DeepHMC

Deep Neural Network based Hamiltonian Monte Carlo algorithm for binary neutron star parameter estimation

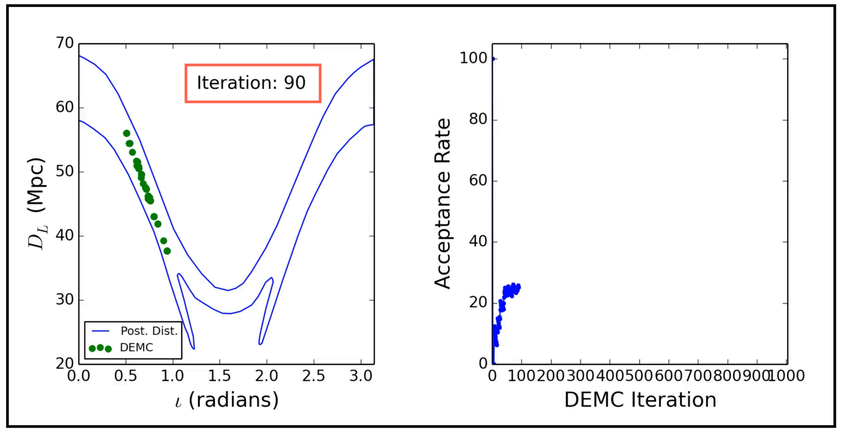
GDR Ondes gravitationelles
LUTH, Observatoire de Paris
October 2023

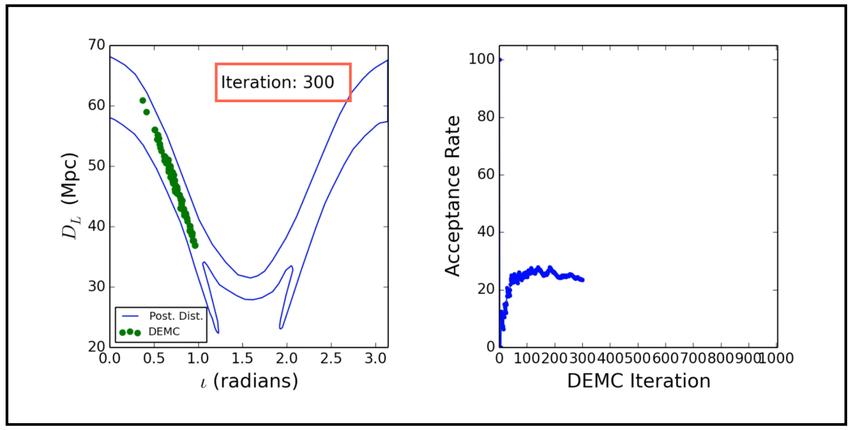
Jules Perret & Ed. Porter
APC /CNRS

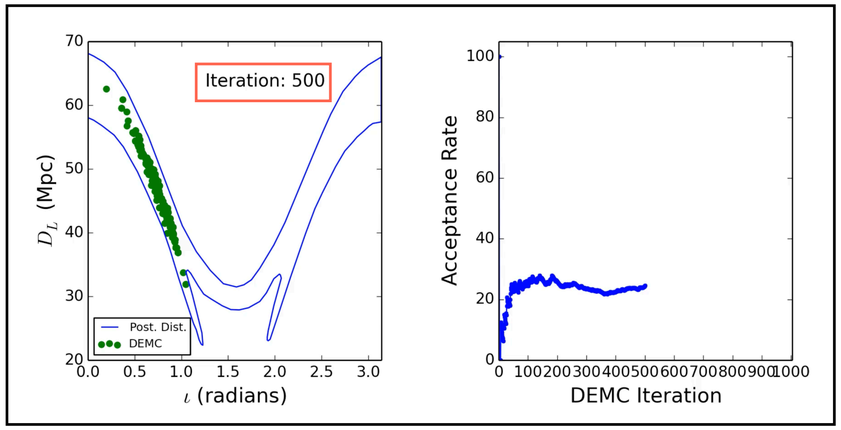
Bayesian inference algorithms used in the LVK

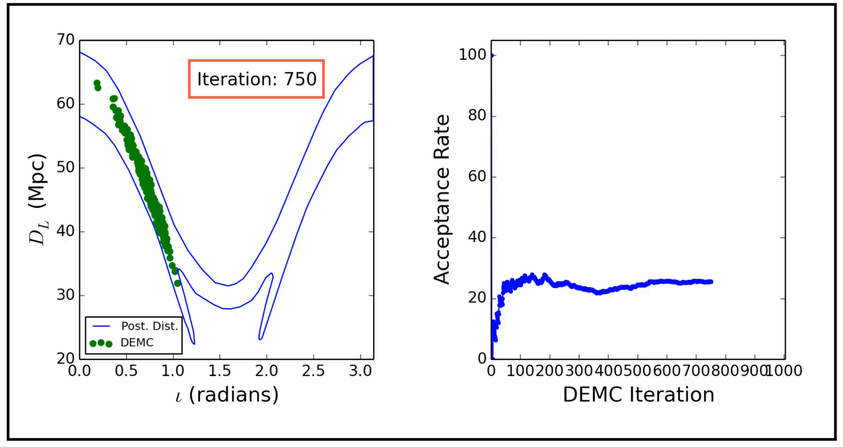
- **Common Bayesian inference samplers (e.g. MCMC, nested sampling) belong to a family of random walk samplers**
  - **Pros:**
    - Relatively easy to construct and tune
  - **Cons**
    - require specifically designed proposal distributions to improve efficiency
    - known to have slow convergence properties, especially as *D* increases
    - difficult to find new accepted points just by random jumping
    - optimal acceptance rate: 23%
    - GW parameter estimation can take weeks to months to run.

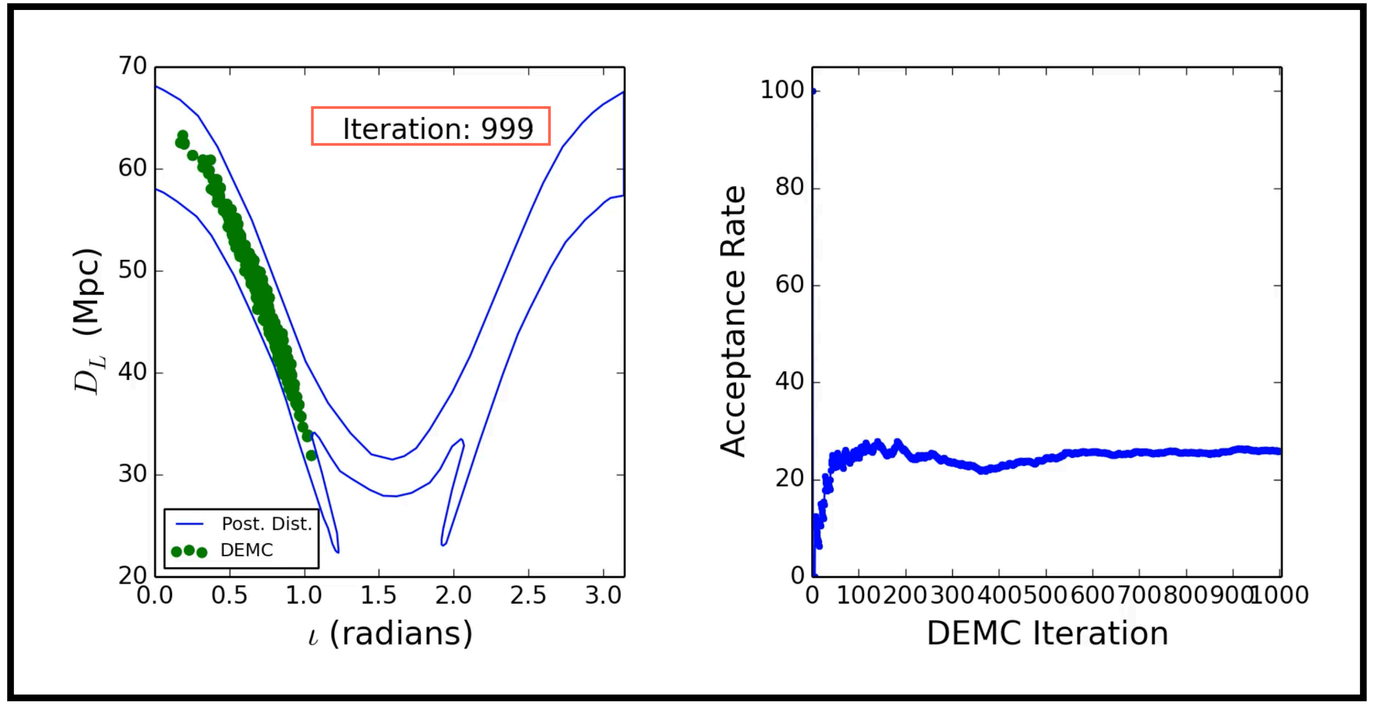
MCMC performance example











Hamiltonian Monte Carlo

- **The HMC is a non-random walk sampler**
  - **Pros:**
    - simulates Hamiltonian dynamics in an inverted log-likelihood potential
    - uses background geometry to guide exploration of the parameter space
    - Hamiltonian trajectories now become the proposal distribution.
    - Optimal acceptance rate: 65%
    - known to converge *D* times faster than MCMC
  - **Cons**
    - difficult to set up and tune
    - log-likelihood gradients are a computational bottleneck

Bayes' theorem and the HMC

$$P(\lambda^\mu \mid s) = \frac{\mathscr{L}(\lambda^\mu)\pi(\lambda^\mu)}{p(s)}$$

$P(s \mid \lambda^\mu) = \mathscr{L}(\lambda^\mu)$: likelihood

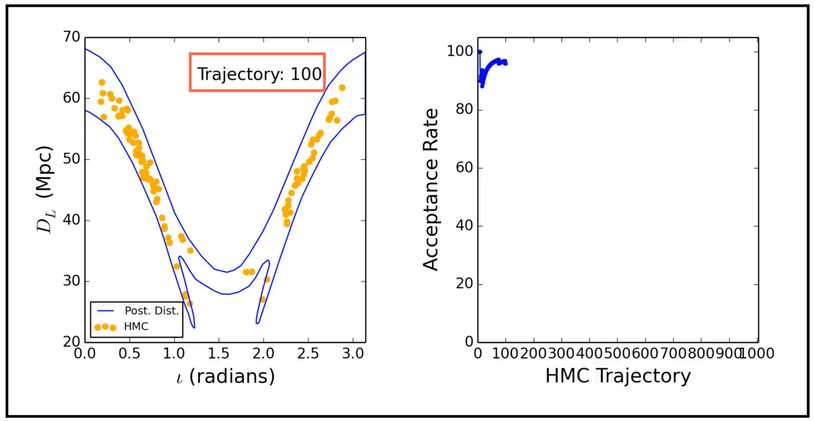
$P(\lambda^\mu \mid s)$: Posterior

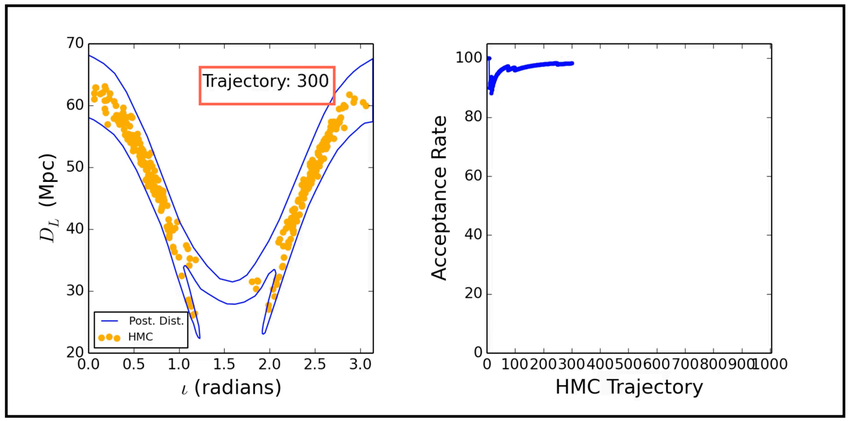
$\pi(\lambda^\mu)$: likelihood

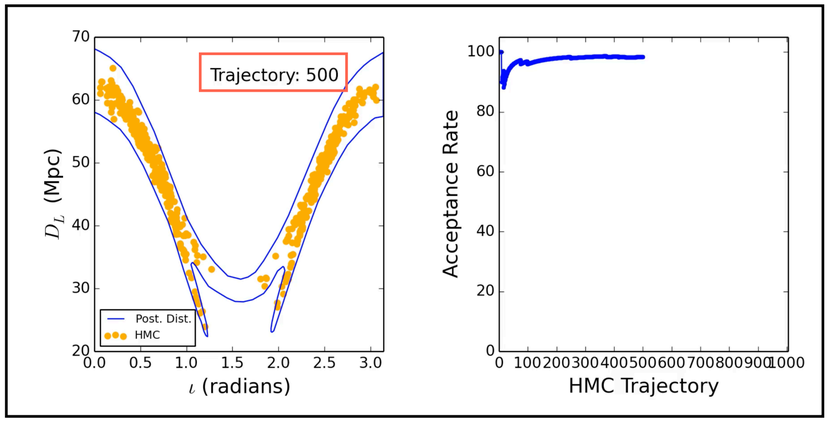
$p(s)$: evidence

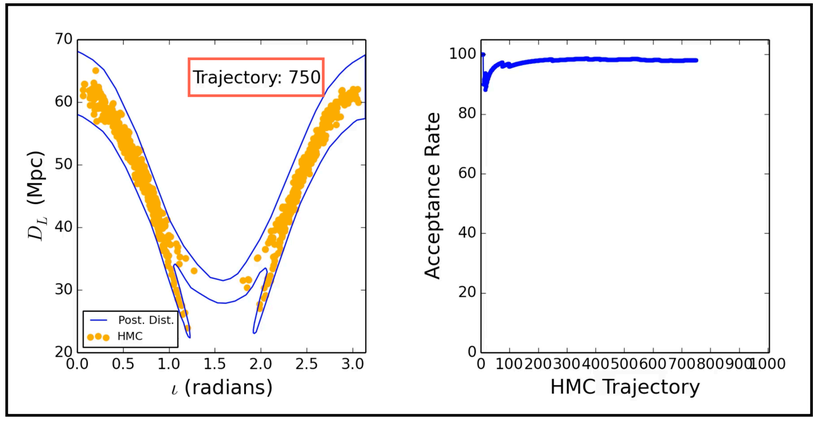
- Equate GW template parameters to state space variables, i.e. $q^\mu = \lambda^\mu$
- Construct a "gravitational" potential energy: $\mathscr{U}(q^\mu) = -\ln[\pi(q^\mu)\mathscr{L}(q^\mu)]$
- Define canonical momenta and Kinetic energy: $\mathscr{K}(p^\mu) = \frac{1}{2}M^{-1}_{\mu\nu}p^\mu p^\nu$
- and a Hamiltonian: $\mathscr{H}(q^\mu, p^\mu) = \mathscr{U}(q^\mu) + \mathscr{K}(p^\mu)$
- This Hamiltonian has a canonical distribution over phase space, and hence we can design the algorithm such that $\Pi(p^\mu) \sim \mathscr{N}(0,1)$

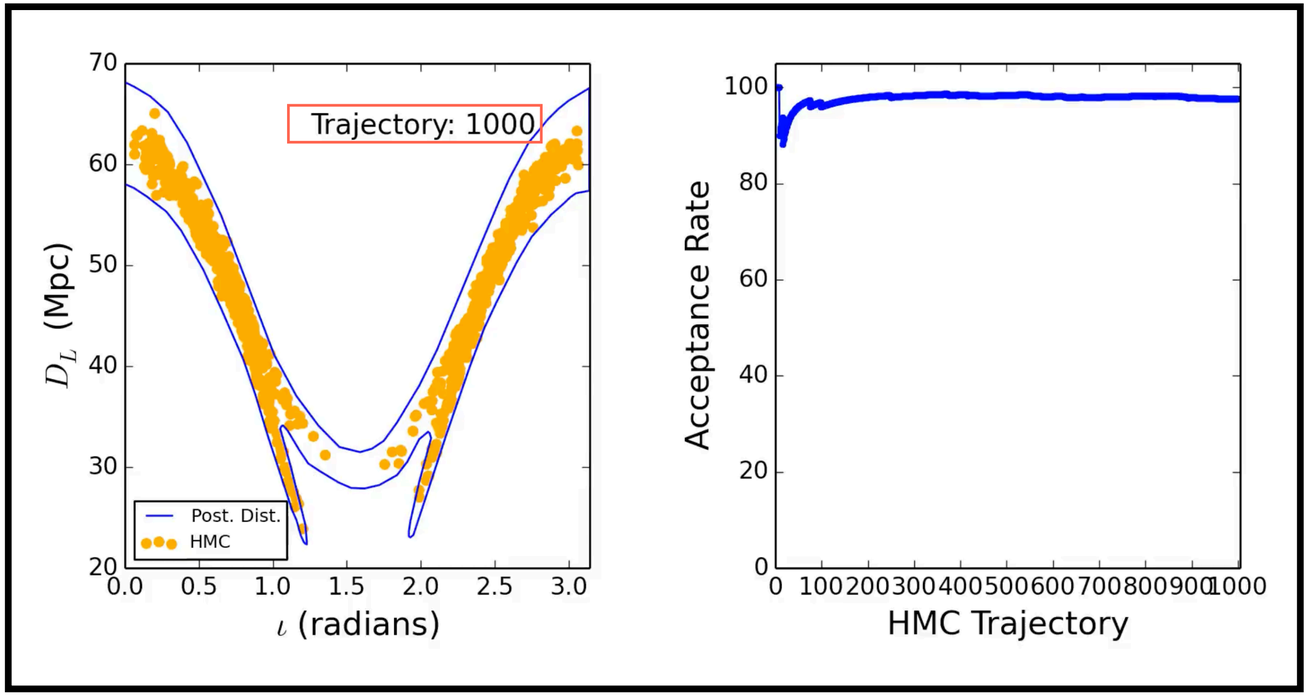
HMC performance example





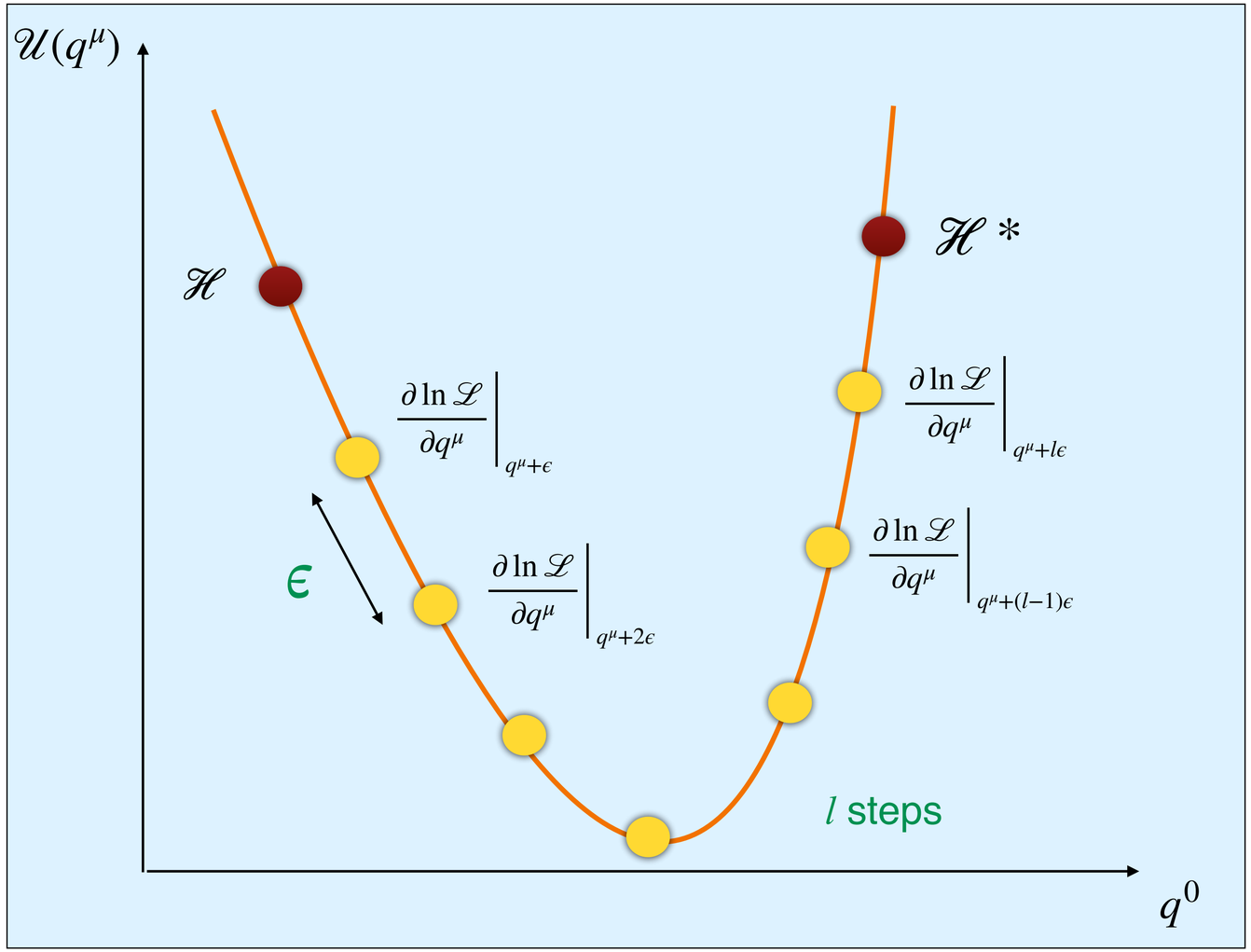






Hamiltonian Monte-Carlo

Structure of a Hamiltonian trajectory



Some numbers:

- **12-D parameter** space (marginalising over phase at coalscence)
- Use **IMRPhenomD_NRTidal** on 128secs of GW170817 data (GWOSC)
- Each 12-*D* numerical gradient takes ~ 2 seconds to generate (bottleneck)
- Each gradient requires 13 waveform generations.
- ~ 7 mins for $l = 200$ step trajectory (2600 waveforms)
- ~ 6.3 years for 500 000- trajectory run (1.3 billion waveforms)
- **Goal : Eliminate the need for waveform generations by building a model to approximate the log-likelihood gradients**

Deep Hamiltonian Monte Carlo

Phase I:

- Information gathering phase: run *N* trajectories recording all positions and gradients.
- Use numerical gradients based on numerical differencing of waveforms.

Phase II:

- Use recorded data from phase I to build a precise model of the gradients of the log-likelihood using a Deep Neural Network.

Phase III:

- Bayesian inference.
- Run Trajectories with the gradient model. No waveform generation needed.
- ≃3000 times faster than phase I trajectories

Comparaison with public GW170817 results

Parameter	LALInferenceMCMC	DeepHMC	JS
θ_{JN}/rad	$2.5^{+0.4}_{-0.5}$	$2.5^{+0.4}_{-0.5}$	0.00123
ψ/rad	$1.5^{+1.5}_{-1.4}$	$1.6^{+1.4}_{-1.5}$	0.00077
D_L/Mpc	38^{+8}_{-17}	38^{+8}_{-16}	0.00107
z	$0.008^{+0.002}_{-0.004}$	$0.008^{+0.002}_{-0.004}$	0.00063
$m_1/\mathrm{M}_\odot$	$1.47^{+0.16}_{-0.10}$	$1.49^{+0.14}_{-0.10}$	0.01516
$m_2\,\mathrm{M}_\odot$	$1.27^{+0.09}_{-0.12}$	$1.25^{+0.09}_{-0.10}$	0.01421
$\mathcal{M}/\mathrm{M}_\odot$	$1.188^{+0.004}_{-0.002}$	$1.188^{+0.004}_{-0.002}$	0.00106
q	$0.86^{+0.13}_{-0.16}$	$0.84^{+0.12}_{-0.14}$	0.01454
δ/rad	$-0.37^{+0.07}_{-0.07}$	$-0.36^{+0.06}_{-0.06}$	0.01910
α/rad	$3.43^{+0.04}_{-0.04}$	$3.42^{+0.04}_{-0.03}$	0.00719
t_c/s	$1187008882.4312^{+0.0012}_{-0.0010}$	$1187008882.4312^{+0.0010}_{-0.0009}$	0.00282
χ_1	$0.002^{+0.027}_{-0.018}$	$0.003^{+0.026}_{-0.017}$	0.00098
χ_2	$0.001^{+0.027}_{-0.020}$	$0.002^{+0.028}_{-0.019}$	0.00217
χ_{eff}	$0.002^{+0.015}_{-0.009}$	$0.004^{+0.015}_{-0.009}$	0.00499
Λ_1	247^{+730}_{-226}	251^{+682}_{-228}	0.00093
Λ_2	401^{+1104}_{-363}	444^{+1085}_{-398}	0.00119
$\tilde{\Lambda}$	370^{+484}_{-247}	397^{+438}_{-262}	0.00328
$\delta\tilde{\Lambda}$	3^{+206}_{-191}	4^{+203}_{-193}	0.00087
ρ_{mf}	$33.26^{+0.09}_{-0.13}$	$32.86^{+0.08}_{-0.13}$	0.68336
$\ln(\mathcal{L}_R)$	548^{+3}_{-4}	540^{+3}_{-4}	0.61110

Runtime comparaison:

Using $f_{low} = 23Hz$ and $f_{sampling} = 4096Hz$

Phase I: 8.8 days for 1500 trajectories giving 279600 points

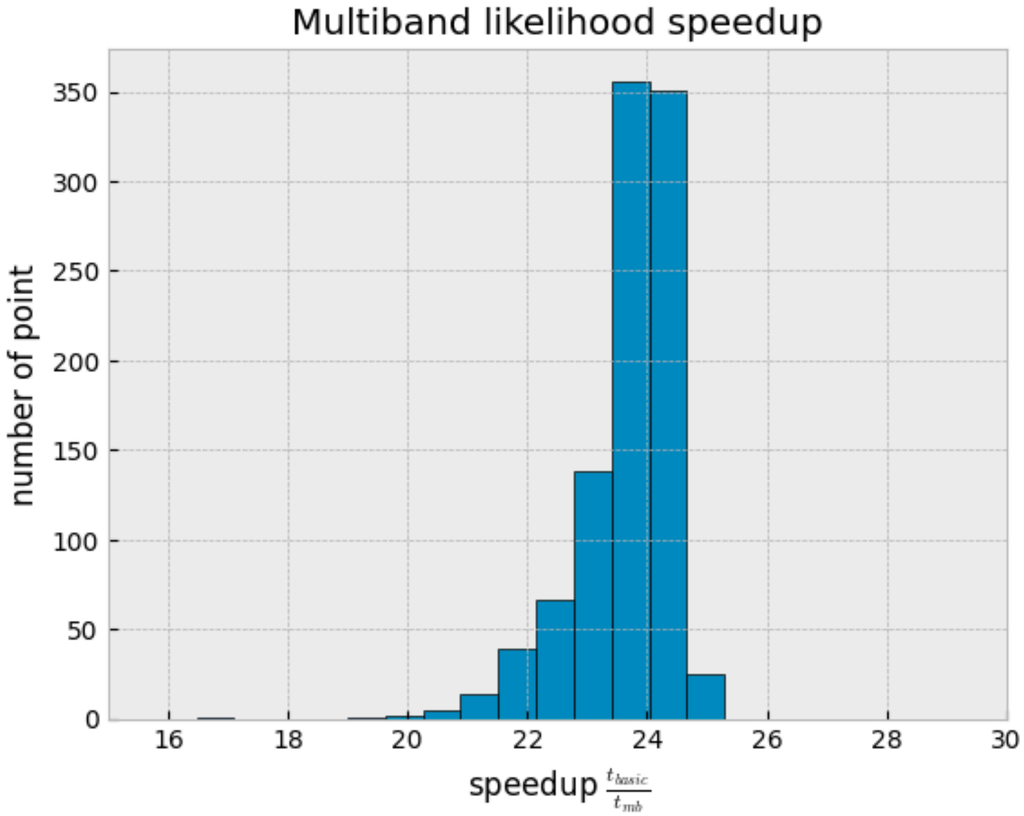
Phase II: 4.6 hours (15 minutes to fit DNN, 266 mins to optimise ϵ)

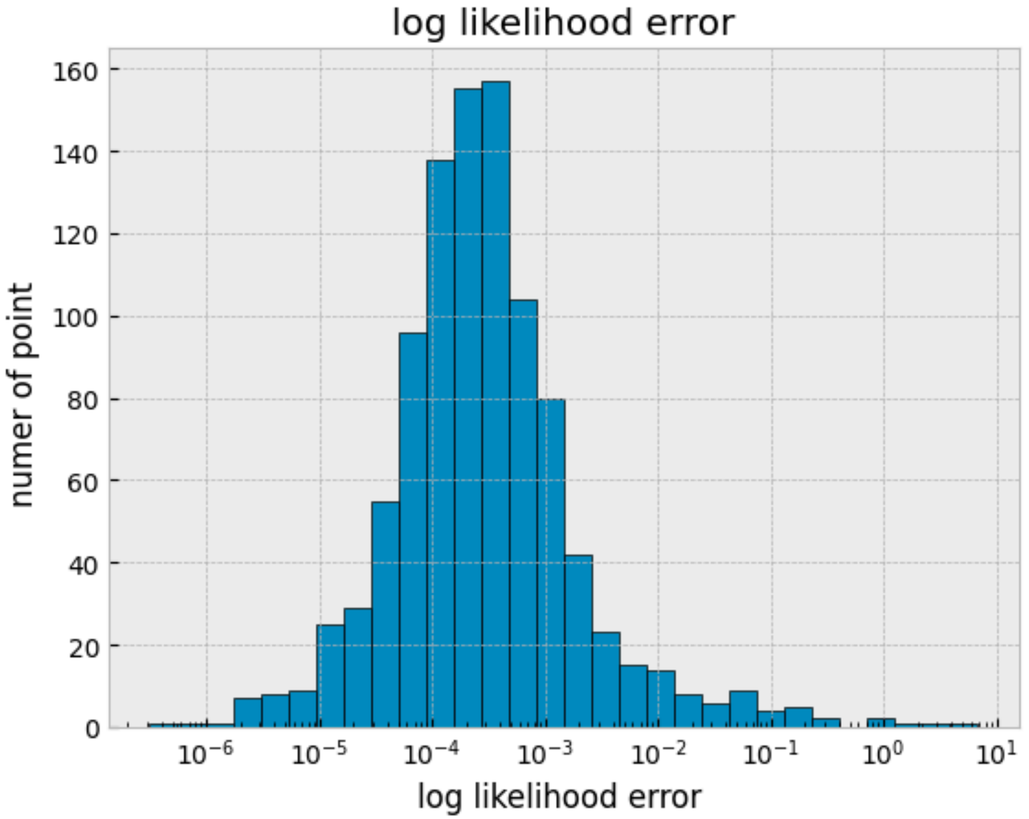
Phase III: 4.4 days for 140 000 trajectories giving 5540 SISs at a cost of **68.9 s/SIS**

On laptop, LALInferenceMCMC has a cost of **approximalty 2000 s/SIS**

DeepHMC: Phase I optimisation

Multiband likelihood: *S. Morisaki 2021 arXiv: 2104.07813*





- Theoretical speed up : ~**x17**
- Speedup measured: ~x**24**

DeepHMC: Phase I optimisation

Optimisation of the likelihood gradients

- version 1: used numerical differencing of waveforms to construct likelihood gradients

$$\partial_\mu \ln \mathcal{L} = \langle s|\partial_\mu h\rangle - \langle h|\partial_\mu h\rangle$$

where

$$\partial_\mu h = \frac{h(\lambda^\mu + \Delta\lambda^\mu) - h(\lambda^\mu - \Delta\lambda^\mu)}{2\Delta\lambda^\mu}$$

  - numerically unstable due to using differences of very small numbers, i.e. $h \sim 10^{-21}$

- version 2: uses direct differencing of log-likelihoods

$$\partial_\mu \ln \mathcal{L} = \frac{\ln \mathcal{L}(\lambda^\mu + \Delta\lambda^\mu) - \ln \mathcal{L}(\lambda^\mu - \Delta\lambda^\mu)}{2\Delta\lambda^\mu}$$

  - numerically stable

DeepHMC: Phase I runtime

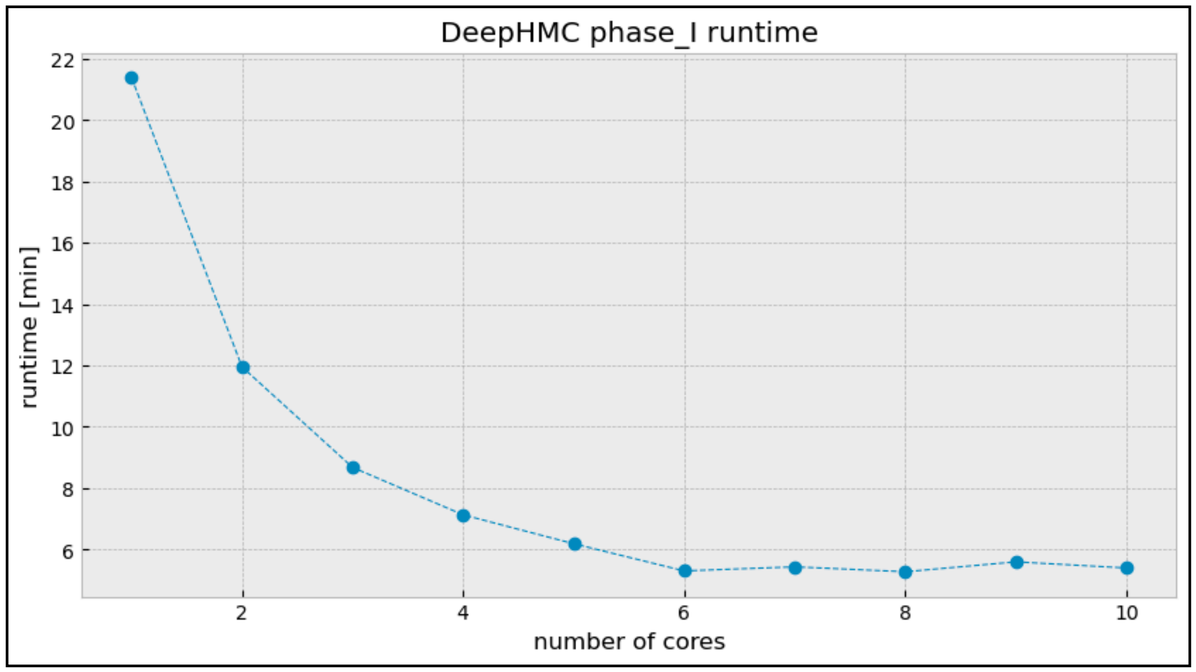
- **Running on a single core**

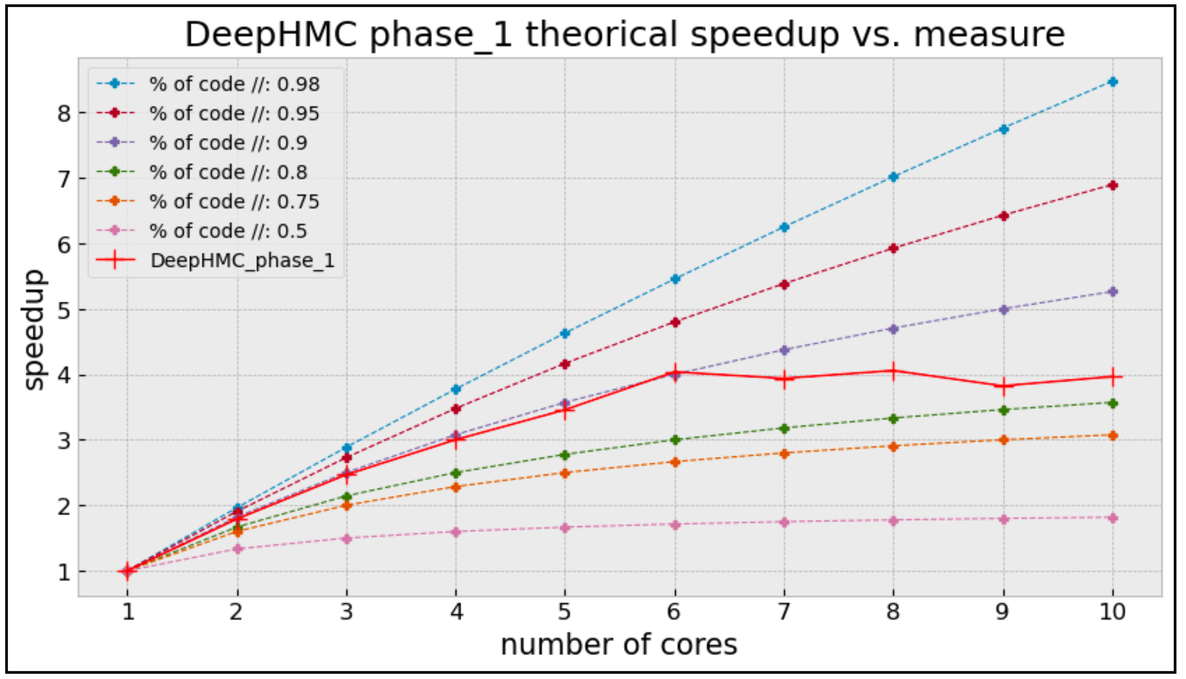
	single trajectory ($l = 200$)	1500 trajectories
DeepHMC.v1	~8.5 minutes	~10 days
DeepHMC.v2	~15 seconds	~5 hours

Parallelisation of Phase I

First measure with a multiprocess version:

	Sequential version (1 core)	Parallel version (10 cores)
Runtime for a run of 100 trajectories	< 22 min	< 6 min





Test realised on personal computer : MacBook Pro 2023 - Apple M2 Pro , 16GB RAM

DeepHMC: future work

1) Full BNS inference in < 1 day

- Phase I:
  - Increase the coverage of the parameter space by increasing *N* ✅
- Phase II:
  - Tune and optimise the Deep Neural Network architecture.
- Phase III:
  - Apply multi-band likelihood

2) Apply to BBH, SSM, NSBH systems

3) Test DeepHMC on ET data mock challenge

Conclusion

- HMC has a superior convergence compared to standard MCMC
- Reduced the log-likelihood gradient bottleneck by using a DNN gradient model.
- Results verified against GW170817.
- DeepHMC.v2
  - Have implemented multiband likelihoods
  - Have optimised the log-likelihood gradient calculations in Phase I
  - Have reduced Phase I runtime to ~5 hours on a single core (~1h on 6 cores)
- Plan to reduce runtime of parameter estimation for BNS to a fraction of a day.