1h28m on H100 GPU (vs several days on 16 CPUs [15])

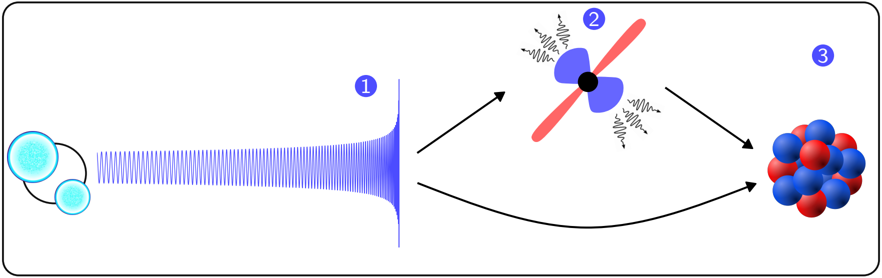




Overview

Analyzing a multi-messenger **binary neutron star** signal:

1. Gravitational waves
2. **Electromagnetic counterparts**
3. Nuclear equation of state

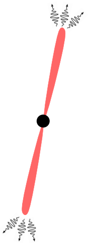


Electromagnetic counterparts (Hauke Koehn, Tim Dietrich)

- Kilonovae and GRB afterglows
- Code libraries too large to 'jaxify' (e.g. AFTERGLOWPY [16])

Electromagnetic counterparts (Hauke Koehn, Tim Dietrich)

- Kilonovae and GRB afterglows
- Code libraries too large to 'jaxify' (e.g. AFTERGLOWPY [16])
- Neural network emulators for inference: FIESTA [17]

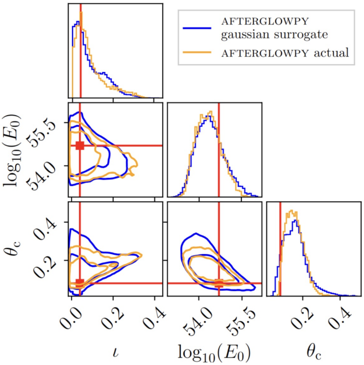


FIESTA

- 1m36s
- 1 H100 GPU

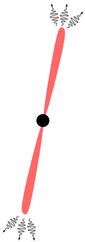
AFTERGLOWPY

- 4 hours
- 30 CPUs



Electromagnetic counterparts (Hauke Koehn, Tim Dietrich)

- Kilonovae and GRB afterglows
- Code libraries too large to 'jaxify' (e.g. AFTERGLOWPY [16])
- Neural network emulators for inference: FIESTA [17]
- Scales well for systematics 'nuisance parameters'

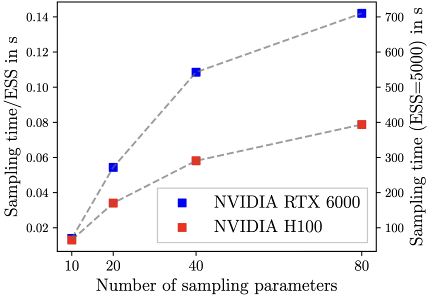


FIESTA

- 1m36s
- 1 H100 GPU

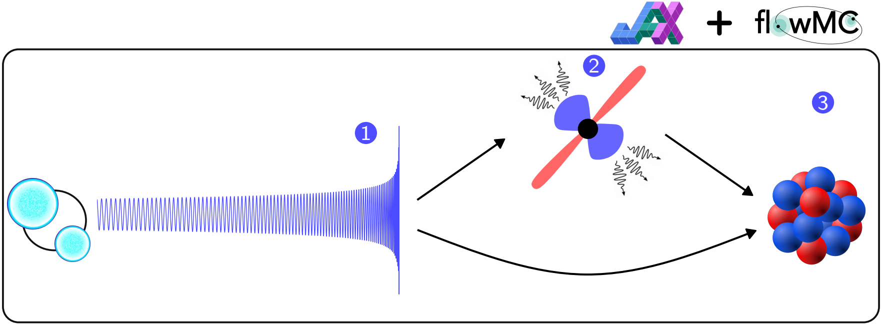
AFTERGLOWPY

- 4 hours
- 30 CPUs



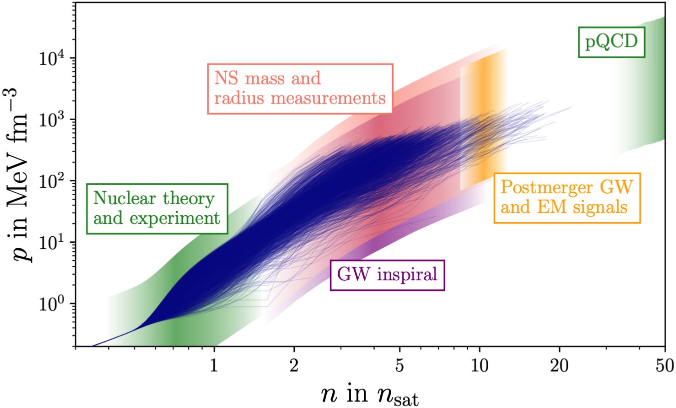
Analyzing a multi-messenger **binary neutron star** signal:

1. Gravitational waves
2. Electromagnetic counterparts
3. **Nuclear equation of state**



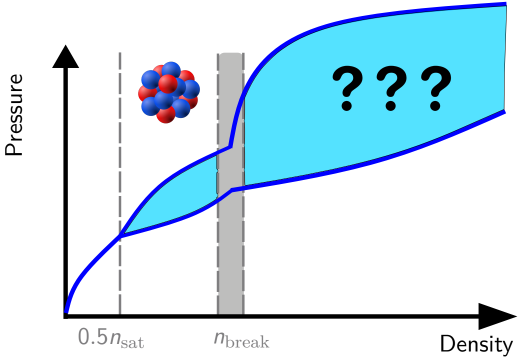
Equation of state inference – warmup

- How do we constrain the EOS from neutron star observations?
- Define parametrization and likelihood
- GWs from binary neutron stars inspiral as example



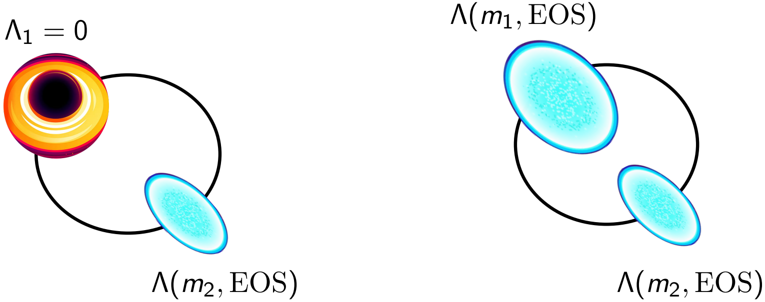
Equation of state inference – parametrization

- $n \leq \frac{1}{2} n_{\mathrm{sat}}$: fixed
- $\frac{1}{2} n_{\mathrm{sat}} \leq n \leq n_{\mathrm{break}}$: metamodel EOS, nuclear physics inspired
- $n \leq n_{\mathrm{break}}$: agnostic extension
- > 26 (!) parameters $\boldsymbol{\theta}_{\mathrm{EOS}}$



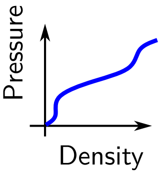
Tidal deformability

- Neutron stars are tidally deformed in a binary, quantified by tidal deformability Λ: affect phase of GWs
- Neutron stars: $\Lambda = \Lambda(m, \mathrm{EOS})$, black holes: $\Lambda = 0$
- GWs give us $p(m_1, m_2, \Lambda_1, \Lambda_2 | d)$ for EOS inference



Equation of state

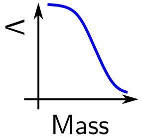
- To predict neutron star properties, we solve the TOV equations: ordinary differential equations (ODEs)



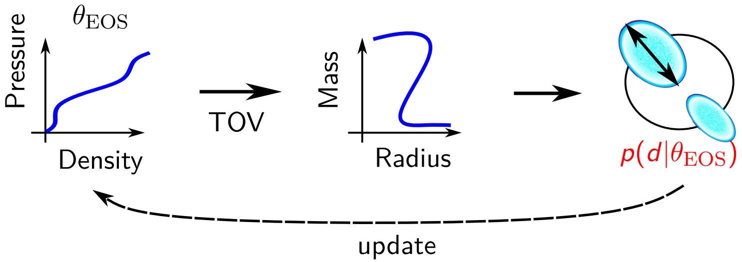
Solve TOV equations

$$\frac{\mathrm{d}p}{\mathrm{d}r} = -\frac{\varepsilon(r)m(r)}{r^2}\frac{\left[1+\frac{p(r)}{\varepsilon(r)}\right]\left[1+\frac{4\pi r^3 p(r)}{m(r)}\right]}{\left[1-\frac{2m(r)}{r}\right]}$$

$$\frac{\mathrm{d}m}{\mathrm{d}r} = 4\pi r^2 \varepsilon(r)\,,$$

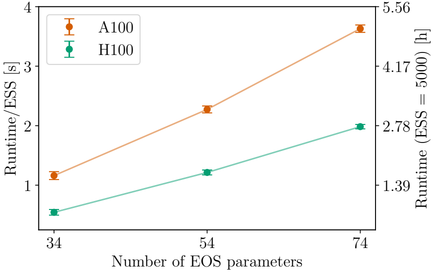


- To predict neutron star properties, we solve the TOV equations: ordinary differential equations (ODEs)
- Done for each sample θ_{EOS}: costly likelihood
- How to make this scalable without compromises?

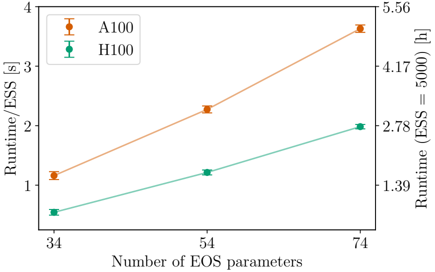


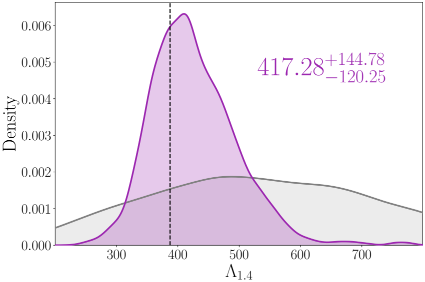
Jester

- Jester [18]: jax-based TOV solver
  - 1000× faster, without compromises!
  - Full inference in ∼hours



- JESTER [18]: JAX-based TOV solver
  - 1000× faster, without compromises!
  - Full inference in ~hours
- JIM+JESTER: from GWs to EOS in a few hours
  - Example: 20 binary neutron stars in O5
- Open up possibilities to study systematics in EOS inference





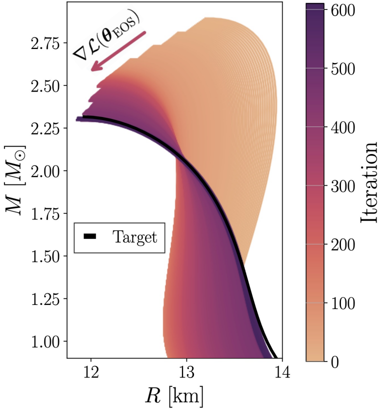
Auto-differentiable ODE solvers

- ODE solvers written in JAX are auto-differentiable
- Frame inference as optimization problem:
  - Gradient descent on loss function $\mathcal{L}(\boldsymbol{\theta}_{\rm EOS})$

Auto-differentiable ODE solvers

- ODE solvers written in JAX are auto-differentiable
- Frame inference as optimization problem:
  - Gradient descent on loss function $\mathcal{L}(\boldsymbol{\theta}_{\rm EOS})$

$$\mathcal{L}(\boldsymbol{\theta}_{\rm EOS}) = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{R_i(\boldsymbol{\theta}_{\rm EOS}) - \hat{R}_i}{\hat{R}_i}\right|$$



Contents

and now, for something completely different...

Contents

Conclusion

- Progress on scalable Bayesian inference, with minimal pre-training
- Hybrid acceleration: GPUs + normalizing flow proposals
  - JAX/GPU: faster likelihoods
  - FLOWMC: sampling converges faster
- Simulators in JAX can remove the need for emulators (GW, TOV)
- Auto-differentiable ODE solvers: inference as optimization problem

Let's talk!

Thanks for listening!

References I

[1] Hauke Koehn et al. "From existing and new nuclear and astrophysical constraints to stringent limits on the equation of state of neutron-rich dense matter". In: (Feb. 2024). arXiv: 2402.04172 [astro-ph.HE].

[2] B. P. Abbott et al. "GW170817: Observation of Gravitational Waves from a Binary Neutron Star Inspiral". In: *Phys. Rev. Lett.* 119.16 (2017), p. 161101. DOI: 10.1103/PhysRevLett.119.161101. arXiv: 1710.05832 [gr-qc].

[3] B. P. Abbott et al. "Gravitational Waves and Gamma-rays from a Binary Neutron Star Merger: GW170817 and GRB 170817A". In: *Astrophys. J. Lett.* 848.2 (2017), p. L13. DOI: 10.3847/2041-8213/aa920c. arXiv: 1710.05834 [astro-ph.HE].

[4] Michele Maggiore et al. "Science Case for the Einstein Telescope". In: *JCAP* 03 (2020), p. 050. DOI: 10.1088/1475-7516/2020/03/050. arXiv: 1912.02622 [astro-ph.CO].

[5] Adrian Abac et al. "The Science of the Einstein Telescope". In: (Mar. 2025). arXiv: 2503.12263 [gr-qc].

[6] Qian Hu and John Veitch. "Costs of Bayesian Parameter Estimation in Third-Generation Gravitational Wave Detectors: a Review of Acceleration Methods". In: (Dec. 2024). arXiv: 2412.02651 [gr-qc].

[7] James Bradbury et al. *JAX: composable transformations of Python+NumPy programs*. Version 0.3.13. 2018. URL: http://github.com/jax-ml/jax.

References II

[8] Marylou Gabrié, Grant M. Rotskoff, and Eric Vanden-Eijnden. "Adaptive Monte Carlo augmented with normalizing flows". In: *Proc. Nat. Acad. Sci.* 119.10 (2022), e2109420119. DOI: 10.1073/pnas.2109420119. arXiv: 2105.12603 [physics.data-an].

[9] Kaze W. k. Wong, Marylou Gabrié, and Daniel Foreman-Mackey. "flowMC: Normalizing flow enhanced sampling package for probabilistic inference in JAX". In: *J. Open Source Softw.* 8.83 (2023), p. 5021. DOI: 10.21105/joss.05021. arXiv: 2211.06397 [astro-ph.IM].

[10] Thomas D. P. Edwards et al. "Differentiable and hardware-accelerated waveforms for gravitational wave data analysis". In: *Phys. Rev. D* 110.6 (2024), p. 064028. DOI: 10.1103/PhysRevD.110.064028. arXiv: 2302.05329 [astro-ph.IM].

[11] Francesco Iacovelli et al. "GWFAST: A Fisher Information Matrix Python Code for Third-generation Gravitational-wave Detectors". In: *Astrophys. J. Supp.* 263.1 (2022), p. 2. DOI: 10.3847/1538-4365/ac9129. arXiv: 2207.06910 [astro-ph.IM].

[12] Rodrigo Tenorio and Davide Gerosa. "Scalable data-analysis framework for long-duration gravitational waves from compact binaries using short Fourier transforms". In: *Phys. Rev. D* 111.10 (2025), p. 104044. DOI: 10.1103/PhysRevD.111.104044. arXiv: 2502.11823 [gr-qc].

References III

[13] Kaze W. K. Wong, Maximiliano Isi, and Thomas D. P. Edwards. "Fast Gravitational-wave Parameter Estimation without Compromises". In: *Astrophys. J.* 958.2 (2023), p. 129. DOI: 10.3847/1538-4357/acf5cd. arXiv: 2302.05333 [astro-ph.IM].

[14] Thibeau Wouters et al. "Robust parameter estimation within minutes on gravitational wave signals from binary neutron star inspirals". In: *Phys. Rev. D* 110.8 (2024), p. 083033. DOI: 10.1103/PhysRevD.110.083033. arXiv: 2404.11397 [astro-ph.IM].

[15] Justin Janquart et al. "Analyses of overlapping gravitational wave signals using hierarchical subtraction and joint parameter estimation". In: *Mon. Not. Roy. Astron. Soc.* 523.2 (2023), pp. 1699–1710. DOI: 10.1093/mnras/stad1542. arXiv: 2211.01304 [gr-qc].

[16] Geoffrey Ryan et al. "Gamma-Ray Burst Afterglows in the Multimessenger Era: Numerical Models and Closure Relations". In: *Astrophys. J.* 896.2 (2020), p. 166. DOI: 10.3847/1538-4357/ab93cf. arXiv: 1909.11691 [astro-ph.HE].

[17] Hauke Koehn et al. "Efficient Bayesian analysis of kilonovae and GRB afterglows with fiesta". In: (July 2025). arXiv: 2507.13807 [astro-ph.HE].

[18] Thibeau Wouters et al. "Leveraging differentiable programming in the inverse problem of neutron stars". In: (Apr. 2025). arXiv: 2504.15893 [astro-ph.HE].

References IV

[19] Kurzgesagt. *Figures taken from "Neutron Stars - The Most Extreme Things that are not Black Holes"*. Accessed on May 14, 2025. 2019. URL: https://www.youtube.com/watch?v=udFxKZRyQt4.

[20] Hergé. *Cover figure created with ChatGPT using this input figure from the comic Destination Moon*. Accessed on May 14, 2025. 2019. URL: https://www.youtube.com/watch?v=udFxKZRyQt4.

[21] Jason D. McEwen et al. *Machine learning assisted Bayesian model comparison: learnt harmonic mean estimator*. 2023. arXiv: 2111.12720 [stat.ME]. URL: https://arxiv.org/abs/2111.12720.

[22] Alicja Polanska et al. *Learned harmonic mean estimation of the marginal likelihood with normalizing flows*. 2024. arXiv: 2307.00048 [stat.ME]. URL: https://arxiv.org/abs/2307.00048.

[23] Alicja Polanska et al. "Accelerated Bayesian parameter estimation and model selection for gravitational waves with normalizing flows". In: *38th conference on Neural Information Processing Systems*. Oct. 2024. arXiv: 2410.21076 [astro-ph.IM].

Evidence calculation: HARMONIC I

Evidence Z can be computed from posterior samples with HARMONIC [21] with the harmonic mean estimator

$$\begin{aligned}\rho &\equiv \mathbb{E}_{P(\theta|d)}\left[\frac{1}{L(\theta)}\right] \\ &= \int \mathrm{d}\theta \frac{1}{\mathcal{L}(\theta)} P(\theta|d) \\ &= \int \mathrm{d}\theta \frac{1}{\mathcal{L}(\theta)} \frac{\mathcal{L}(\theta)\pi(\theta)}{Z} = \frac{1}{Z}\end{aligned}$$

Therefore, estimate ρ with posterior samples:

$$\hat{\rho} = \frac{1}{N}\sum_{i=1}^{N} \frac{1}{\mathcal{L}(\theta)}, \quad \theta_i \sim P(\theta|d)$$

Evidence calculation: HARMONIC II

Can be interpreted as importance sampling

$$\rho = \int \mathrm{d}\theta \frac{1}{Z} \frac{\pi(\theta)}{P(\theta|d)} P(\theta|d)\,,$$

but with target = prior and sampling density = posterior. Therefore, importance sampling is inefficient – how to solve?
New proposal:

$$\begin{aligned}\rho &= \mathbb{E}_{P(\theta|d)}\left[\frac{\varphi(\theta)}{\mathcal{L}(\theta)\pi(\theta)}\right] \\ &= \int \mathrm{d}\theta\, \frac{\varphi(\theta)}{\mathcal{L}(\theta)\pi(\theta)}\, P(\theta|d) \\ &= \int \mathrm{d}\theta\, \frac{\varphi(\theta)}{\mathcal{L}(\theta)\pi(\theta)}\, \frac{\mathcal{L}(\theta)\pi(\theta)}{Z} = \frac{1}{Z}\end{aligned}$$

Evidence calculation: HARMONIC III

Use the following estimator:

$$\hat{\rho} = \frac{1}{N} \sum_{i=1}^{N} \frac{\varphi(\theta_i)}{\mathcal{L}(\theta_i)\pi(\theta_i)}, \quad \theta_i \sim P(\theta|d)$$

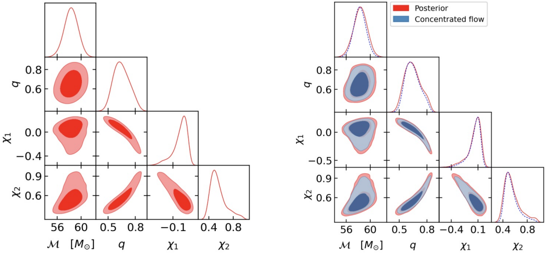
Replace the target distribution π with φ: only requirement is that it is normalized

In practice, this can be achieved with a normalizing flow [22].

This has been verified to give accurate evidences (similar values as nested sampling) when GW posteriors are used [23].

Table 1: Total wall times to compute the evidence estimates for the examples discussed in the main text. We run BILBY on 16 CPU cores and JIM + `harmonic` on 1 GPU.

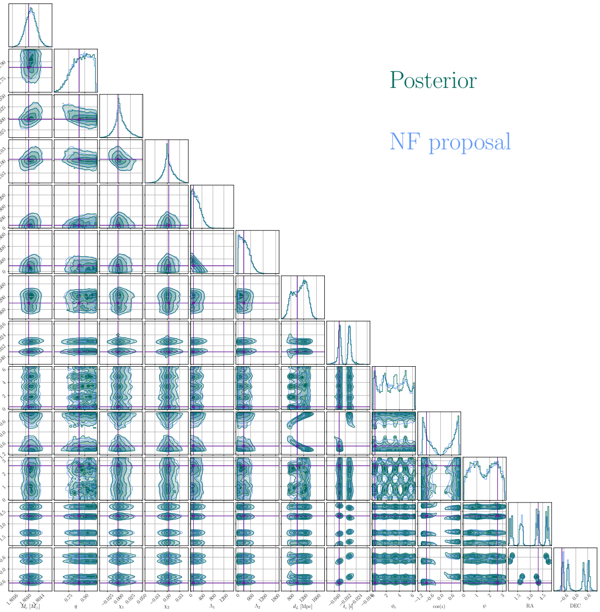
Example	Method	$\log(z)$	Sampling time	Evidence estimation time
4D	BILBY	390.33 ± 0.11	31.3 min	–
	JIM + `harmonic`	$390.360^{+0.006}_{-0.006}$	3.4 min	1.9 min
11D	BILBY	378.29 ± 0.15	3.5 h	–
	JIM + `harmonic`	$378.420^{+0.09}_{-0.08}$	11.8 min	2.4 min



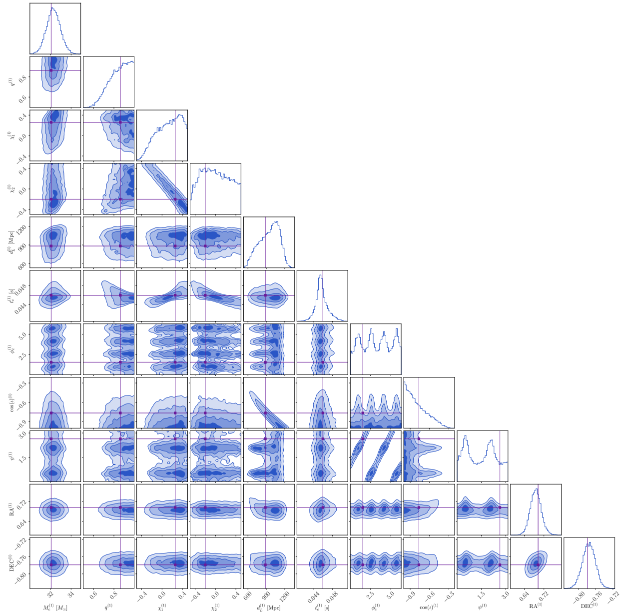
(a) BILBY (b) JIM + `harmonic`

Figure 1: Corner plots for the 4-dimensional posterior samples from (a) BILBY and (b) JIM used for inference (solid red) alongside the concentrated flow at $T = 0.8$ used in the learned harmonic mean (dashed blue).

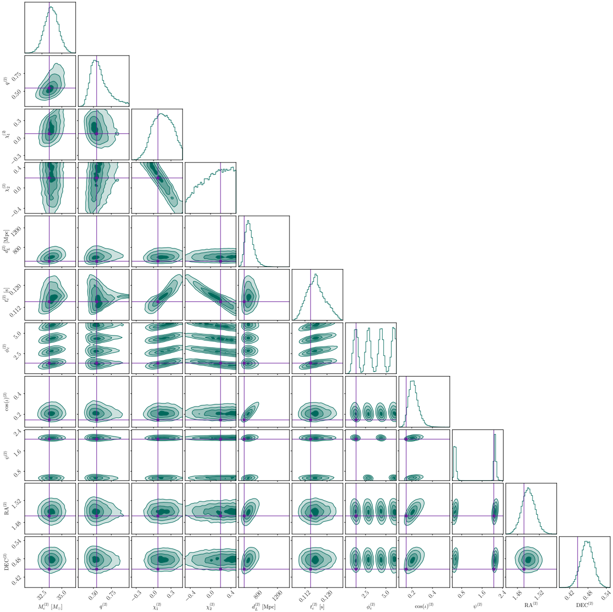
BNS in ET-Δ example: all parameters



Overlapping signals: all parameters signal A

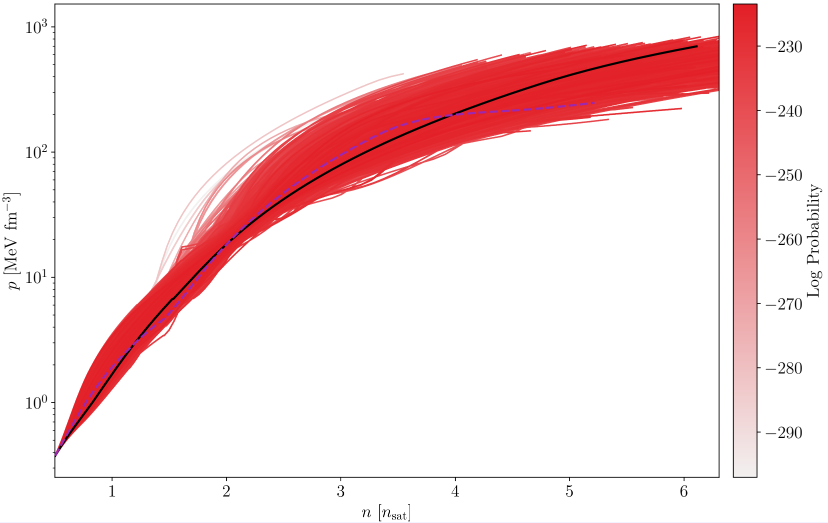


Overlapping signals: all parameters signal B



Equation of state O5 projection with 20 BNS: EOS

- **Purple**: target
- **Red**: posterior EOS samples (**black**: maximum log posterior)



Equation of state O5 projection with 20 BNS: NS