BAYESIAN INFERENCE IN GRAVITATIONAL-WAVE ASTRONOMY

PyCBC Inference online workshop 2020

Collin Capano

Max Planck Institute for Gravitational Physics

(Albert Einstein Institute)

Hannover, Germany

OUTLINE

I. OVERVIEW

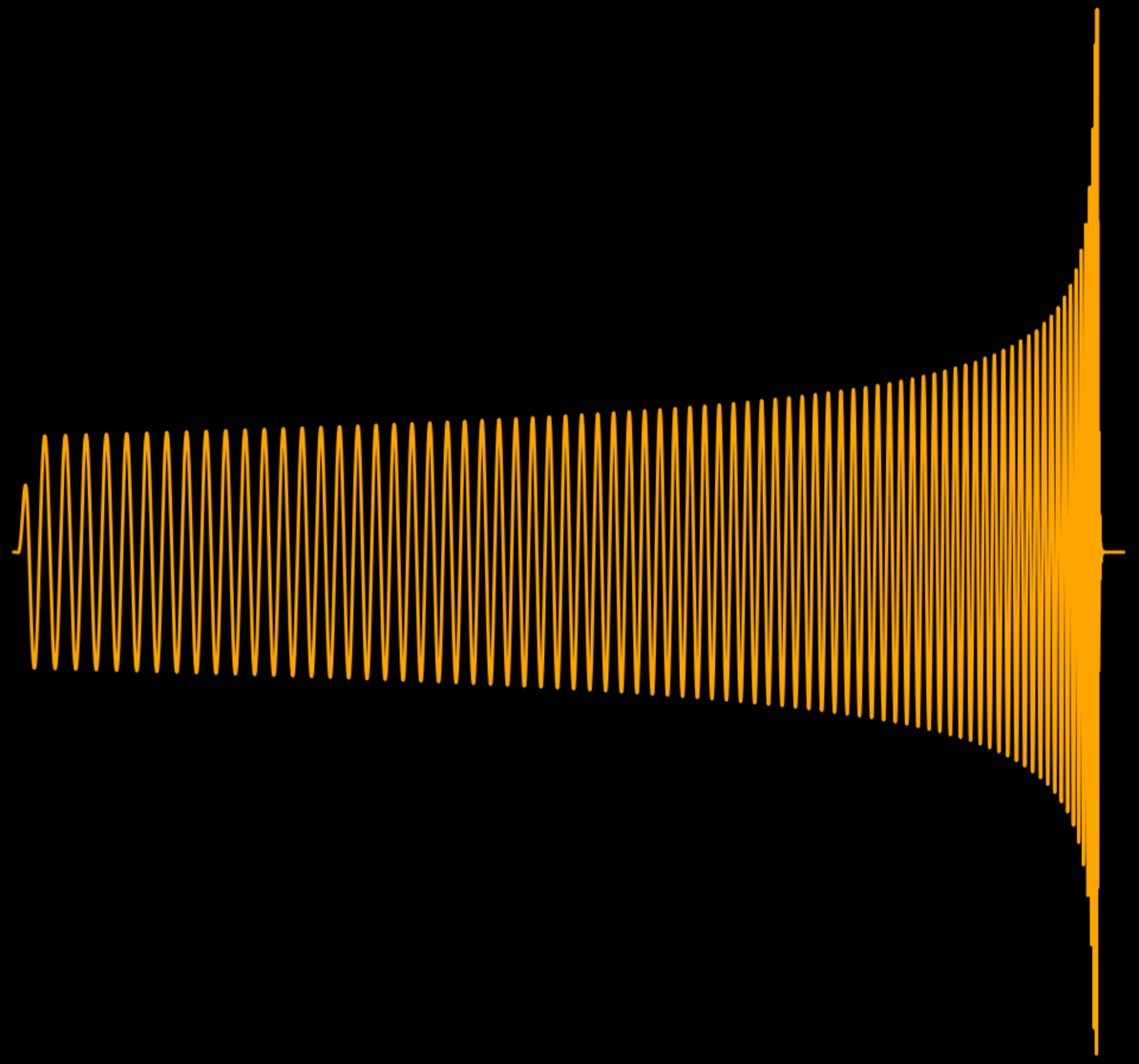
II. COMPACT BINARY PARAMETERS

III. BAYESIAN INFERENCE AND ITS APPLICATION TO GRAVITATIONAL WAVES

IV. MODELS AND SAMPLERS IN PYCBC INFERENCE

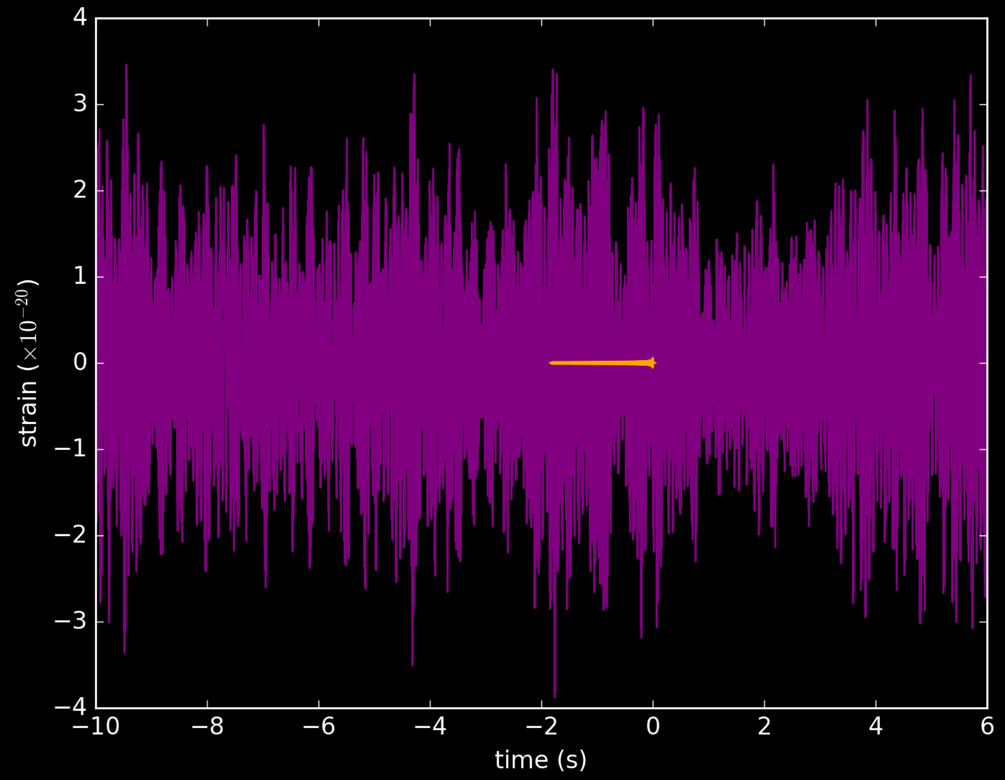
STATEMENT OF PROBLEM

- To do gravitational wave astronomy we need to detect signals in noise (search) and determine the parameters of their source (parameter estimation). But…
- Gravitational waves are weak!
- Even “loud” signals detected by Advanced LIGO & Virgo are buried in noise.
- Need tools that produce probabilistic statements about signals’ existence and parameters.



STATEMENT OF PROBLEM

- To do gravitational wave astronomy we need to detect signals in noise (search) and determine the parameters of their source (parameter estimation). But…
- Gravitational waves are weak!
- Even “loud” signals detected by Advanced LIGO & Virgo are buried in noise.
- Need tools that produce probabilistic statements about signals’ existence and parameters.



SEARCHES VS PARAMETER ESTIMATION

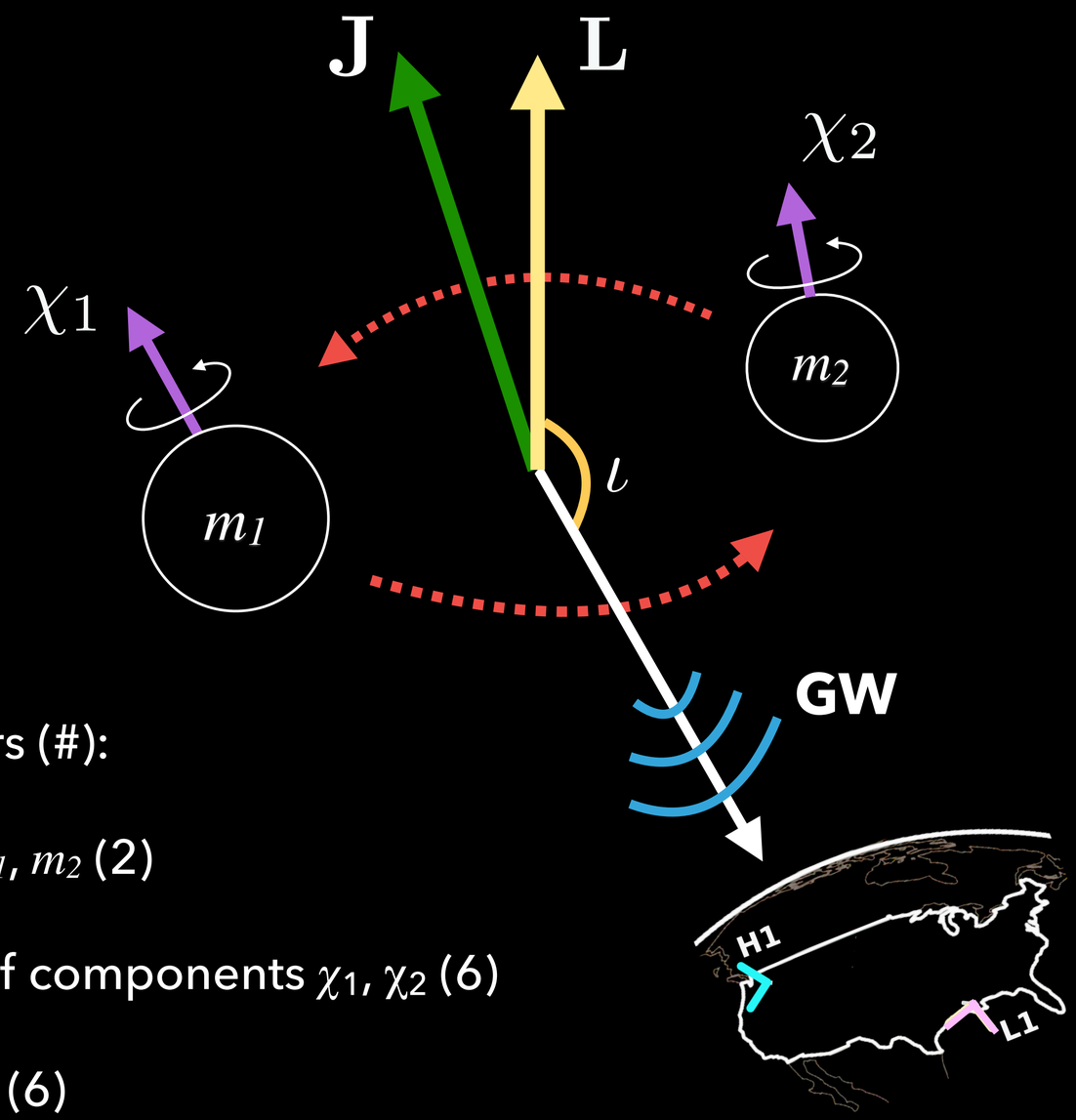
- **Searches**: find times when a measureable gravitational-wave signal (“event”) exists in the detectors & measure statistical significance of candidate events
  - Use match filtering, discrete “template banks”
  - Trade parameter accuracy for computational efficiency
- **Parameter estimation**: followup candidate events provided by searches
  - Assume signal exists in the data, use Bayesian inference to accurately measure source parameters
  - Trade computational efficiency for parameter accuracy
  - Also used for model selection, testing alternative theories

PYCBC

- Python-based open source software suite for gravitational-wave astronomy
- pycbc.org
- Inititally developed to search for gravitational-waves from compact binary mergers
- Has evolved to include parameter estimation
- **PyCBC Inference**: module in PyCBC that does Bayesian inference on gravitational-waves using external stochastic sampling packages

COMPACT BINARY PARAMETERS

BINARY BLACK HOLE PARAMETERS



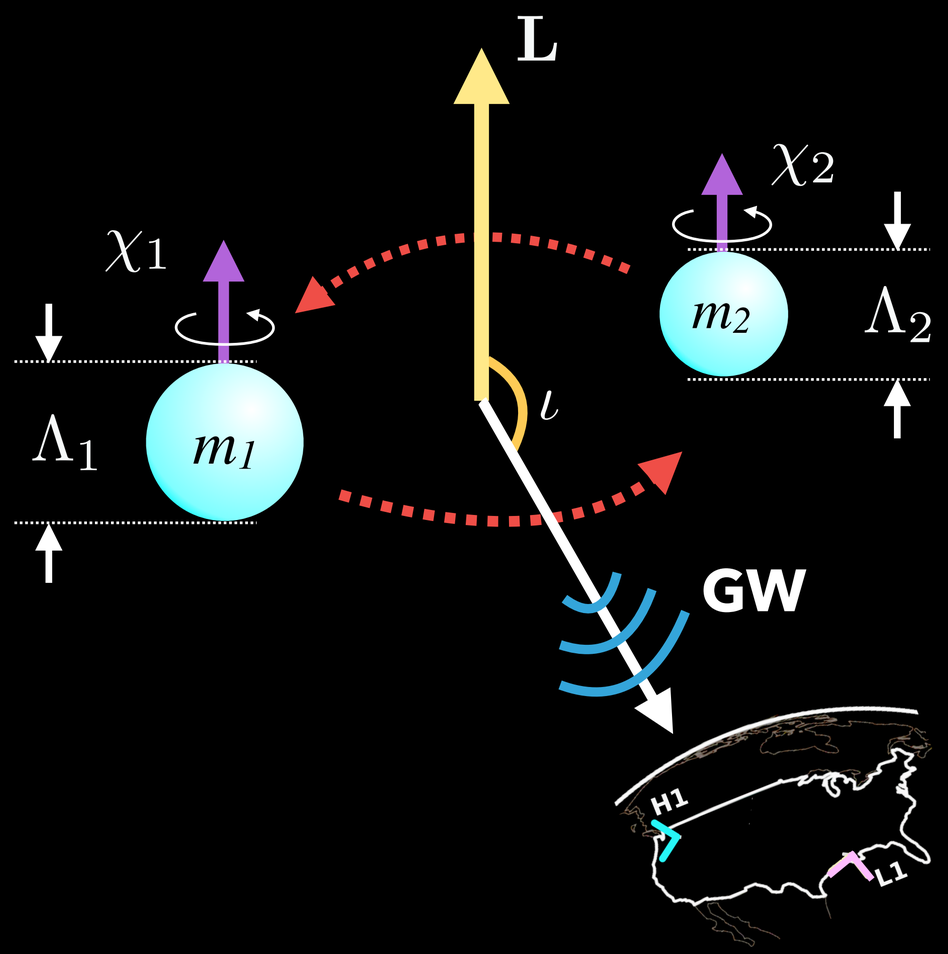
- Possible BBH parameters (#):
  - component masses m_1, m_2 (2)
  - dimensionless spins of components χ_1, χ_2 (6)
  - location & orientation (6)

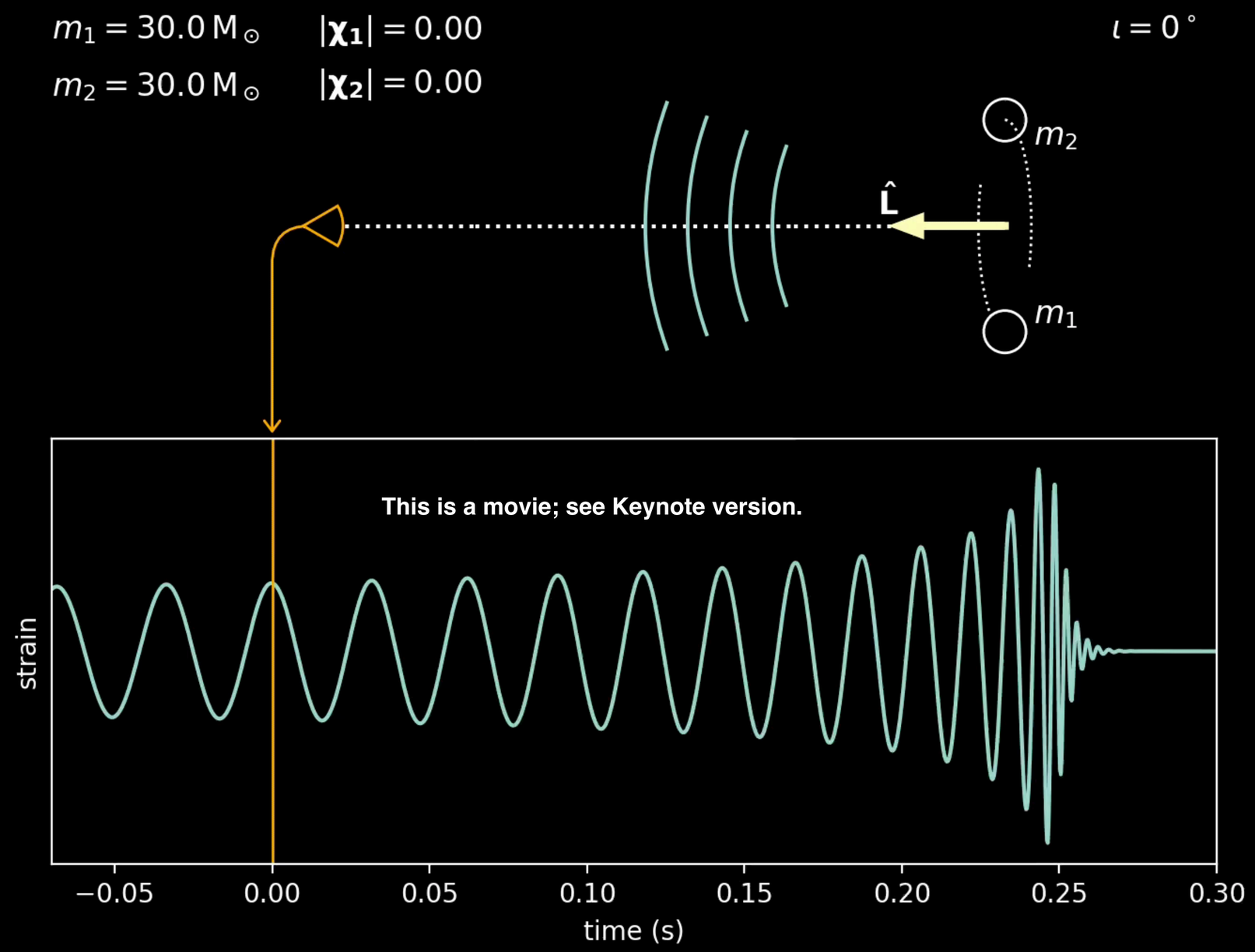
BINARY NEUTRON STAR PARAMETERS

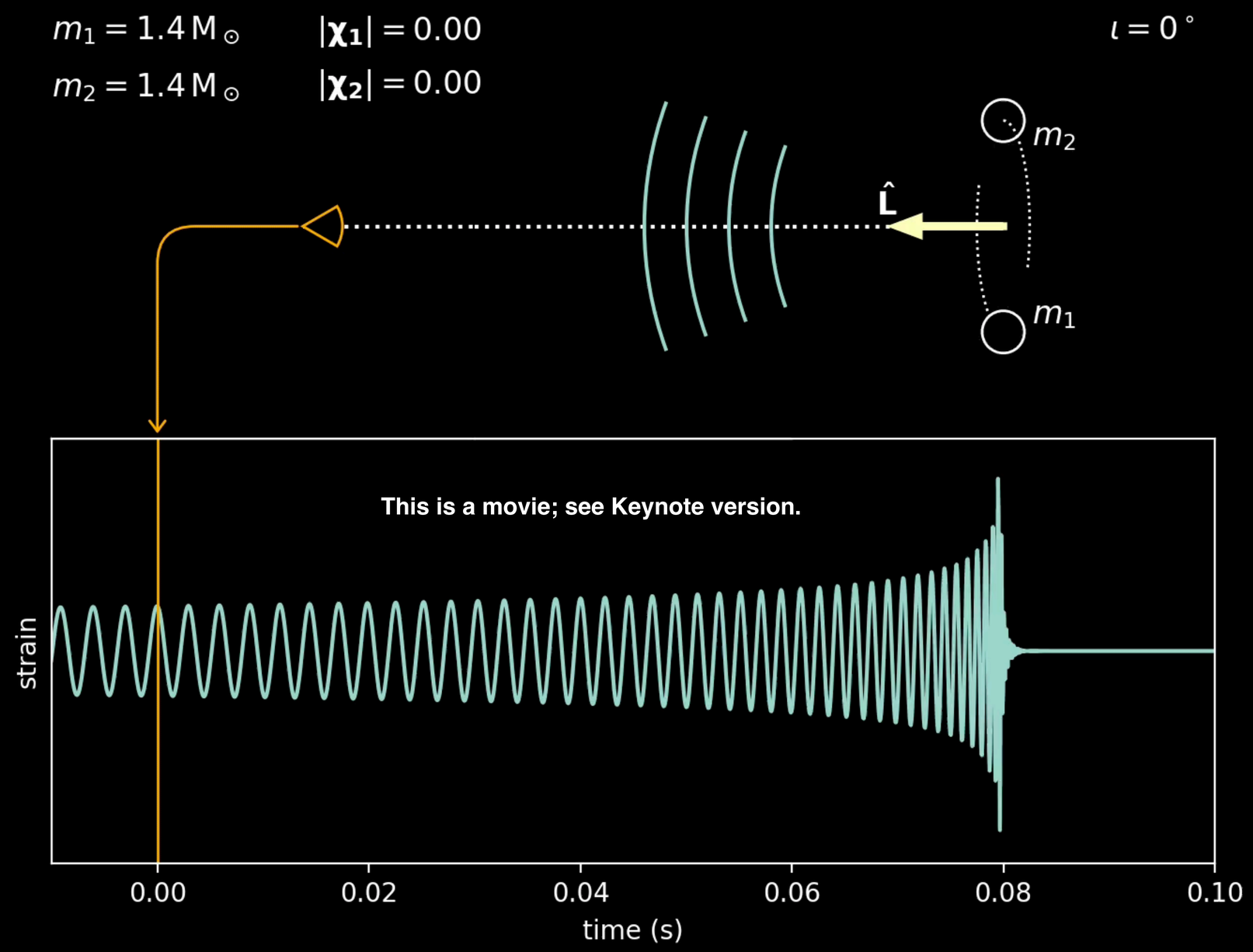
- Tidal forces cause neutron stars to deform

$$\Lambda_i \equiv \frac{2}{3} k_2 \left(\frac{R_i c^2}{G m_i} \right)^5$$

- Tidal deformability depends on the equation of state of the matter in the core
- Stars' deformability is encoded in gravitational wave




$m_1 = 30.0\,\mathrm{M}_\odot$
$m_2 = 30.0\,\mathrm{M}_\odot$
$|\boldsymbol{\chi_1}| = 0.00$
$|\boldsymbol{\chi_2}| = 0.00$
$\iota = 0^\circ$
m_2
$\hat{\mathbf{L}}$
m_1
This is a movie; see Keynote version.
strain
−0.05
0.00
0.05
0.10
0.15
0.20
0.25
0.30
time (s)


$m_1 = 1.4\,M_\odot$
$m_2 = 1.4\,M_\odot$
$|\chi_1| = 0.00$
$|\chi_2| = 0.00$
$\iota = 0^\circ$
m_2
$\hat{L}$
m_1
This is a movie; see Keynote version.
strain
0.00
0.02
0.04
0.06
0.08
0.10
time (s)

BAYESIAN INFERENCE PRIMER

BAYES' THEOREM

- GW's source parameters are estimated using Bayesian inference.
- Assume a signal h exists in some data d.
- Probability that the signal has parameters $\vartheta = \{m_1, m_2, \ldots\}$ is:

$$p(\vec{\vartheta}|d,h) = \frac{p(d|\vec{\vartheta},h)\,p(\vec{\vartheta}|h)}{p(d|h)}$$

- **Prior**: Represents our state of knowledge about the true parameters *before* measuring the data
- **Likelihood**: The probability of observing the data assuming that a particular set of parameters is true.
- **Posterior**: Represents our state of knowledge about the parameters *after* measuring the data.

Recommend: D.S. Sivia & J. Skilling, "Data Analysis: A Bayesian Tutorial"

BAYES' THEOREM

- GW's source parameters are estimated using Bayesian inference.
- Assume a signal h exists in some data d.
- Probability that the signal has parameters $\vartheta = \{m_1, m_2, \ldots\}$ is:

$$p(\vec{\vartheta}|d,h) = \frac{p(d|\vec{\vartheta},h)\,p(\vec{\vartheta}|h)}{p(d|h)}$$

Posterior

- **Prior**: Represents our state of knowledge about the true parameters *before* measuring the data
- **Likelihood**: The probability of observing the data assuming that a particular set of parameters is true.
- **Posterior**: Represents our state of knowledge about the parameters *after* measuring the data.

Recommend: D.S. Sivia & J. Skilling, "Data Analysis: A Bayesian Tutorial"

BAYES' THEOREM

- GW's source parameters are estimated using Bayesian inference.
- Assume a signal h exists in some data d.
- Probability that the signal has parameters $\vartheta = \{m_1, m_2, \ldots\}$ is:

$$p(\vec{\vartheta}|d,h) = \frac{p(d|\vec{\vartheta},h)\,p(\vec{\vartheta}|h)}{p(d|h)}$$

Posterior = **Likelihood**

- **Prior**: Represents our state of knowledge about the true parameters *before* measuring the data
- **Likelihood**: The probability of observing the data assuming that a particular set of parameters is true.
- **Posterior**: Represents our state of knowledge about the parameters *after* measuring the data.

Recommend: D.S. Sivia & J. Skilling, "Data Analysis: A Bayesian Tutorial"

BAYES' THEOREM

- GW's source parameters are estimated using Bayesian inference.
- Assume a signal h exists in some data d.
- Probability that the signal has parameters $\vartheta = \{m_1, m_2, \ldots\}$ is:

$$p(\vec{\vartheta}|d,h) = \frac{p(d|\vec{\vartheta},h)\,p(\vec{\vartheta}|h)}{p(d|h)}$$

Posterior = **Likelihood** **Prior**

- **Prior**: Represents our state of knowledge about the true parameters *before* measuring the data
- **Likelihood**: The probability of observing the data assuming that a particular set of parameters is true.
- **Posterior**: Represents our state of knowledge about the parameters *after* measuring the data.

Recommend: D.S. Sivia & J. Skilling, "Data Analysis: A Bayesian Tutorial"

BAYES' THEOREM

- GW's source parameters are estimated using Bayesian inference.
- Assume a signal h exists in some data d.
- Probability that the signal has parameters $\vartheta = \{m_1, m_2, \ldots\}$ is:

$$p(\vec{\vartheta}|d,h) = \frac{p(d|\vec{\vartheta},h)\,p(\vec{\vartheta}|h)}{p(d|h)}$$

Posterior = $\frac{\textbf{Likelihood} \times \textbf{Prior}}{\textbf{Evidence}}$

- **Prior**: Represents our state of knowledge about the true parameters *before* measuring the data
- **Likelihood**: The probability of observing the data assuming that a particular set of parameters is true.
- **Posterior**: Represents our state of knowledge about the parameters *after* measuring the data.

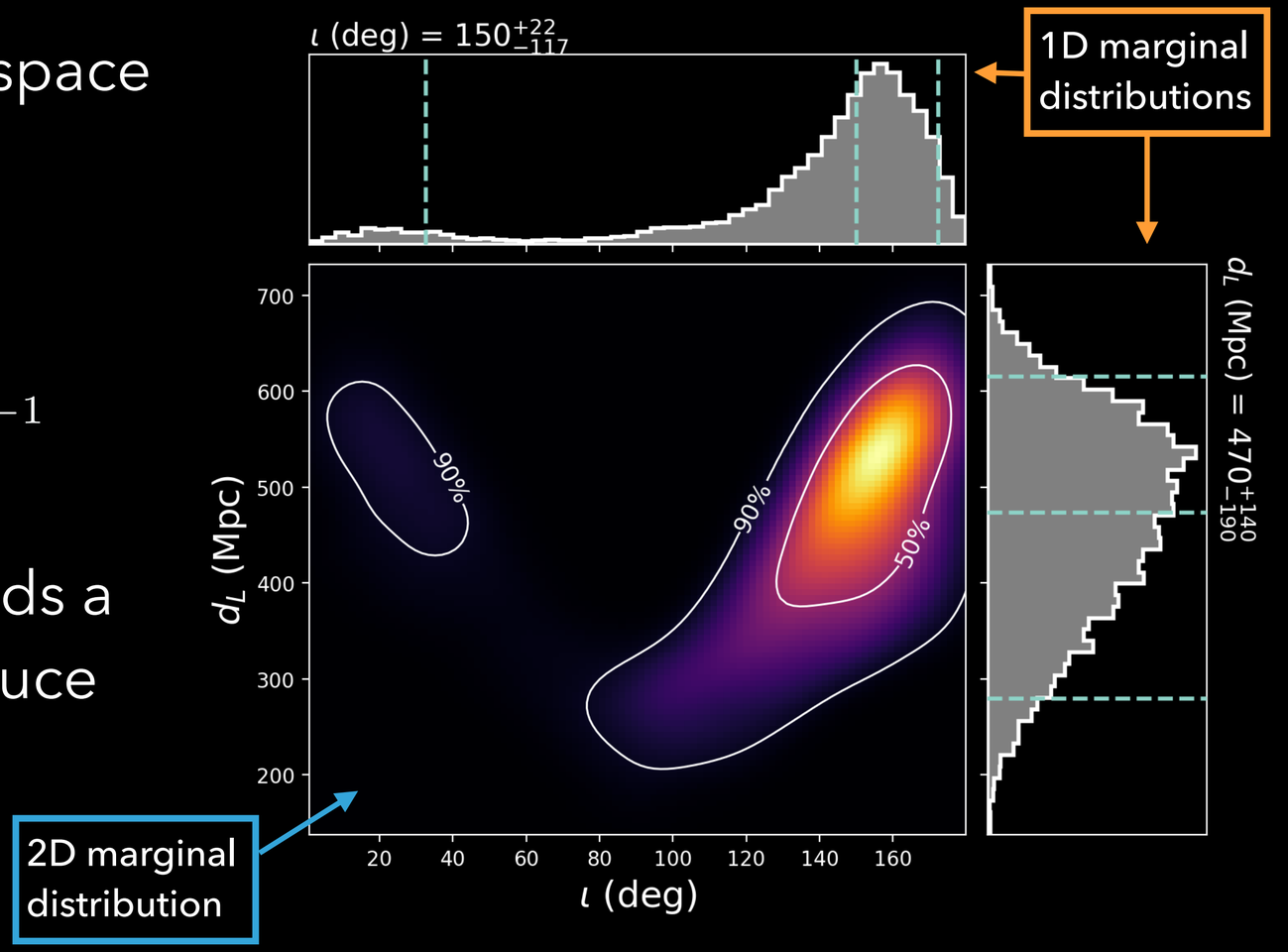
Recommend: D.S. Sivia & J. Skilling, "Data Analysis: A Bayesian Tutorial"

MARGINALIZATION

- Integrating the posterior over the prior space of a subset of parameters yields the **marginal** probability:

$$p(\vartheta_n|d,h) \propto \int p(d|\vec{\vartheta},h)p(\vec{\vartheta}|h)\mathrm{d}\vartheta_1 \cdots \mathrm{d}\vartheta_{n-1}$$

- Marginalizing all but one parameter yields a 1D distribution from which we can produce credible intervals.



Created using `pycbc_inference_plot_posterior` using data from gwcatalog.org,

EVIDENCE

$$p(\vec{\vartheta}|d,h) = \frac{p(d|\vec{\vartheta},h)\,p(\vec{\vartheta}|h)}{p(d|h)}$$

- Marginalizing the likelihood over all parameters yields the **evidence**:

$$p(d|h) = \int p(d|\vec{\vartheta},h)p(\vec{\vartheta}|h)\mathrm{d}\vec{\vartheta}$$

The evidence can be used for model selection. Taking the ratio of evidences for two models yields the **Bayes factor**:

$$\mathcal{B}(A,B|d) = \frac{p(d|B)}{p(d|A)}$$

- A Bayes factor > 1 indicates model B is favored over model A
- Generally only necessary to estimate evidence for model selection.
- If you are only doing parameter estimation for a single model, it is sufficient to measure the posterior up to a proportionality constant.

LIKELIHOOD

$$p(\vec{\vartheta}|d,h) = \frac{p(d|\vec{\vartheta},h)p(\vec{\vartheta}|h)}{p(d|h)}$$

- The **likelihood** function requires both a signal model and a noise model.
- In gravitational-wave analyses it is common to assume wide-sense stationary (WSS) Gaussian noise. In that case:

$$\log p(d|\vec{\vartheta},h) \propto -\frac{1}{2}\sum_{i=1}^{N_d}\left\langle d_i - h_i(\vec{\vartheta}),\, d_i - h_i(\vec{\vartheta})\right\rangle$$

where:

$$\langle a,\, b\rangle = 4\Re\int_0^\infty \frac{\tilde{a}^*(f)\tilde{b}(f)}{S_n(f)}\mathrm{d}f$$

LIKELIHOOD

$$p(\vec{\vartheta}|d,h) = \frac{p(d|\vec{\vartheta},h)p(\vec{\vartheta}|h)}{p(d|h)}$$

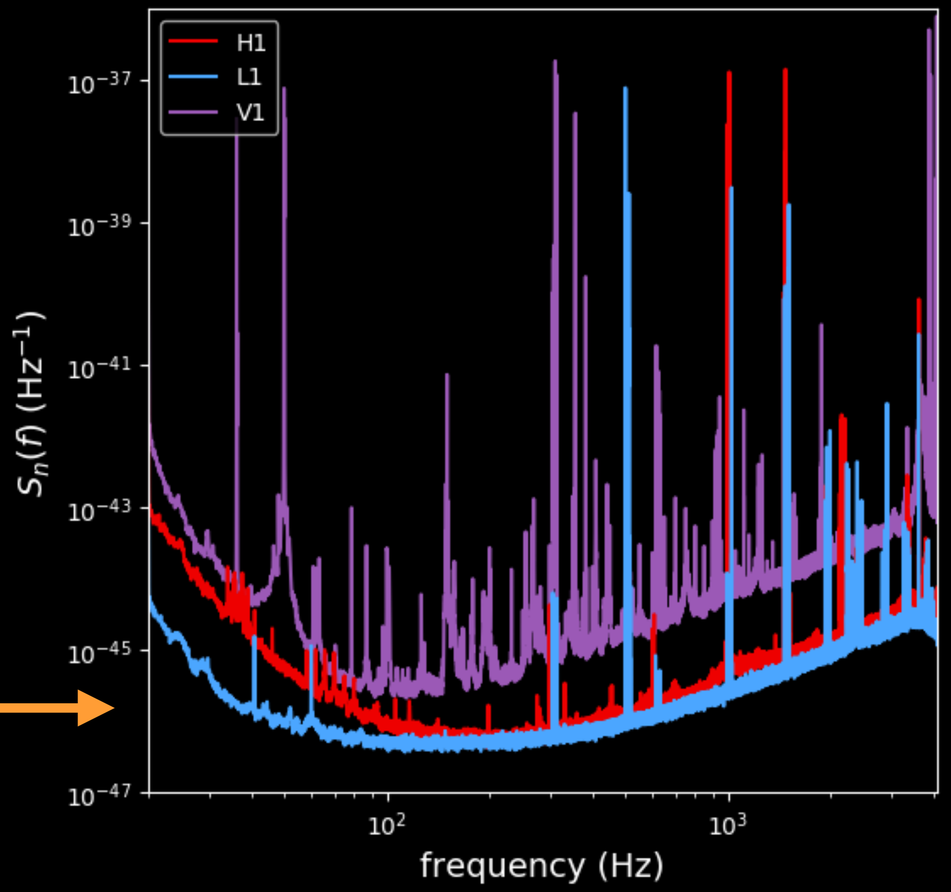
- The **likelihood** function requires both a signal model and a noise model.
- In gravitational-wave analyses it is common to assume wide-sense stationary (WSS) Gaussian noise. In that case:

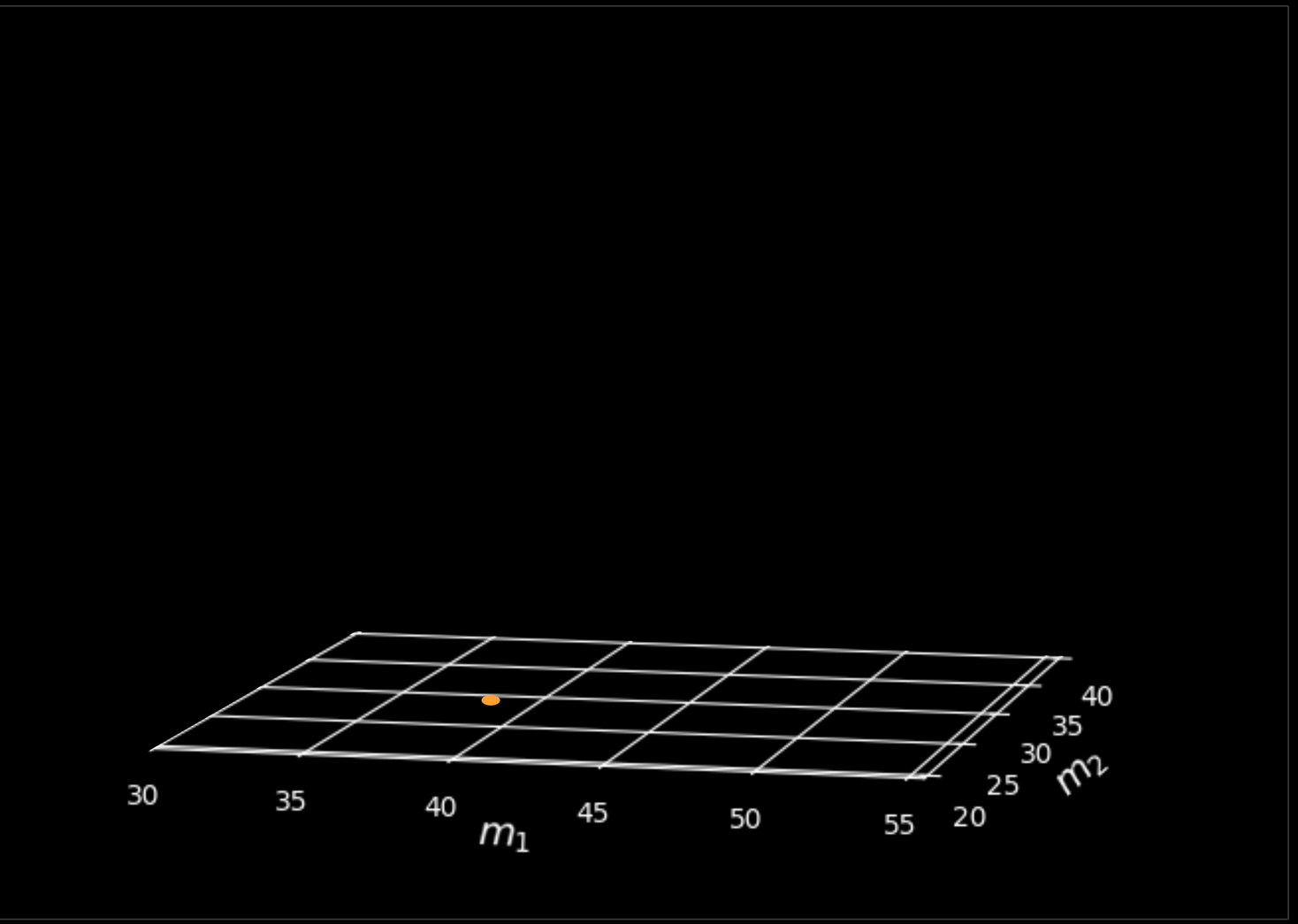
$$\log p(d|\vec{\vartheta},h) \propto -\frac{1}{2}\sum_{i=1}^{N_d}\left\langle d_i - h_i(\vec{\vartheta}),\, d_i - h_i(\vec{\vartheta})\right\rangle$$

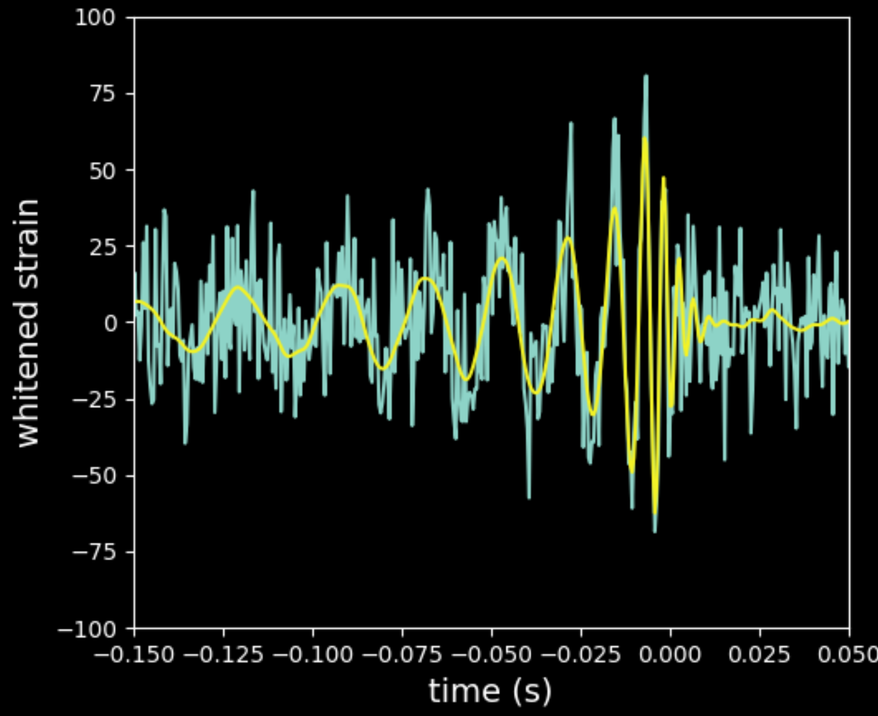
where:

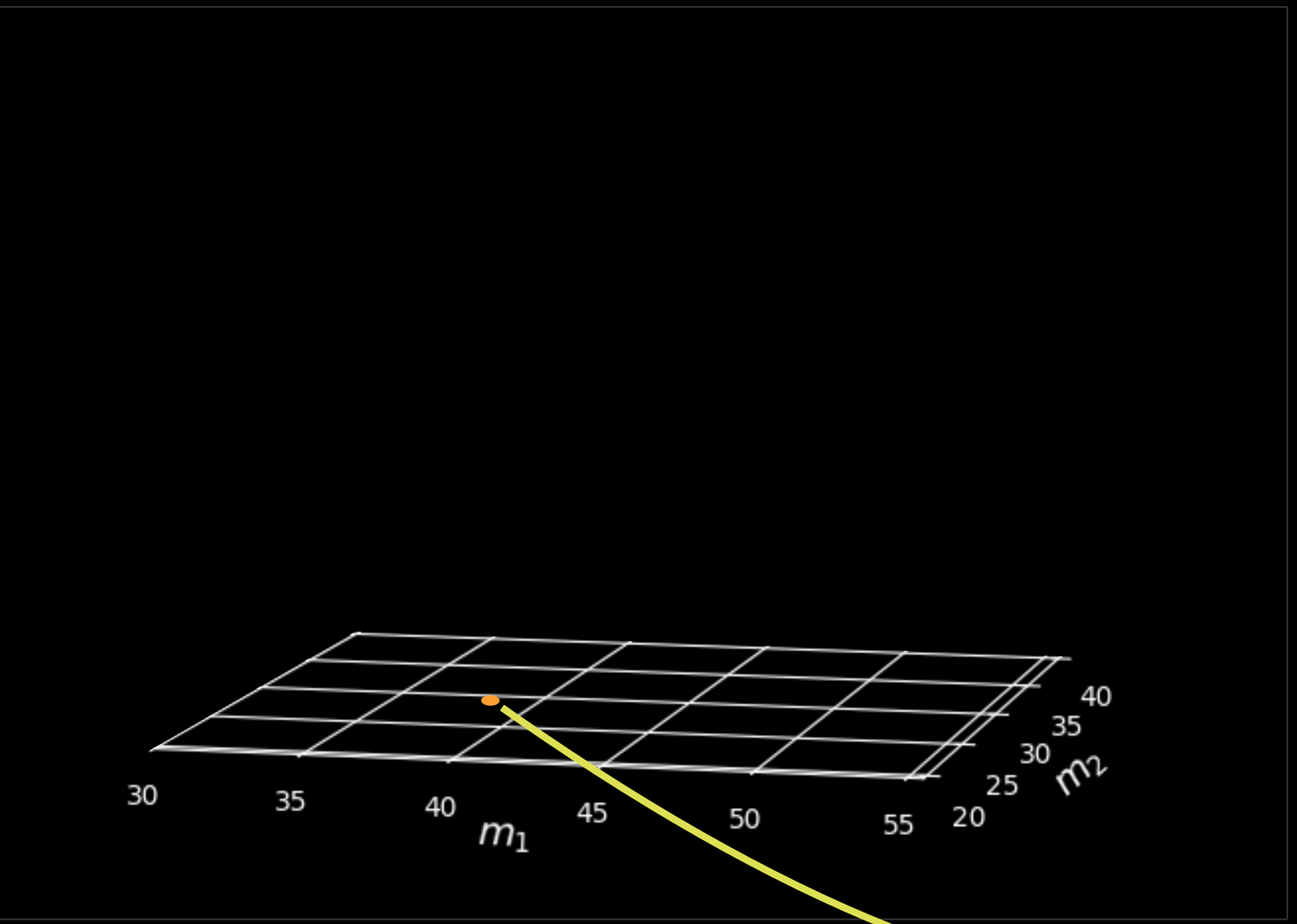
$$\langle a,\, b\rangle = 4\Re\int_0^\infty \frac{\tilde{a}^*(f)\tilde{b}(f)}{S_n(f)}\mathrm{d}f$$

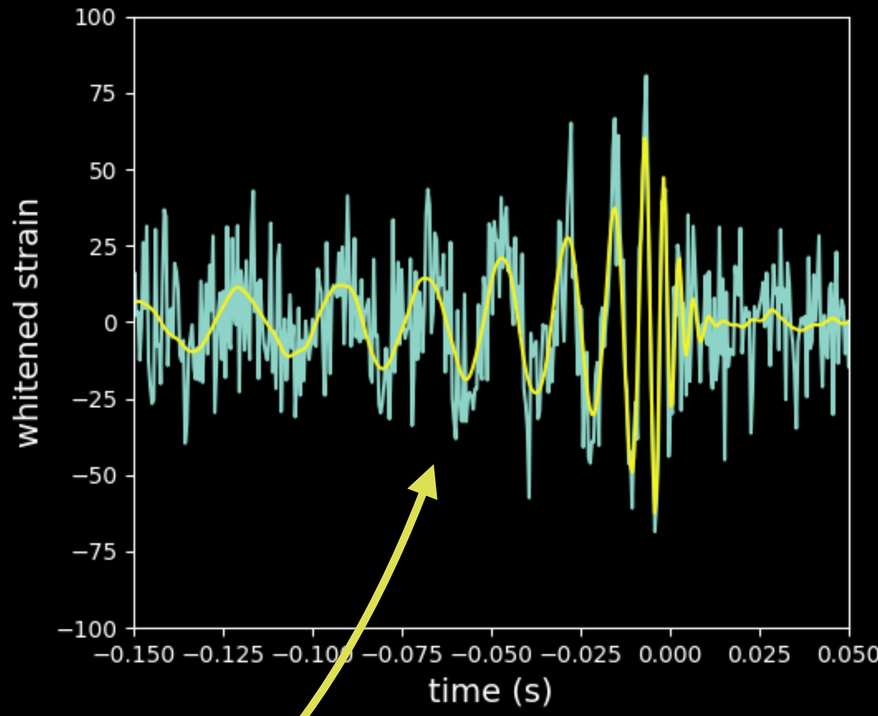
Power Spectral Density (PSD)

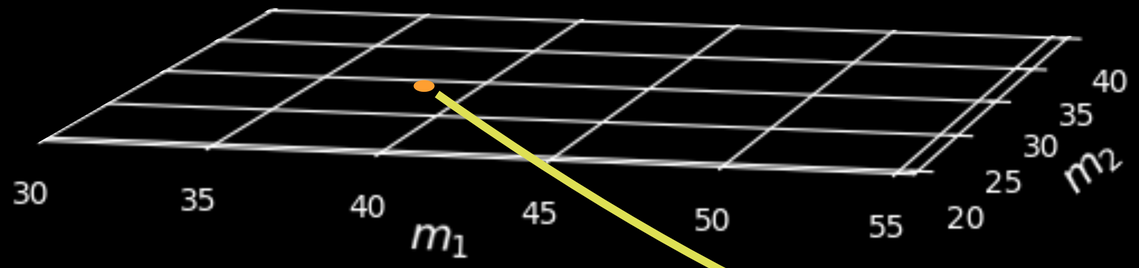



30
35
40
45
50
55
m_1
20
25
30
35
40
m_2

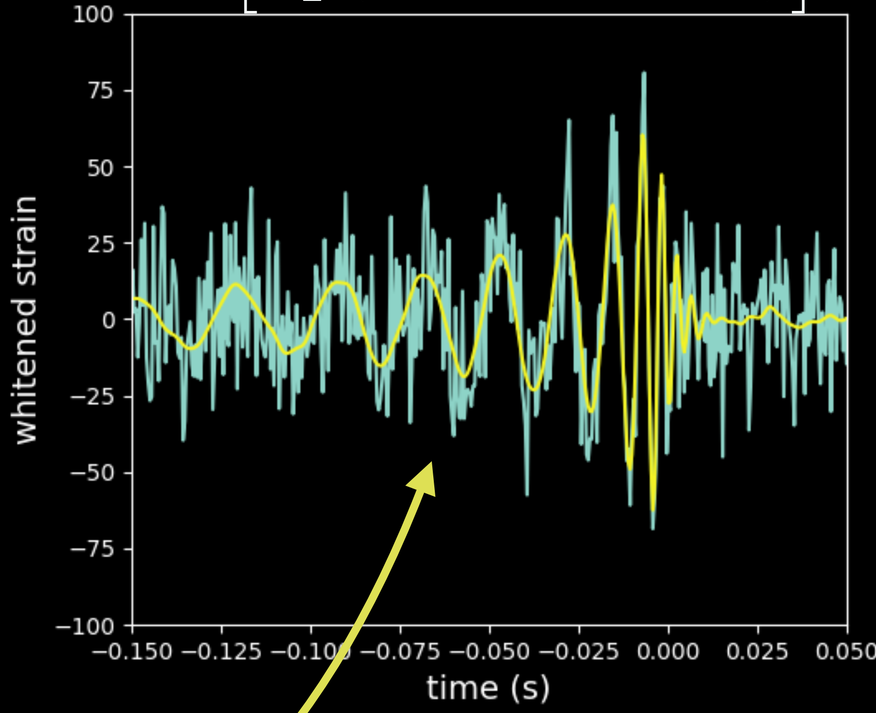

100
75
50
25
0
−25
−50
−75
−100
whitened strain
−0.150
−0.125
−0.100
−0.075
−0.050
−0.025
0.000
0.025
0.050
time (s)

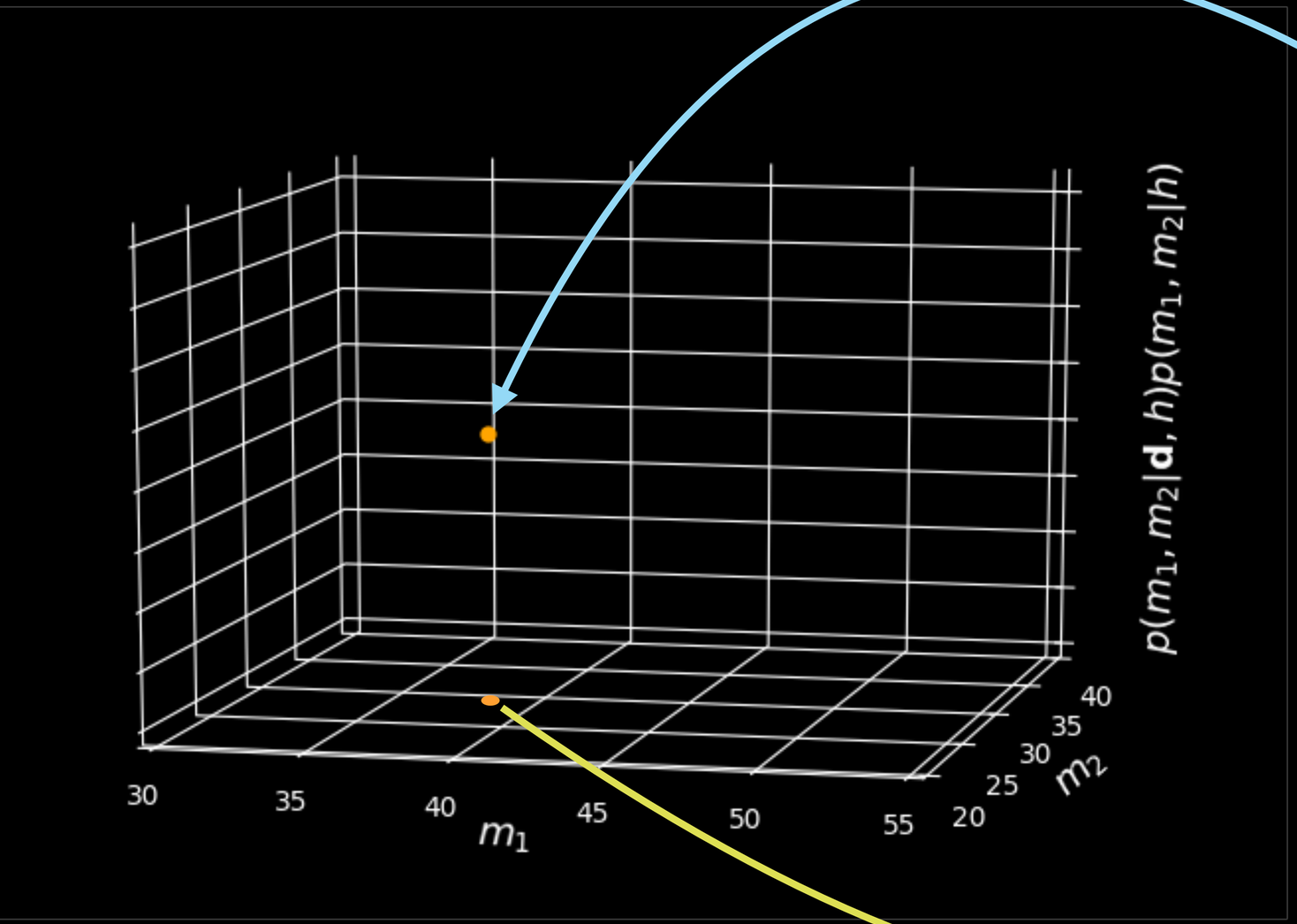

30
35
40
45
50
55
m_1
20
25
30
35
40
m_2


whitened strain
100
75
50
25
0
−25
−50
−75
−100
−0.150
−0.125
−0.100
−0.075
−0.050
−0.025
0.000
0.025
0.050
time (s)

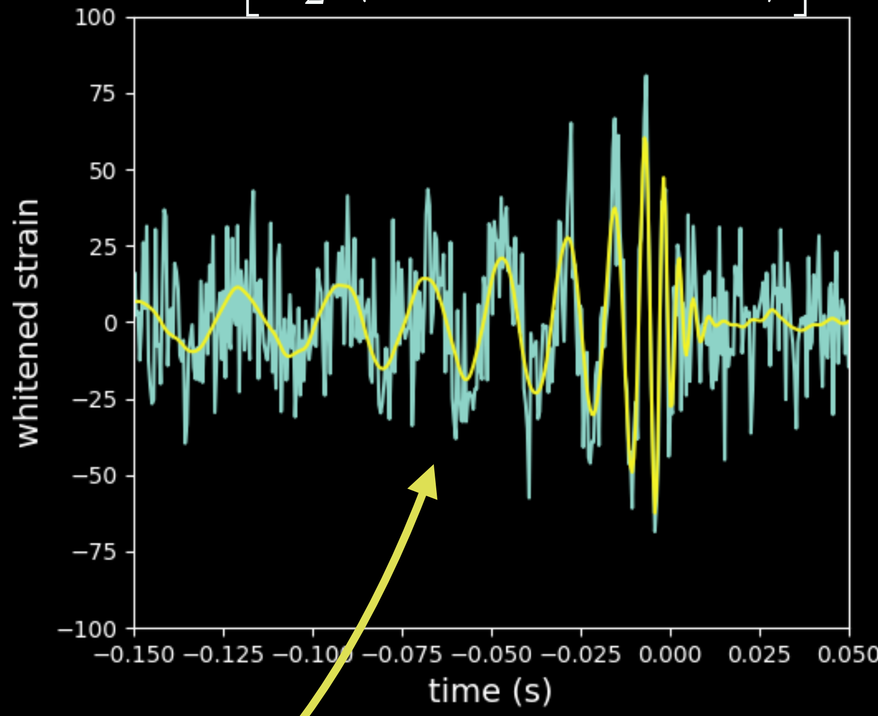


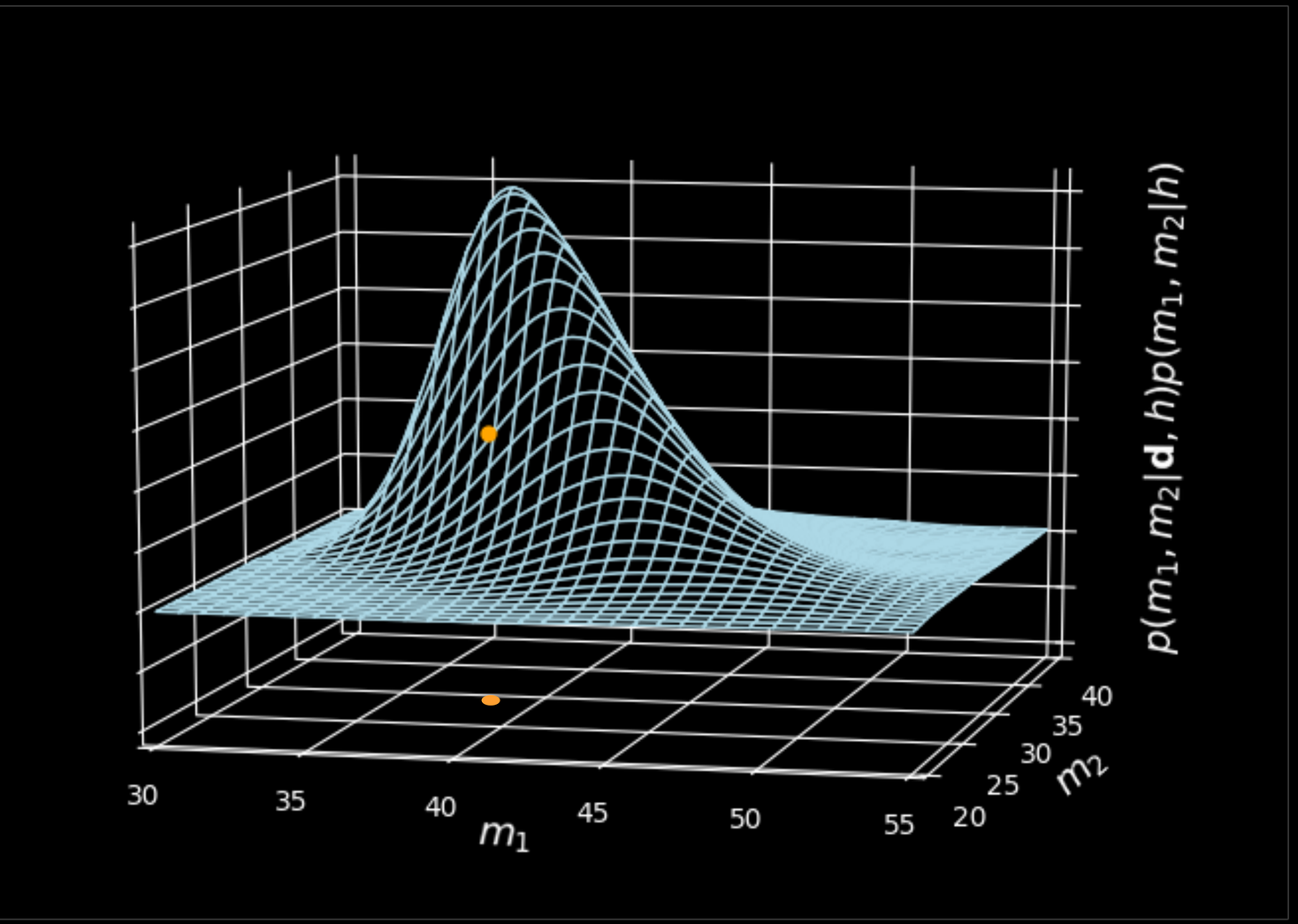
$$\exp\left[-\frac{1}{2}\left\langle \mathbf{d}-\mathbf{h}(\vec{\vartheta}),\mathbf{d}-\mathbf{h}(\vec{\vartheta})\right\rangle\right]$$

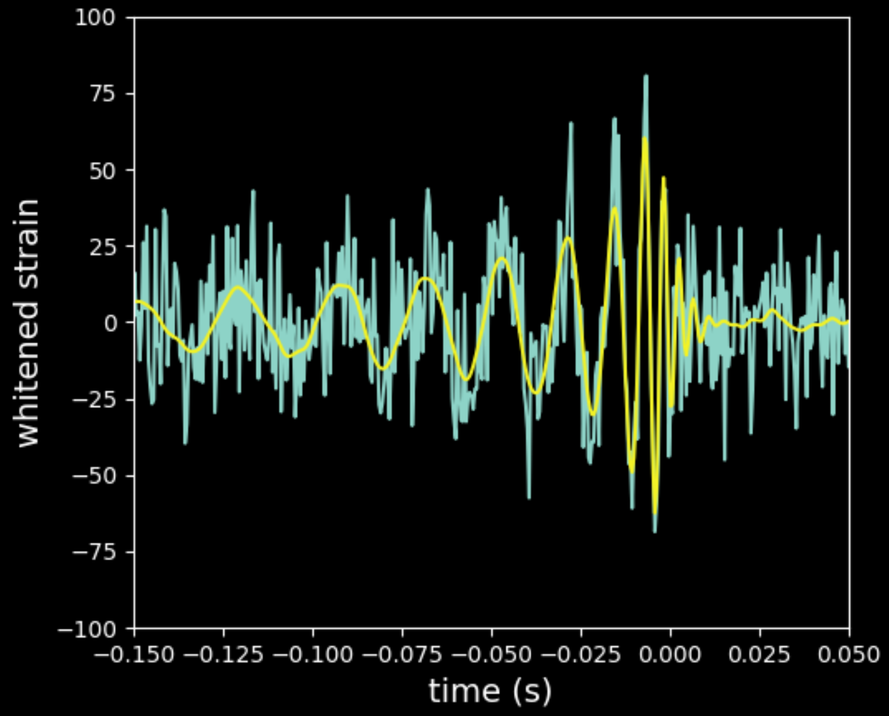




$$\exp\left[-\frac{1}{2}\left\langle \mathbf{d}-\mathbf{h}(\vec{\vartheta}), \mathbf{d}-\mathbf{h}(\vec{\vartheta})\right\rangle\right]$$




$p(m_1, m_2|\mathbf{d}, h)p(m_1, m_2|h)$
30
35
40
45
50
55
m_1
20
25
30
35
40
m_2

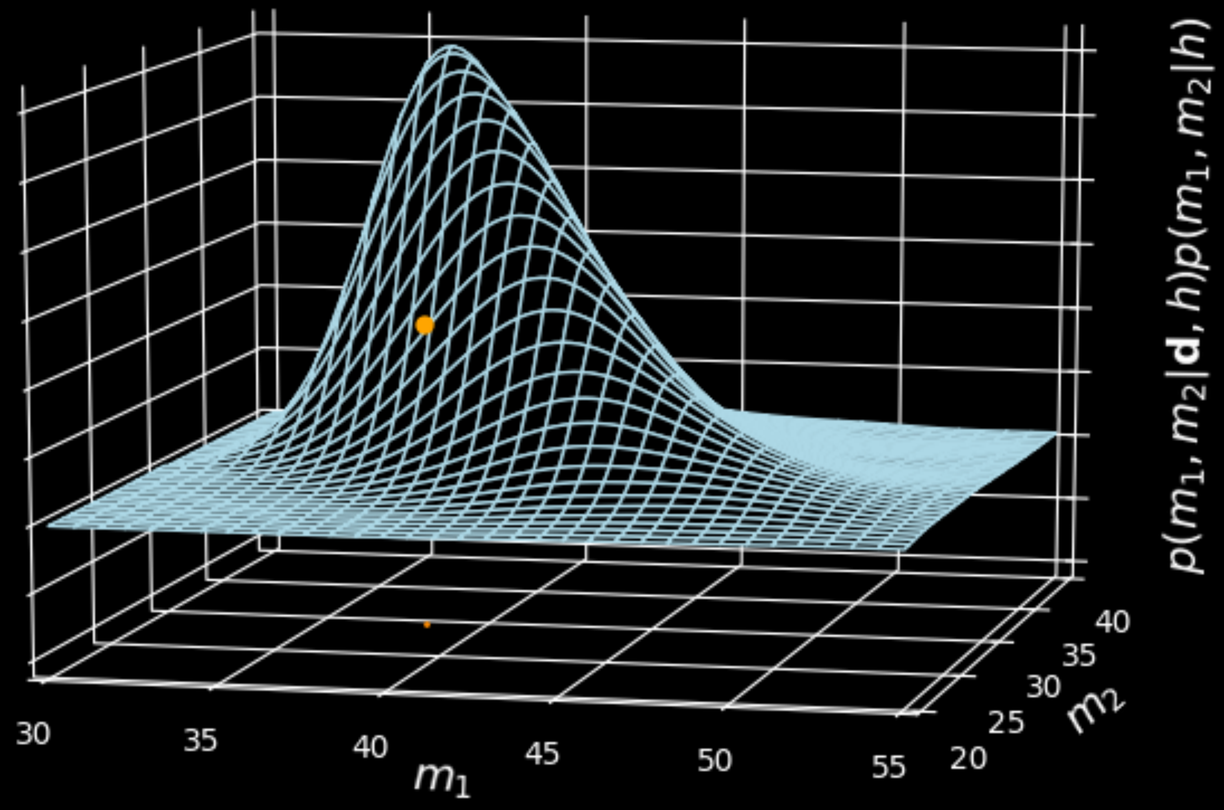

whitened strain
100
75
50
25
0
−25
−50
−75
−100
−0.150
−0.125
−0.100
−0.075
−0.050
−0.025
0.000
0.025
0.050
time (s)

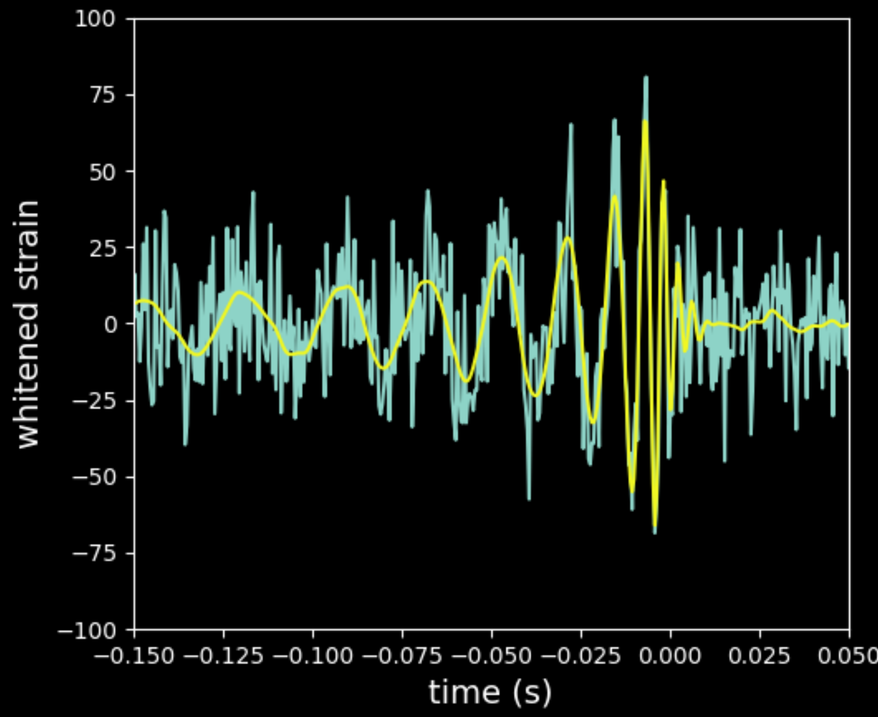
STOCHASTIC SAMPLERS

- Necessary to use stochastic samplers to map out the posterior
- PyCBC Inference has support for two general classes of samplers:
  - Markov chain Monte Carlo (MCMC): emcee, emcee_pt, epsie
  - Nested samplers: dynesty, PyMultiNest, cpnest, ultranest
- Hamiltonian Monte Carlo (e.g., Stan) is another method, but difficult to apply to gravitational waves*

* Bouffanis & Porter, PRD 100 104023 (2019), arXiv:1810.07443

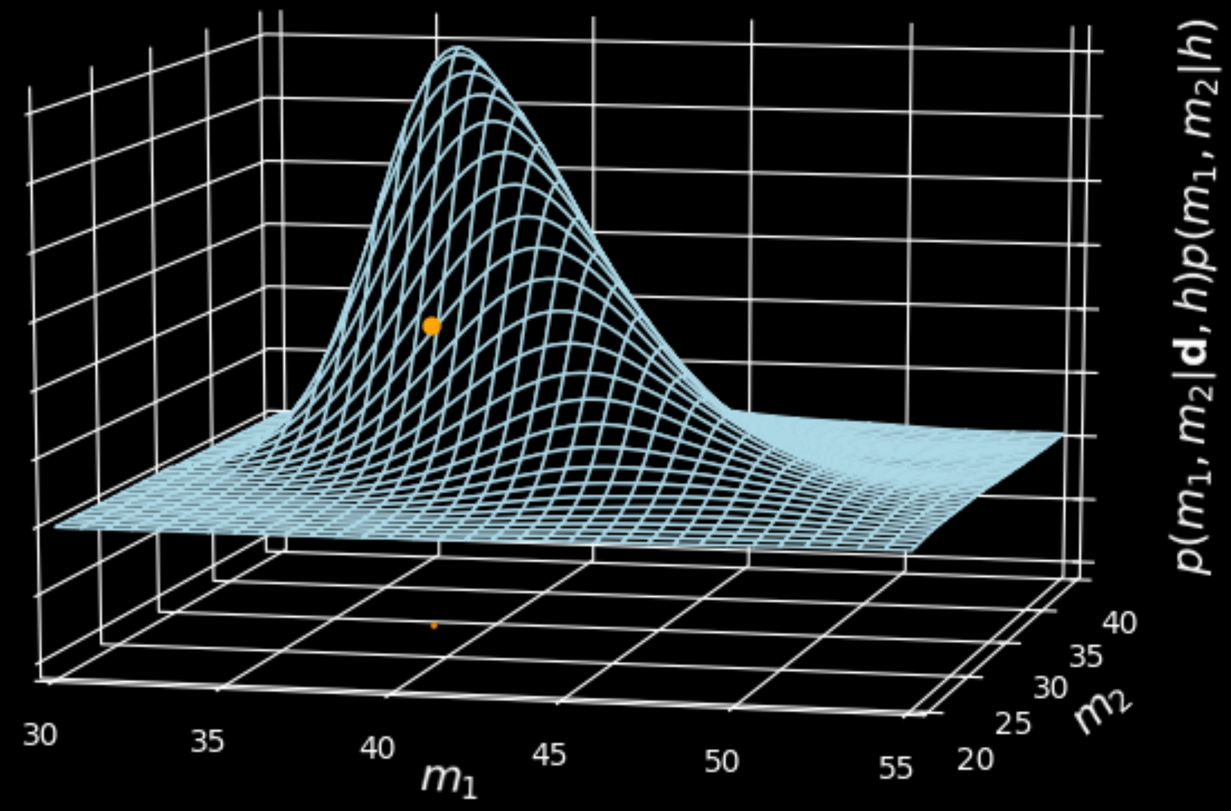
MARKOV CHAIN MONTE CARLO (MCMC)

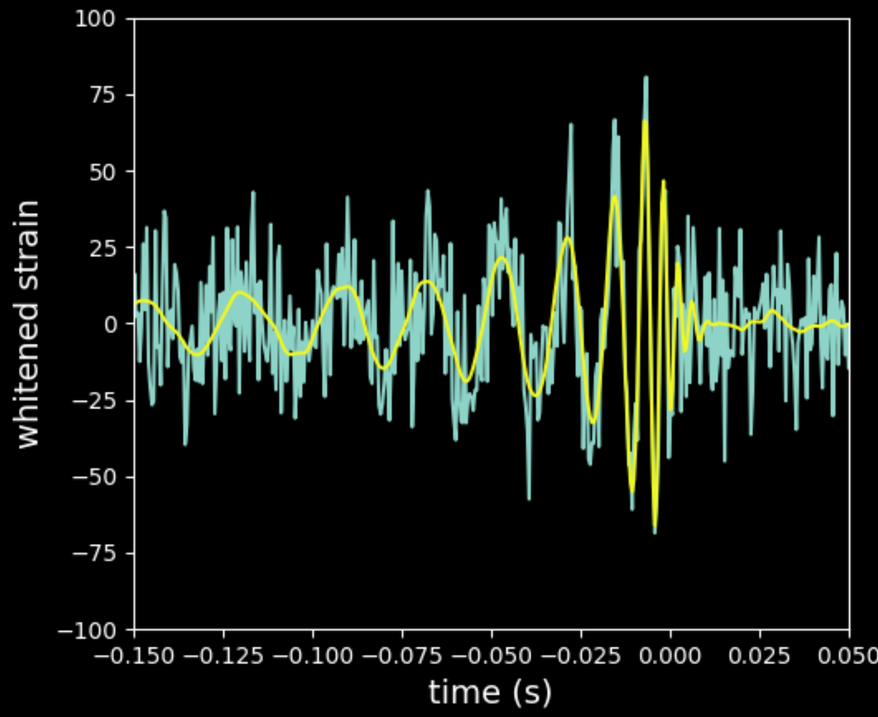




Metropolis-Hastings algorithm:

MARKOV CHAIN MONTE CARLO (MCMC)

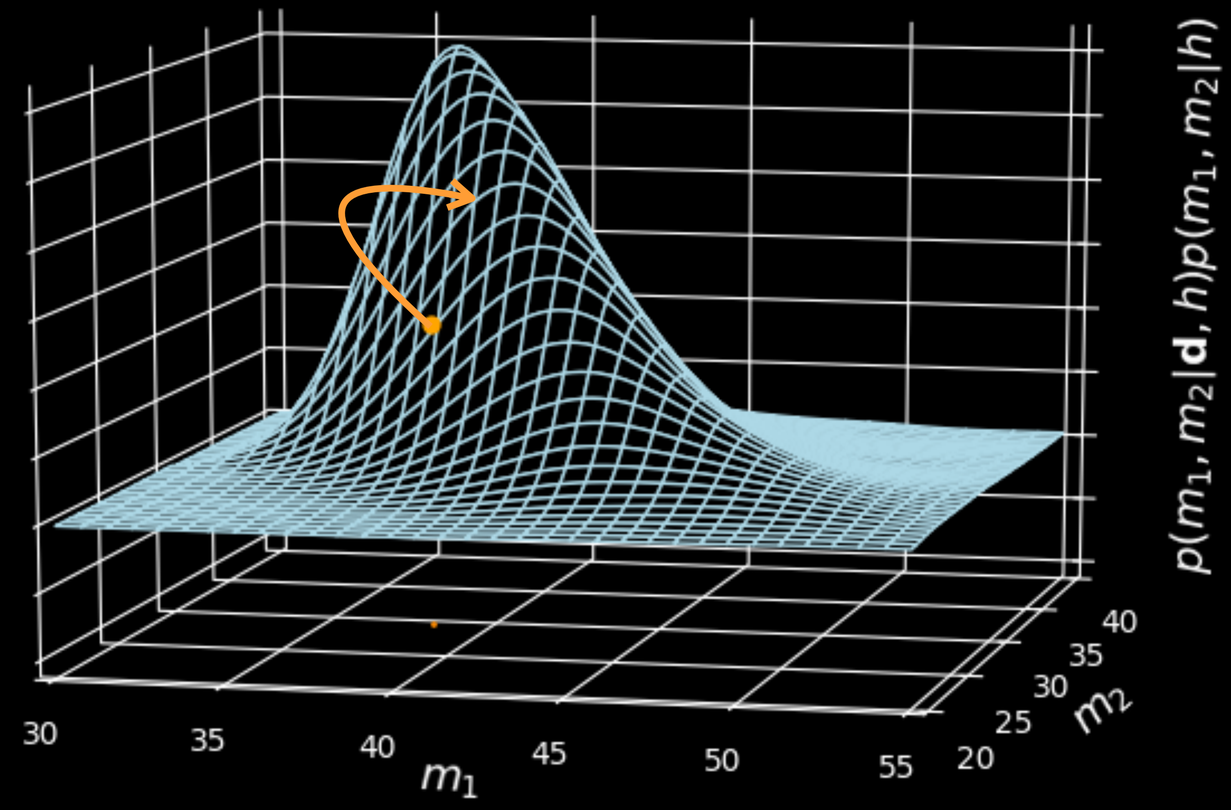


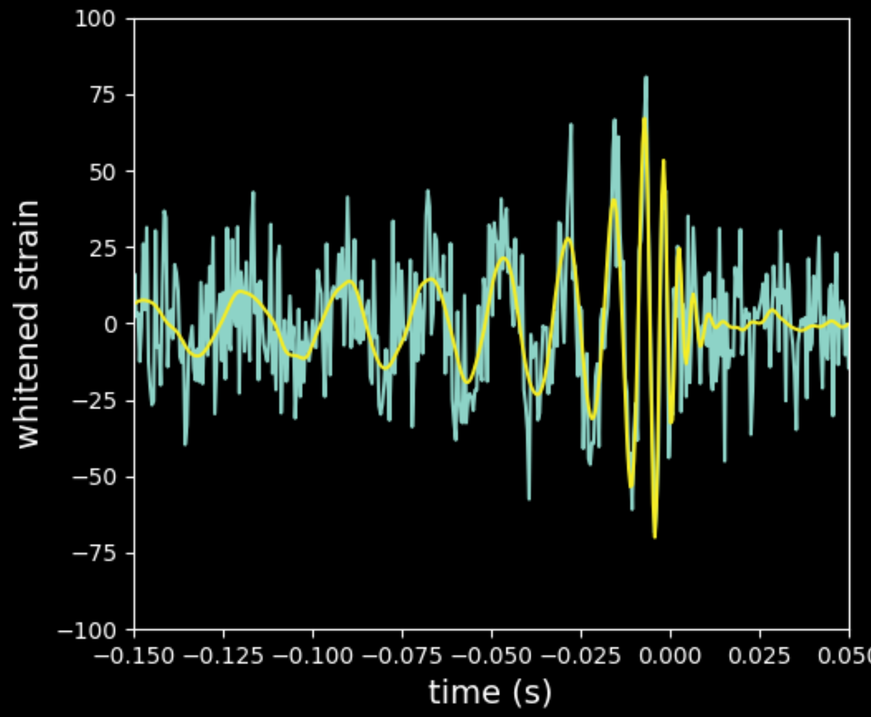


Metropolis-Hastings algorithm:

1. Start with arbitrary point ϑ_0

MARKOV CHAIN MONTE CARLO (MCMC)

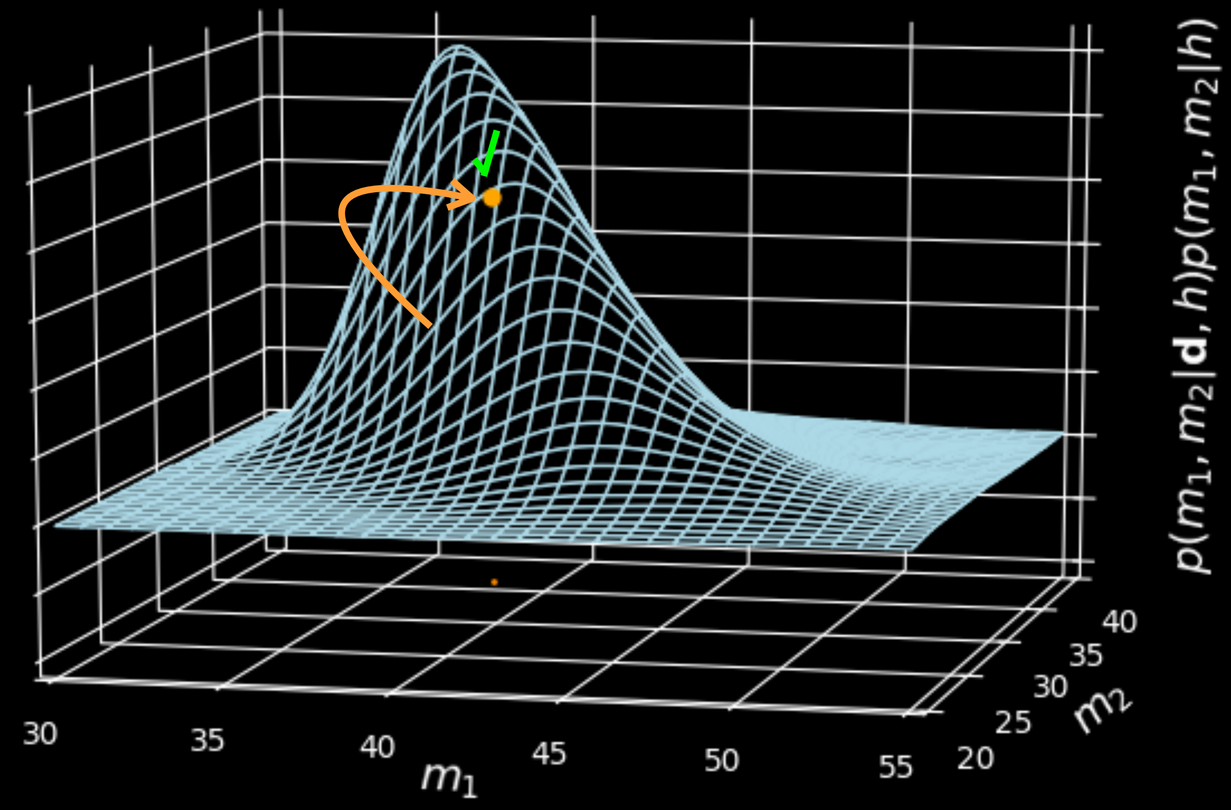


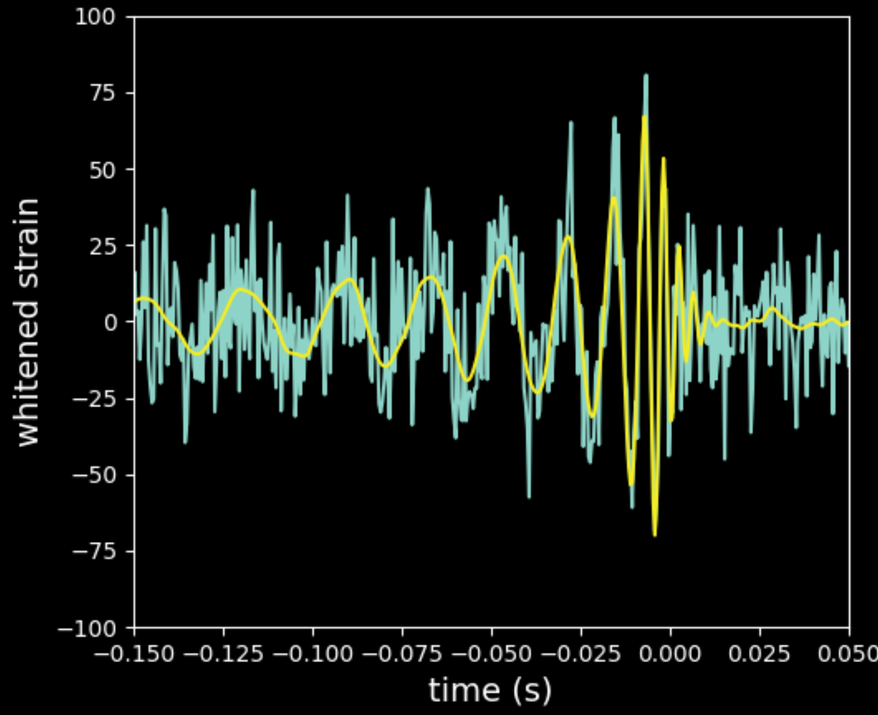


Metropolis-Hastings algorithm:

1. Start with arbitrary point ϑ_0
2. Draw a new point ϑ_1 from arbitrary distribution $Q(\vartheta_1 \mid \vartheta_0)$

MARKOV CHAIN MONTE CARLO (MCMC)



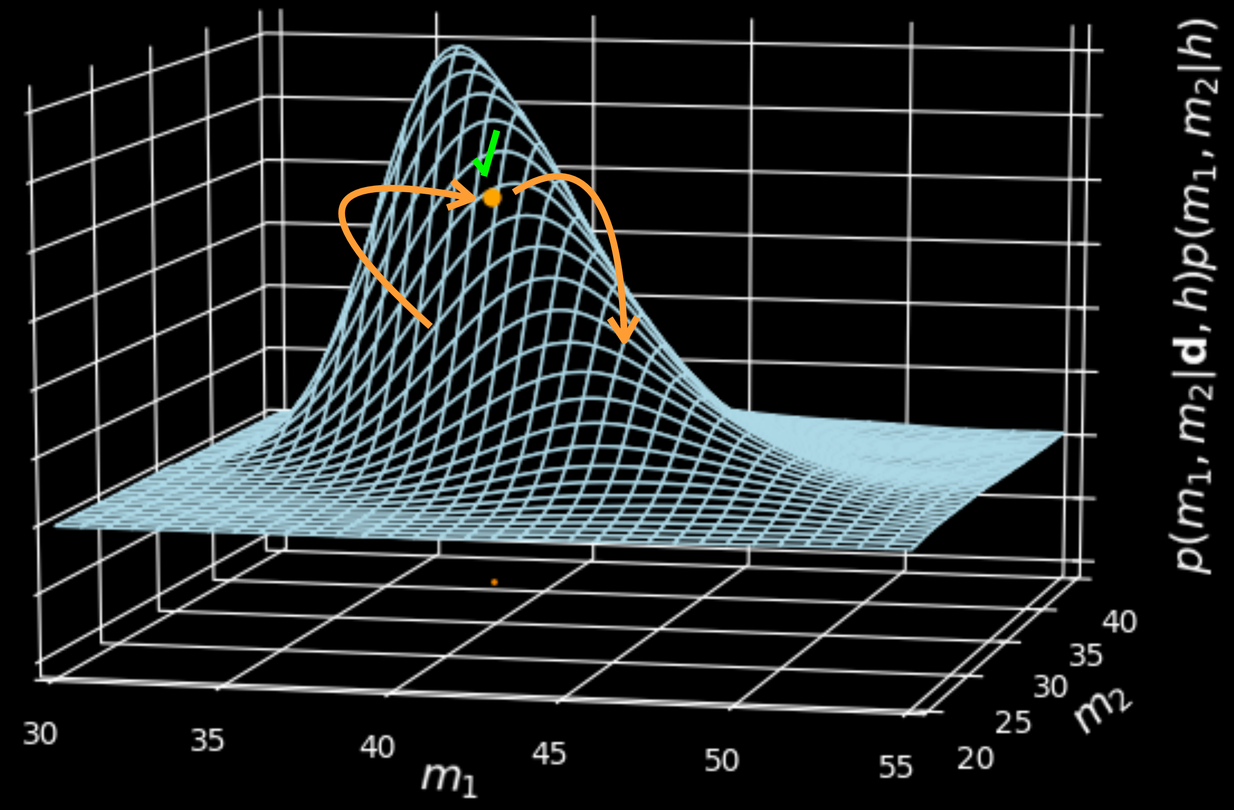


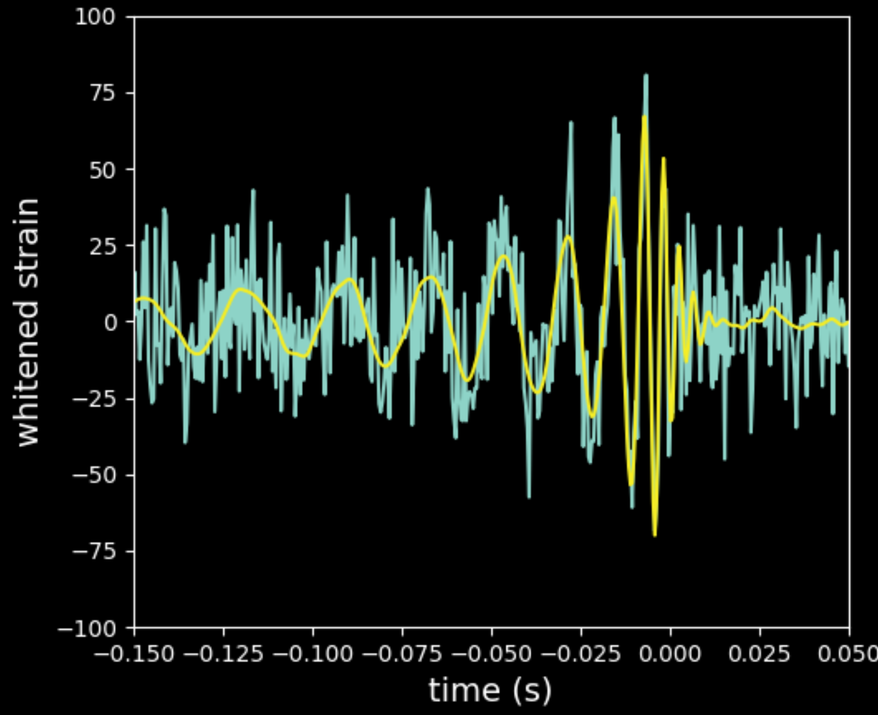
Metropolis-Hastings algorithm:

1. Start with arbitrary point ϑ_0
2. Draw a new point ϑ_1 from arbitrary distribution $Q(\vartheta_1 \mid \vartheta_0)$
3. Accept with probability

$$A = \min\left[1,\ \frac{p(\vartheta_1|\mathbf{d},h)}{p(\vartheta_0|\mathbf{d},h)}\frac{Q(\vartheta_0|\vartheta_1)}{Q(\vartheta_1|\vartheta_0)}\right]$$

MARKOV CHAIN MONTE CARLO (MCMC)



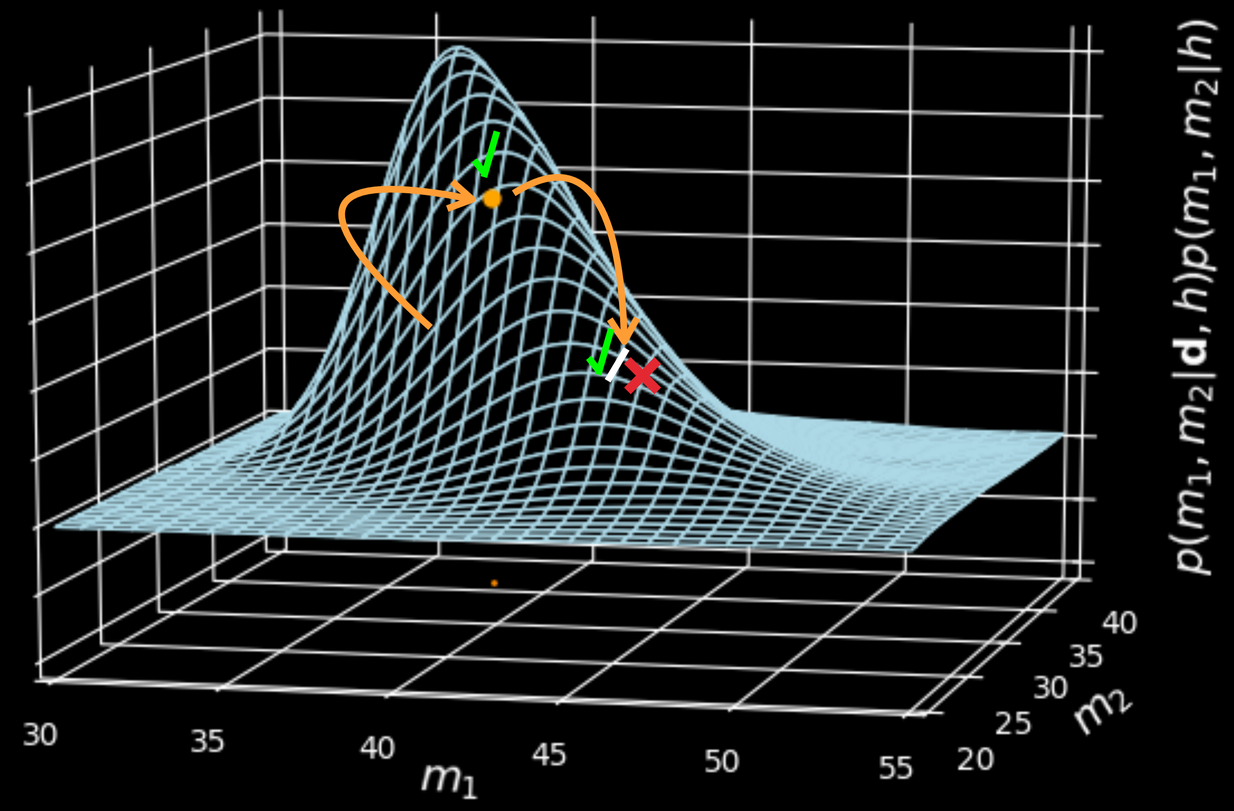


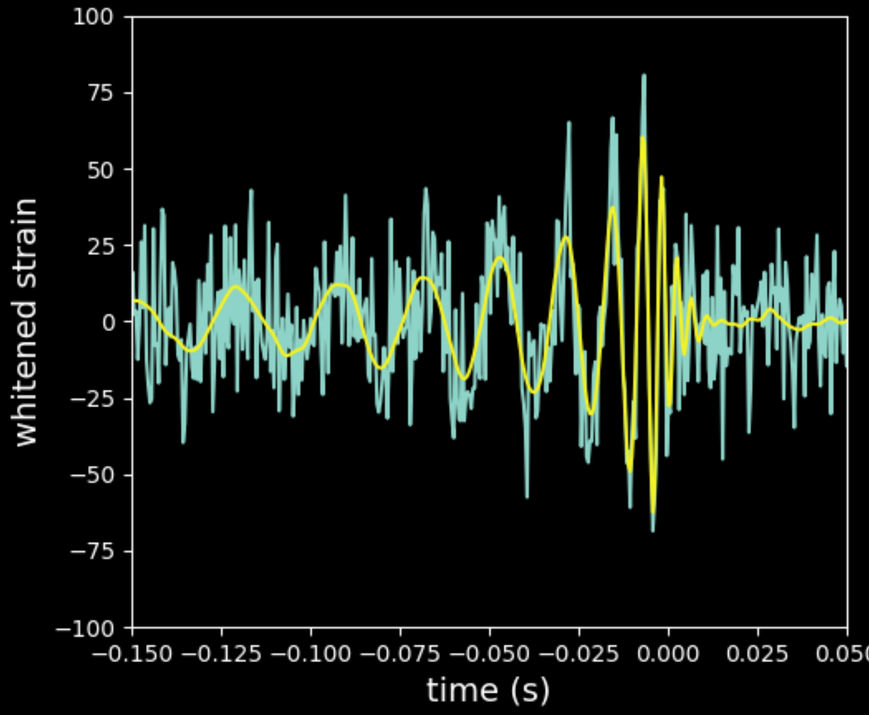
Metropolis-Hastings algorithm:

1. Start with arbitrary point ϑ_0
2. Draw a new point ϑ_1 from arbitrary distribution $Q(\vartheta_1 \mid \vartheta_0)$
3. Accept with probability

$$A = \min\left[1,\ \frac{p(\vartheta_1|\mathbf{d},h)}{p(\vartheta_0|\mathbf{d},h)}\frac{Q(\vartheta_0|\vartheta_1)}{Q(\vartheta_1|\vartheta_0)}\right]$$

MARKOV CHAIN MONTE CARLO (MCMC)



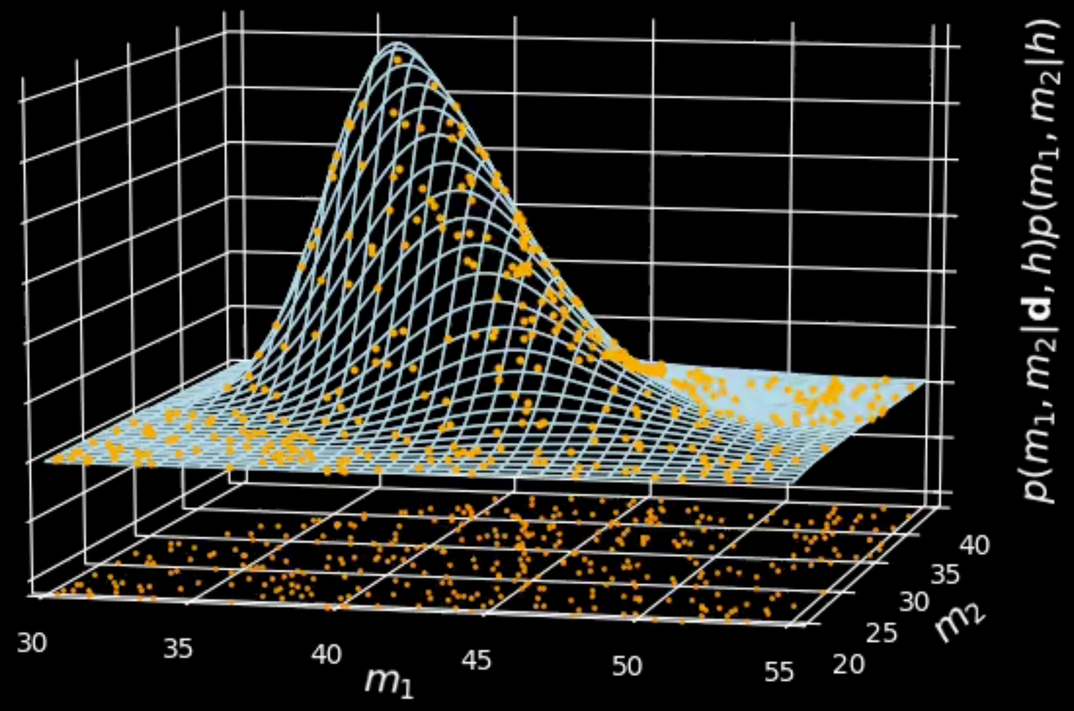


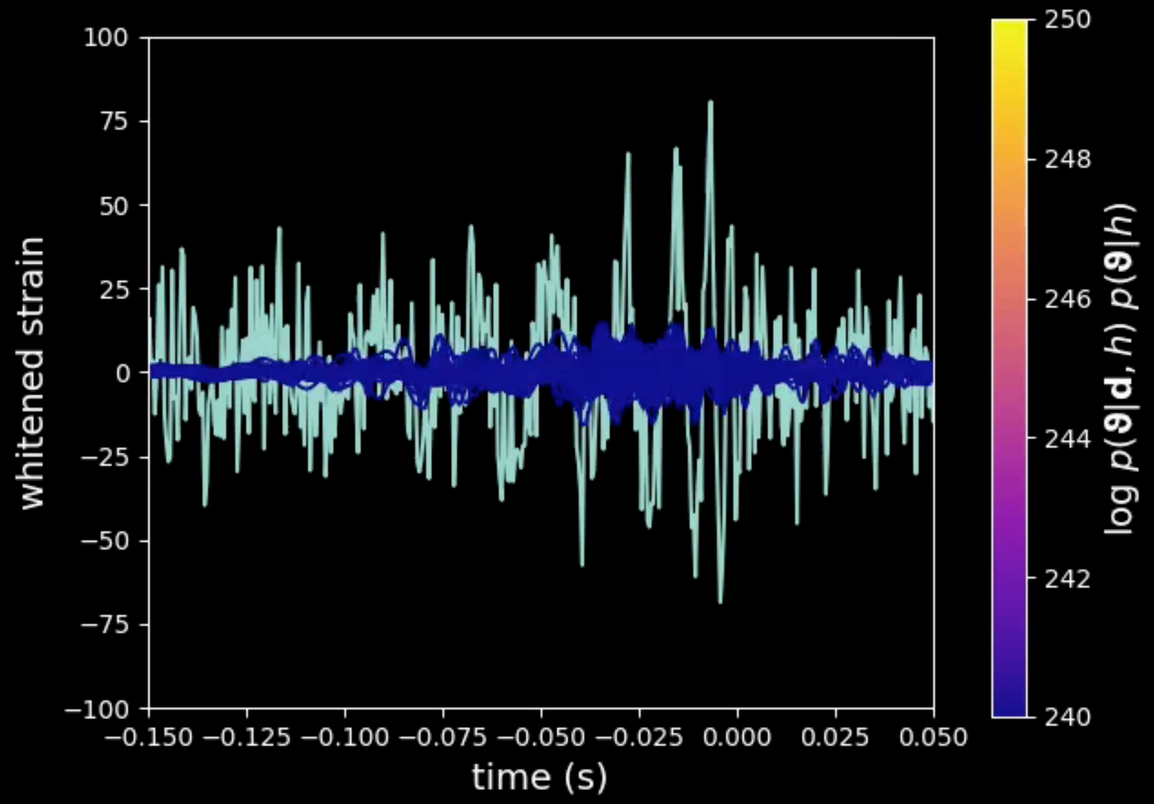
Metropolis-Hastings algorithm:

1. Start with arbitrary point ϑ_0
2. Draw a new point ϑ_1 from arbitrary distribution $Q(\vartheta_1 \mid \vartheta_0)$
3. Accept with probability

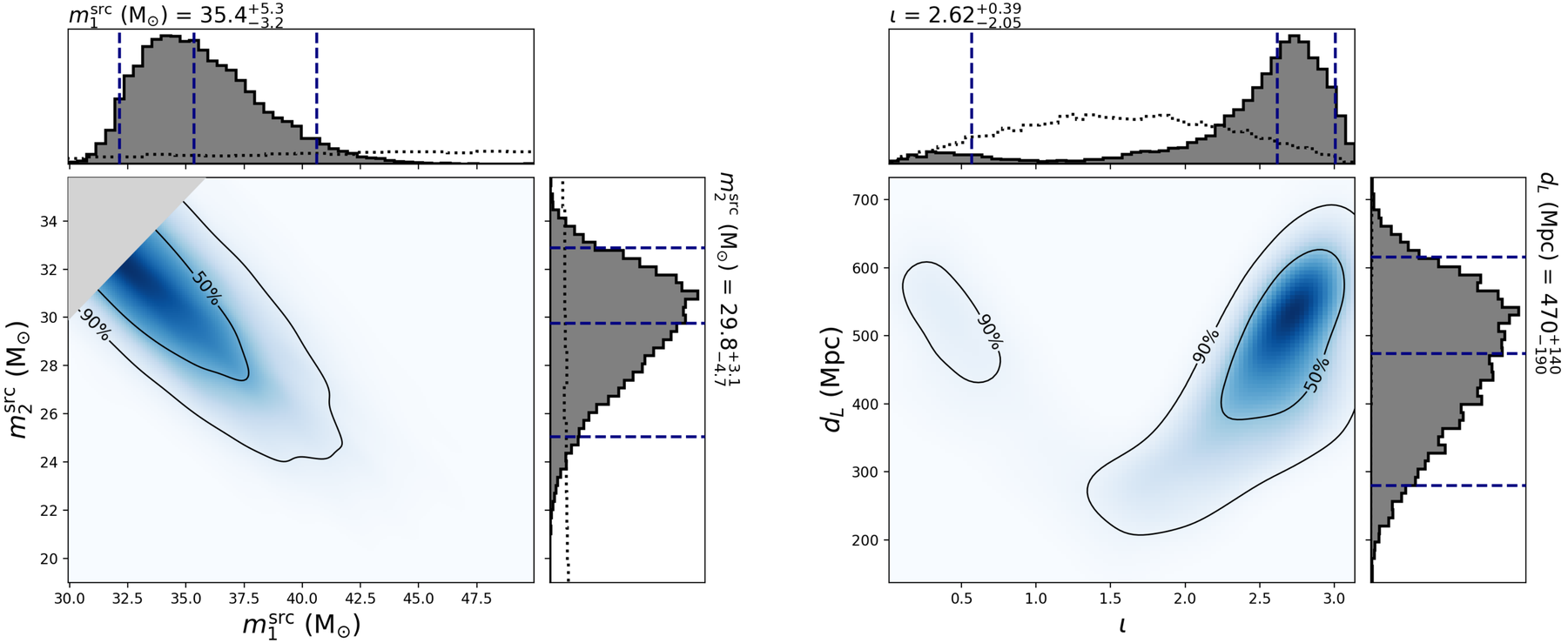
$$A = \min\left[1,\ \frac{p(\vartheta_1|\mathbf{d},h)}{p(\vartheta_0|\mathbf{d},h)}\frac{Q(\vartheta_0|\vartheta_1)}{Q(\vartheta_1|\vartheta_0)}\right]$$

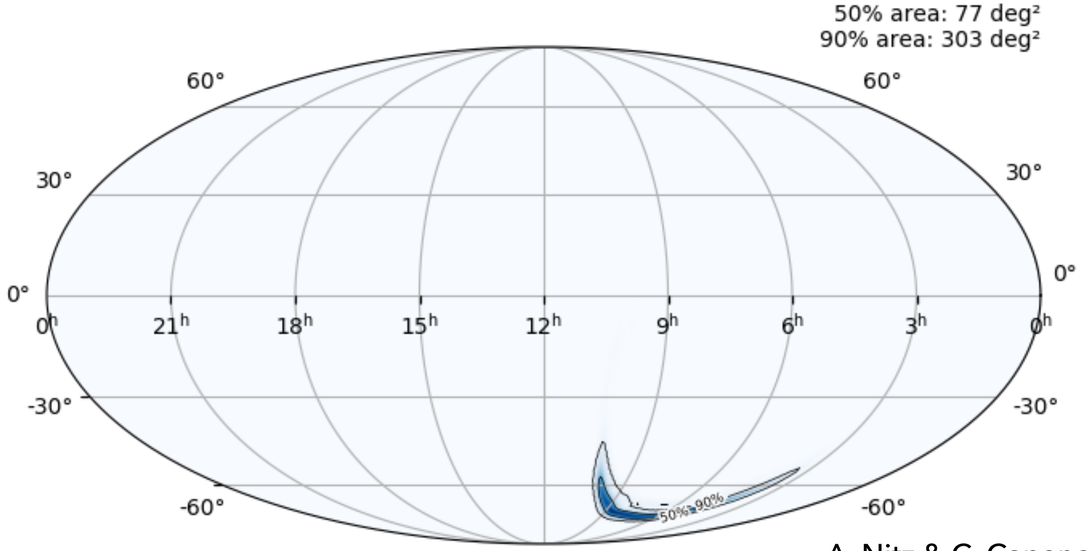
MARKOV CHAIN MONTE CARLO (MCMC)





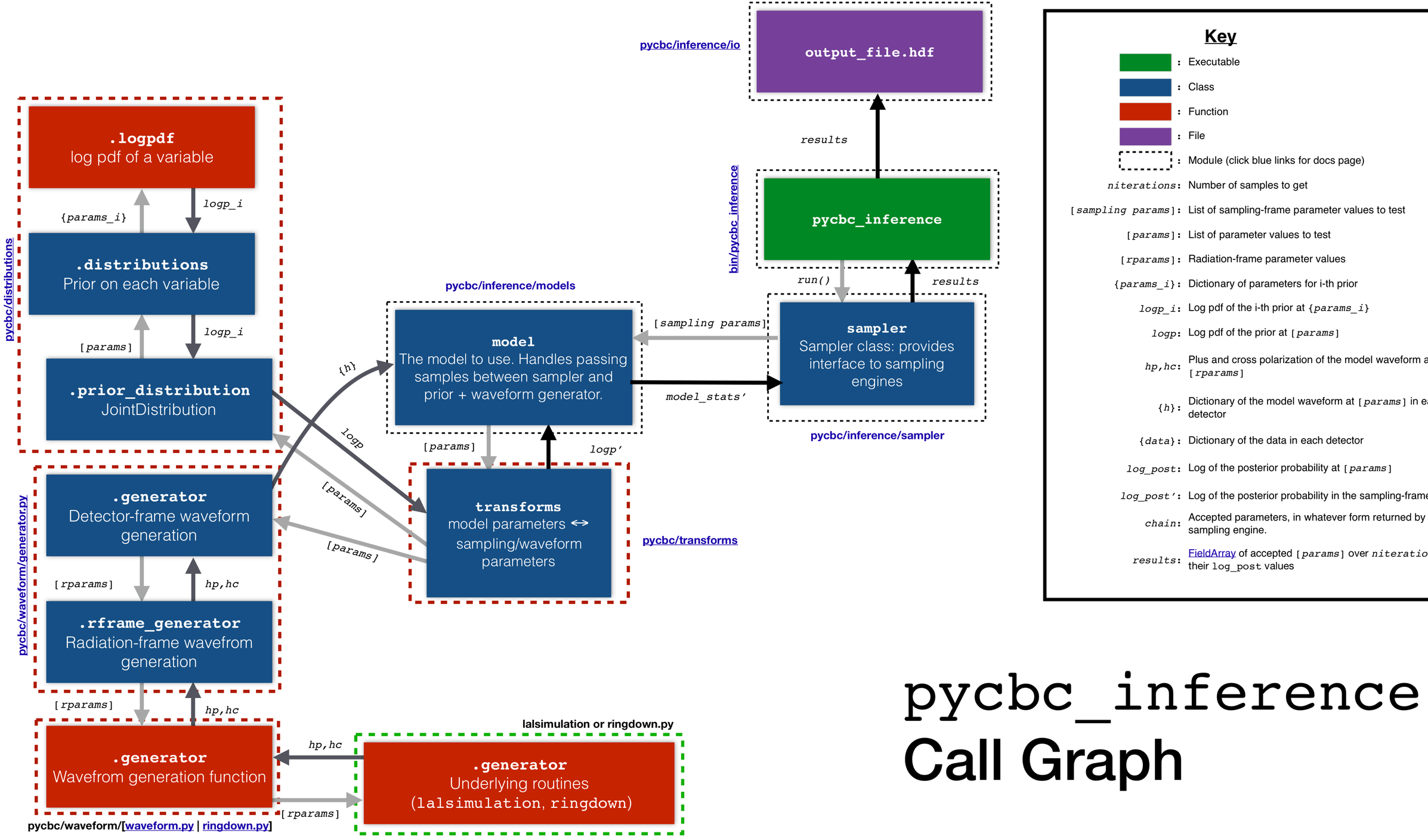
This is a movie; see Keynote version.





A. Nitz & C. Capano, gwcatalog.org/top30_bbh

MODELS & SAMPLERS IN PYCBC INFERENCE



pycbc_inference
Call Graph

MODELS IN PYCBC INFERENCE

- A “model” defines the type of analysis that you are doing
- Defines the likelihood function
- We will be looking at models in **tutorial 1**
- Two general types in PyCBC Inference
- **Analytic models**: define an analytic likelihood, used for testing purposes
  - `test_normal`, `test_eggbox`, `test_rosenbrock`, `test_volcano`
- **Data-based models**: likelihood requires gravitational-wave data and waveform model to calculate inner products
  - `gaussian_noise`, `marginalized_phase`, `relative`, `single_template`

DATA-BASED MODELS

- All of the (current) data-based models in PyCBC assume stationary Gaussian noise
- Differences are various approximations or simplifying assumptions to speed up likelihood evaluation & convergence
- `gaussian_noise`: standard Gaussian likelihood
- `marginalized_phase`: analytically marginalizes over unknown phase; can be used with waveform models that go like $h(f, \phi) \sim h(f, 0) \exp[i\, \phi]$
- `single_template`: fixes intrinsic parameters (masses, spins), just varies extrinsic (sky location, inclination, distance); useful for quickly getting estimate of sky location
- `relative`: uses “relative binning” method from Zackay et al*.; linear interpolation of likeihood around peak to allow for fast calculation

* Zackay et al., arXiv:1806.08792 (2018)

SAMPLERS

- PyCBC Inference uses external packages for samplers
- MCMC: emcee, emcee_pt, epsie
- Nested samplers: dynesty, PyMultiNest, cpnest, ultranest
- Different samplers have strengths and weaknesses

SAMPLERS

- PyCBC Inference uses external packages for samplers
- MCMC: emcee, emcee_pt, epsie
- Nested samplers: dynesty, PyMultiNest, cpnest, ultranest
- Different samplers have strengths and weaknesses

MCMC SAMPLERS

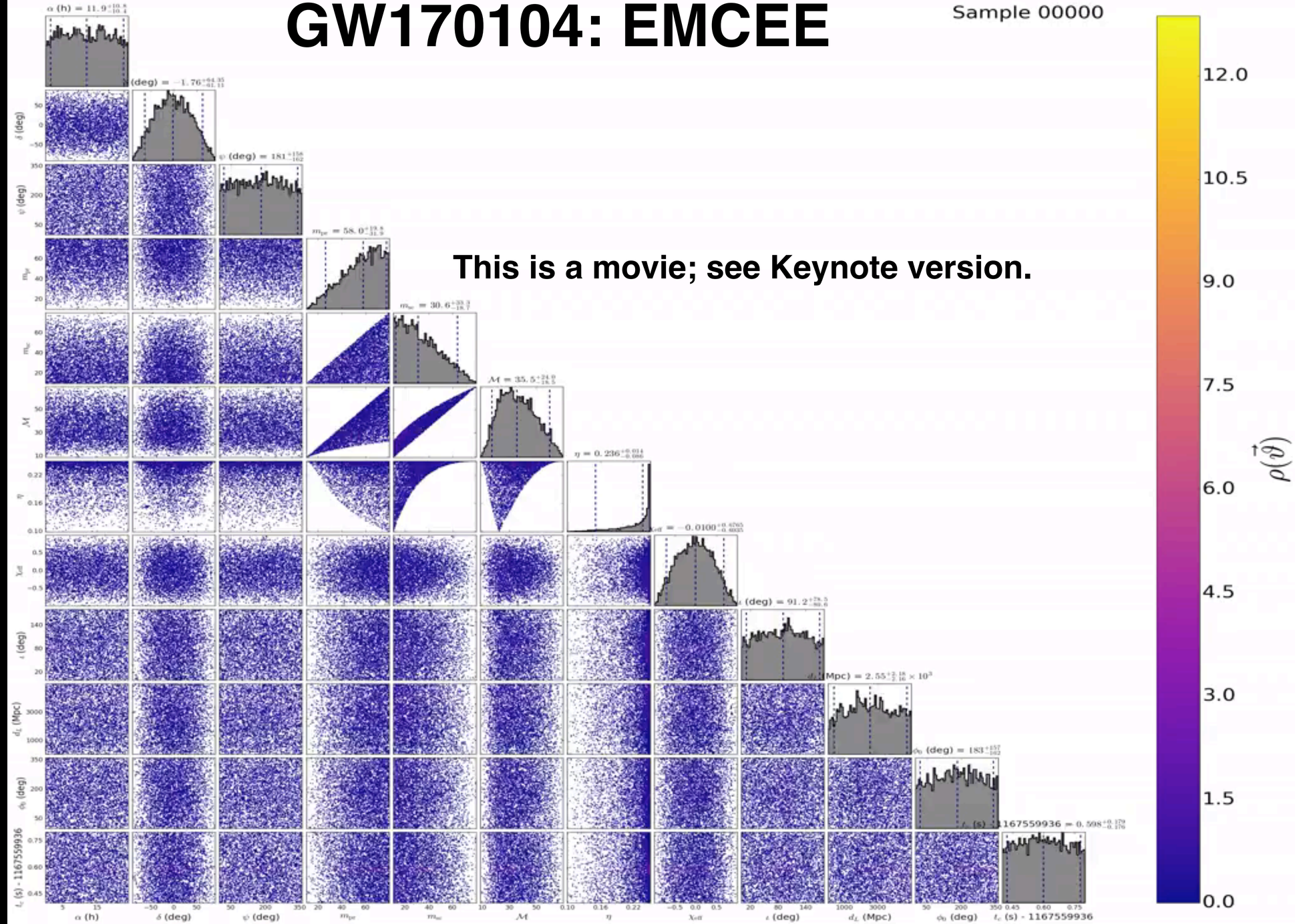
- All MCMC samplers in PyCBC inference use multiple, parallel chains
- emcee, emcee_pt are “ensemble” samplers – the chains share information
- Require some **burn-in** period before the chains resemble the posterior
- Successive samples in a chain are not independent
- Need to calculate the **autocorrelation time** (ACT) of the chains
- ACT gives the number of iterations needed to acquire a new independent sample
- Thinning the chains by the ACT yields independent samples of the posterior

EMCEE

- `emcee`: an ensemble MCMC package
  - Documentation: http://dfm.io/emcee/current/
  - Paper: https://arxiv.org/abs/1202.3665
- Open source, all python; can be installed with pip
- Popular in astronomy

GW170104: EMCEE

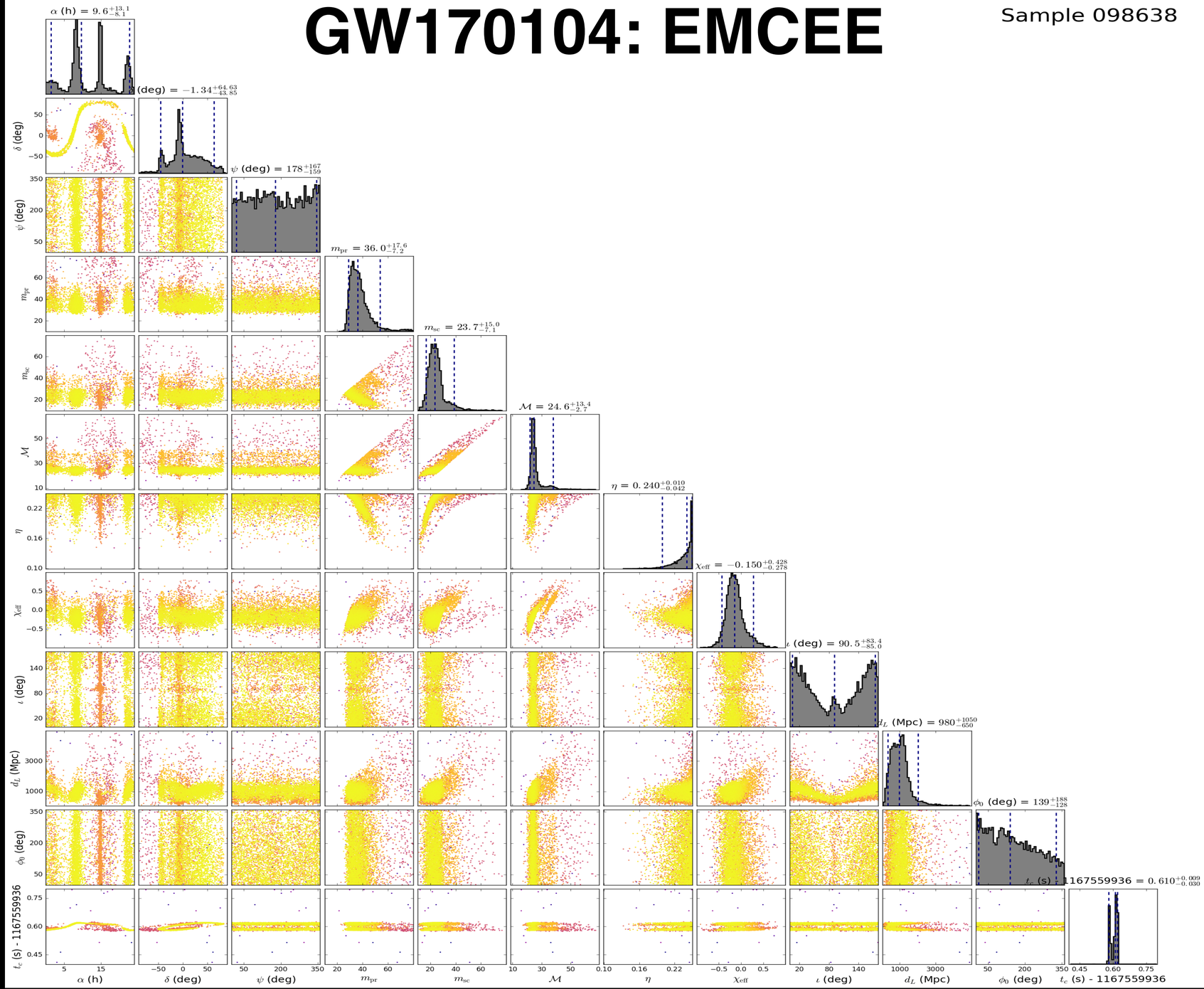
Sample 00000



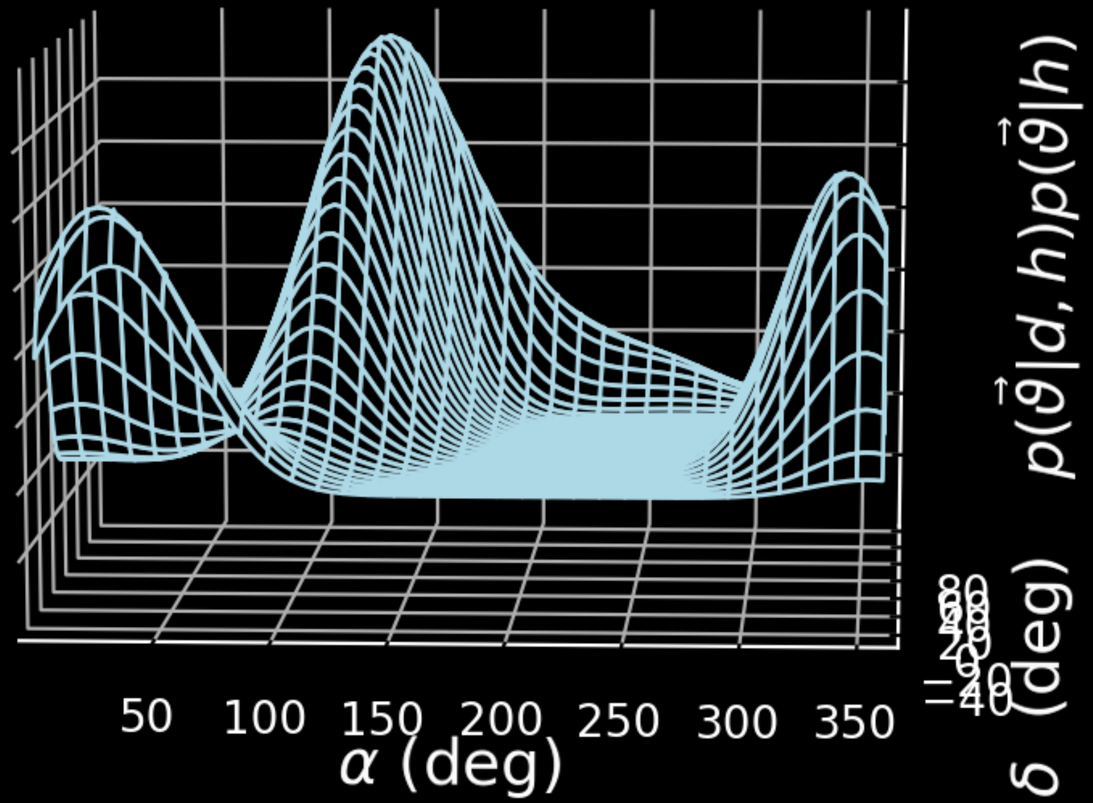
This is a movie; see Keynote version.

GW170104: EMCEE

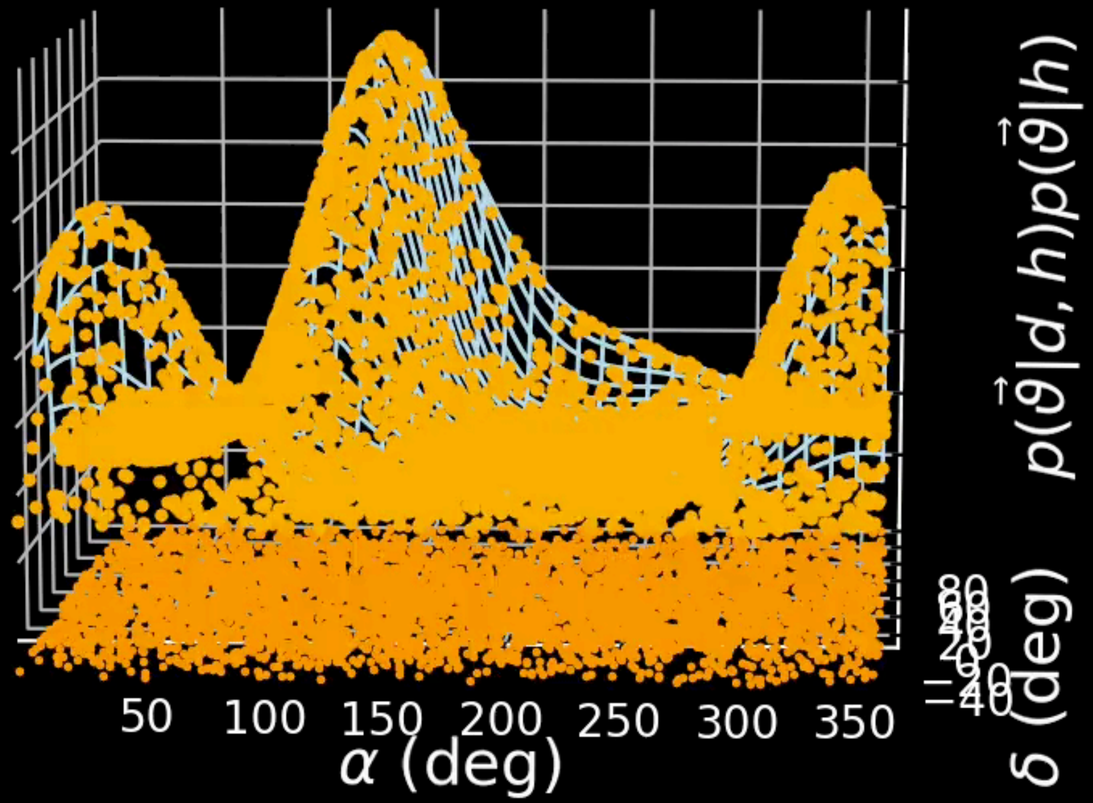
Sample 098638



GW170104: EMCEE



GW170104: EMCEE



This is a movie; see Keynote version.

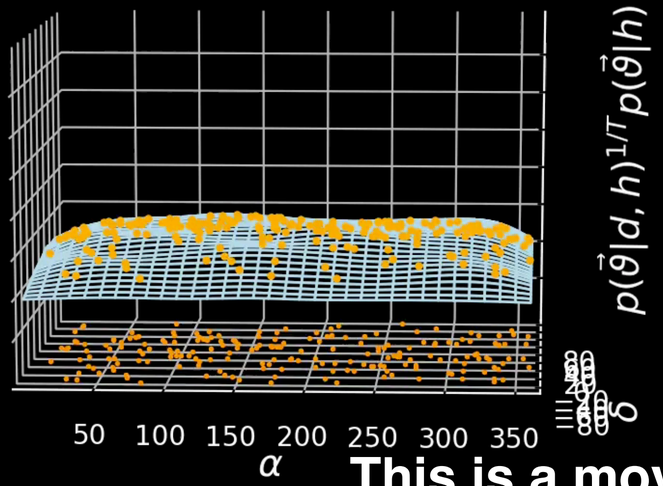
EMCEE PT

- `emcee` struggles to converge for GW problems
- **`emcee_pt`**:* **parallel tempered** version of emcee
- Uses same proposal distribution as emcee, but with parallel tempering

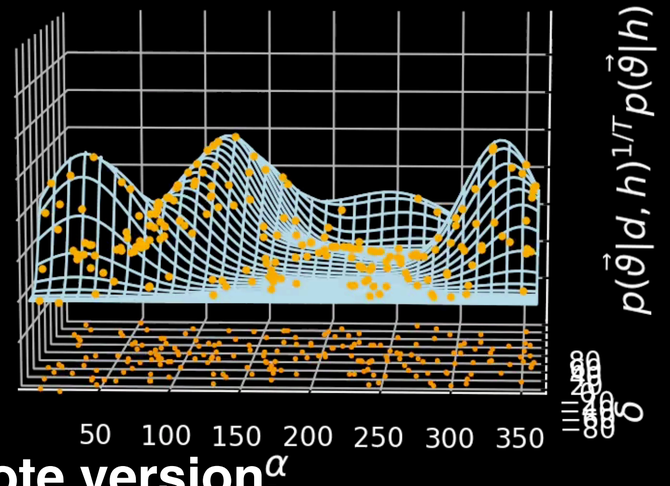
*Note: Current version of emcee_pt is called “ptemcee” & lives in a separate repository. We will probably add support for that in a future update.

PARALLEL TEMPERED MCMC

$T = 10^5$

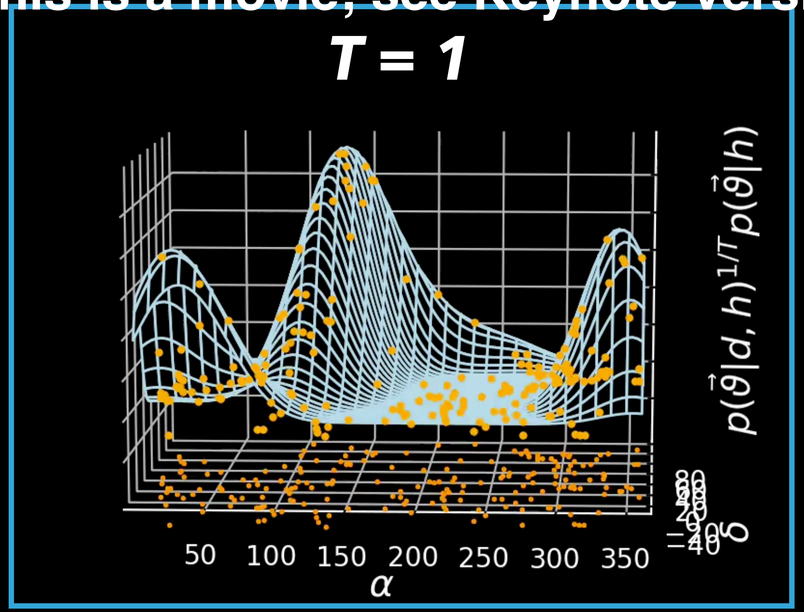


$T = 3.5$



This is a movie; see Keynote version.

$T = 1$



BURN IN TESTS

How do you know when your sampler is converged?

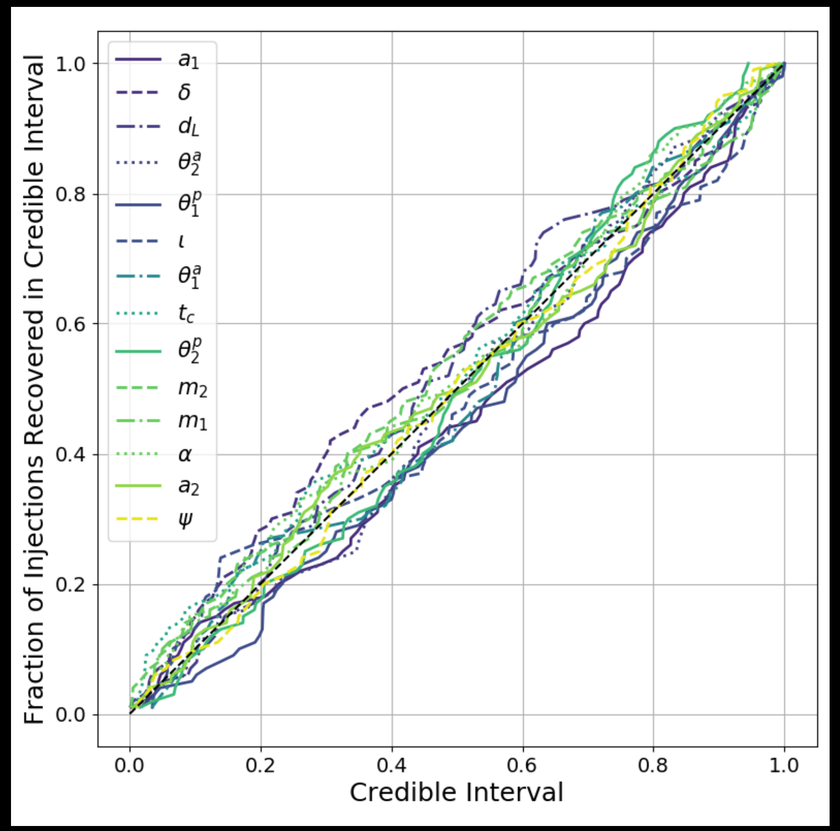
- No single universal test works for all samplers and problems
- Multiple different tests available in PyCBC Inference; see the burn_in module
  - Note: these are for MCMC tests; nested samplers have their own convergence tests
- For emcee_pt, the `nacl` & `max_posterior` tests generally work well
  - `nacl`: checks that the sampler has run for several autocorrelation times
  - `max_posterior`: checks that all chains have sampled a point close to the max posterior

PERCENTILE - PERCENTILE TEST

How do you know the sampler is producing correct results?

- Add signals to realizations of Gaussian noise
- Run PE on each signal, produce marginal posteriors for each parameter
- Test: for each parameter, do X% of the injected values fall within the X% credible interval?

emcee_pt

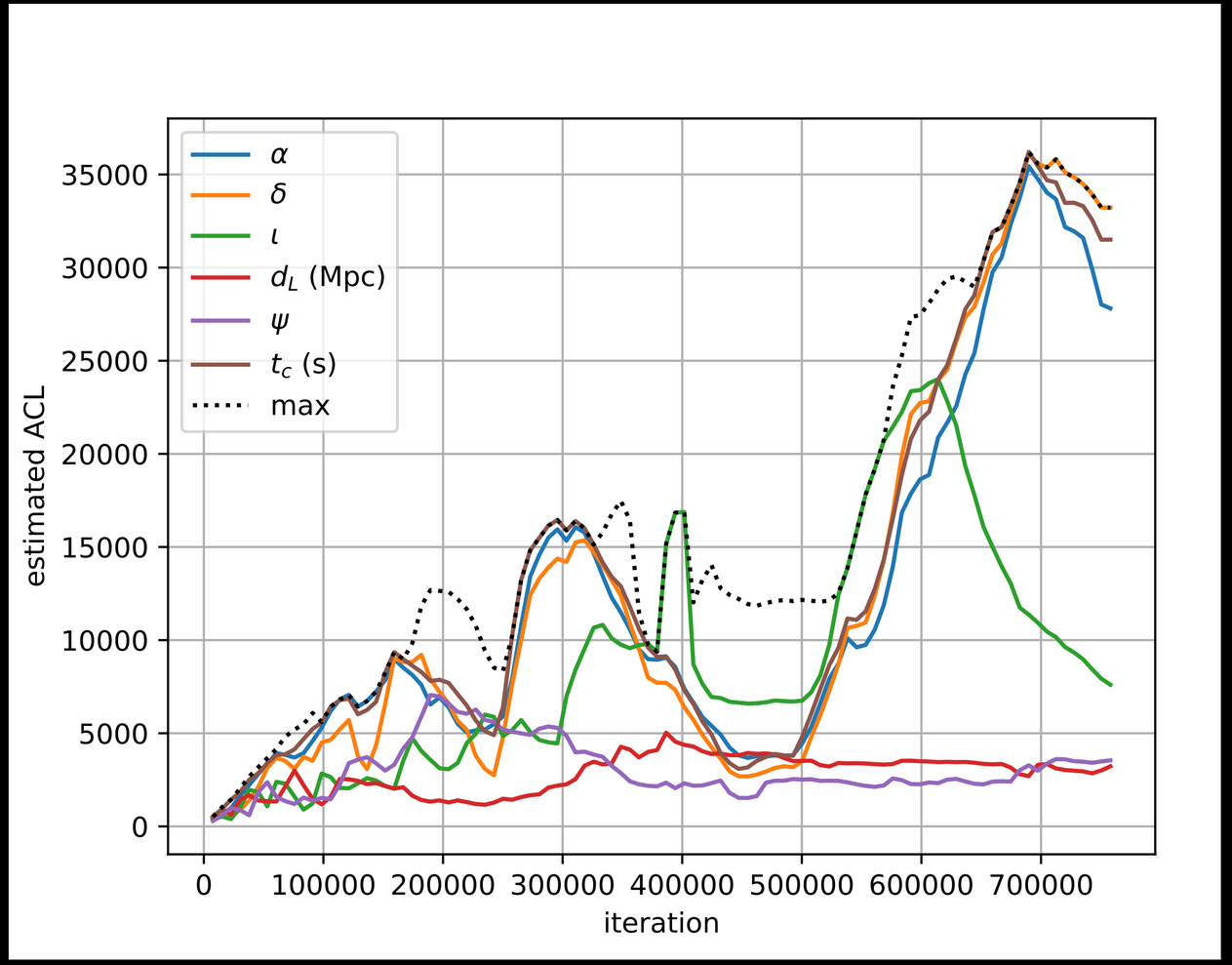


C. Biwer, **C. Capano**, et al., arXiv:1807.10312

Probability of obtaining this graph if emcee_pt provides unbiased parameter estimates: **70% (pass)**

EMCEE PT ISSUES

- Can be slow to converge
- ACT can grow unbounded –> number of effective samples hits a limit (~8 per chain)
- Acquire more samples by running multiple instances, combine results



SAMPLERS

- We offer multiple samplers in PyCBC as each has different strengths & weaknesses.
- MCMC:
  - pros: have been tested extensively; can use any prior with them
  - cons: slow to converge, require parallel tempering
- Nested samplers:
  - pros: provide accurate evidence; fast! (at least dynesty is)
  - cons: only certain priors can be used with them
- **Choose the sampler that best suits your problem!**

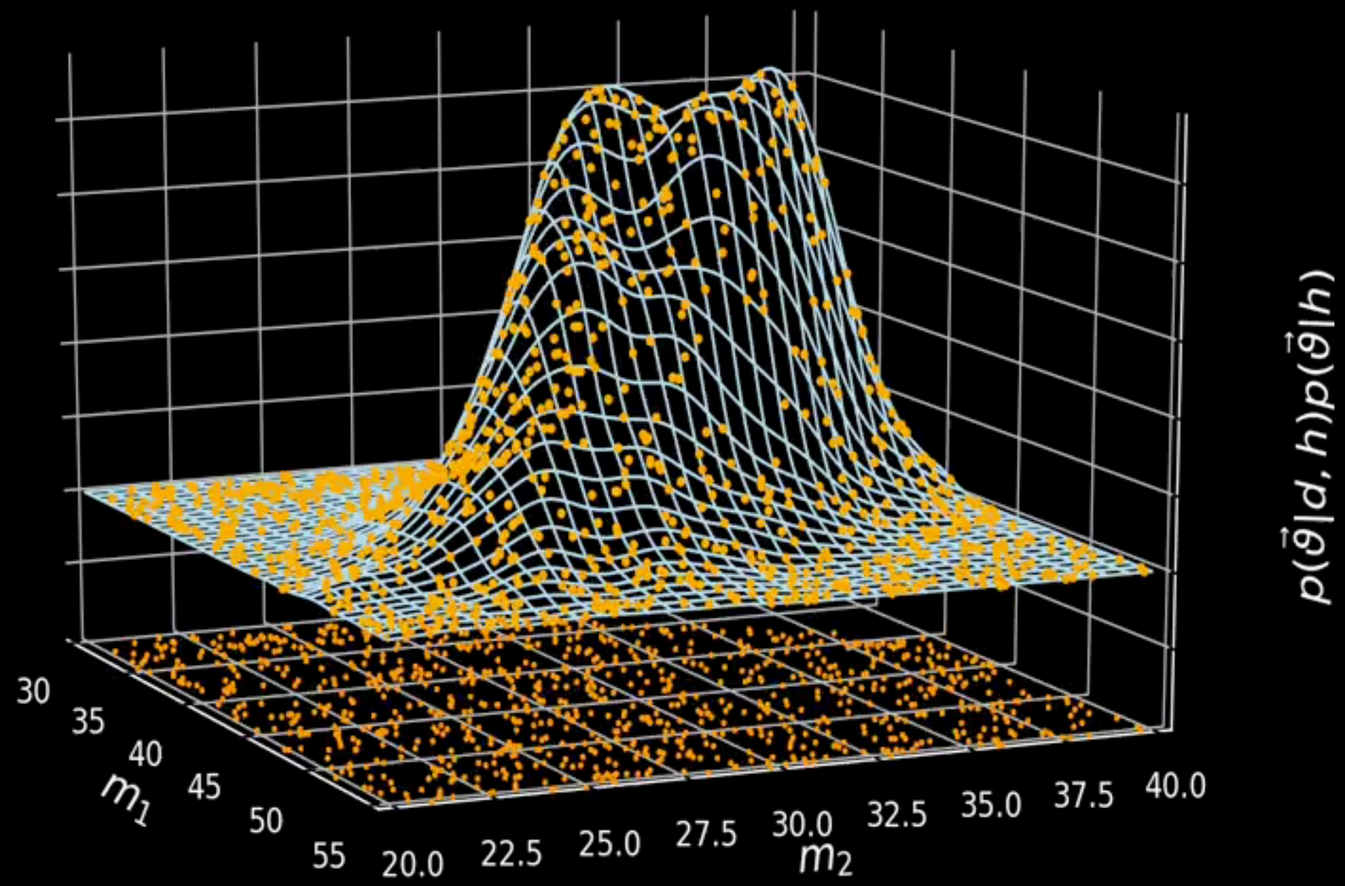
LOOKING AHEAD

- In the tutorials we will go more in-depth into the topics introduced here
  1. Learn about models and do parameter estimation in a notebook
  2. PyCBC Inference code overview
  3. Using PyCBC Inference on an analytic test model
  4. How to estimate the parameters of a binary merger
  5. How to load, plot, and manipulate results
  6. Advanced configuration settings
  7. How to add custom waveform models to PyCBC
- We hope you find this workshop to be engaging and instructive!

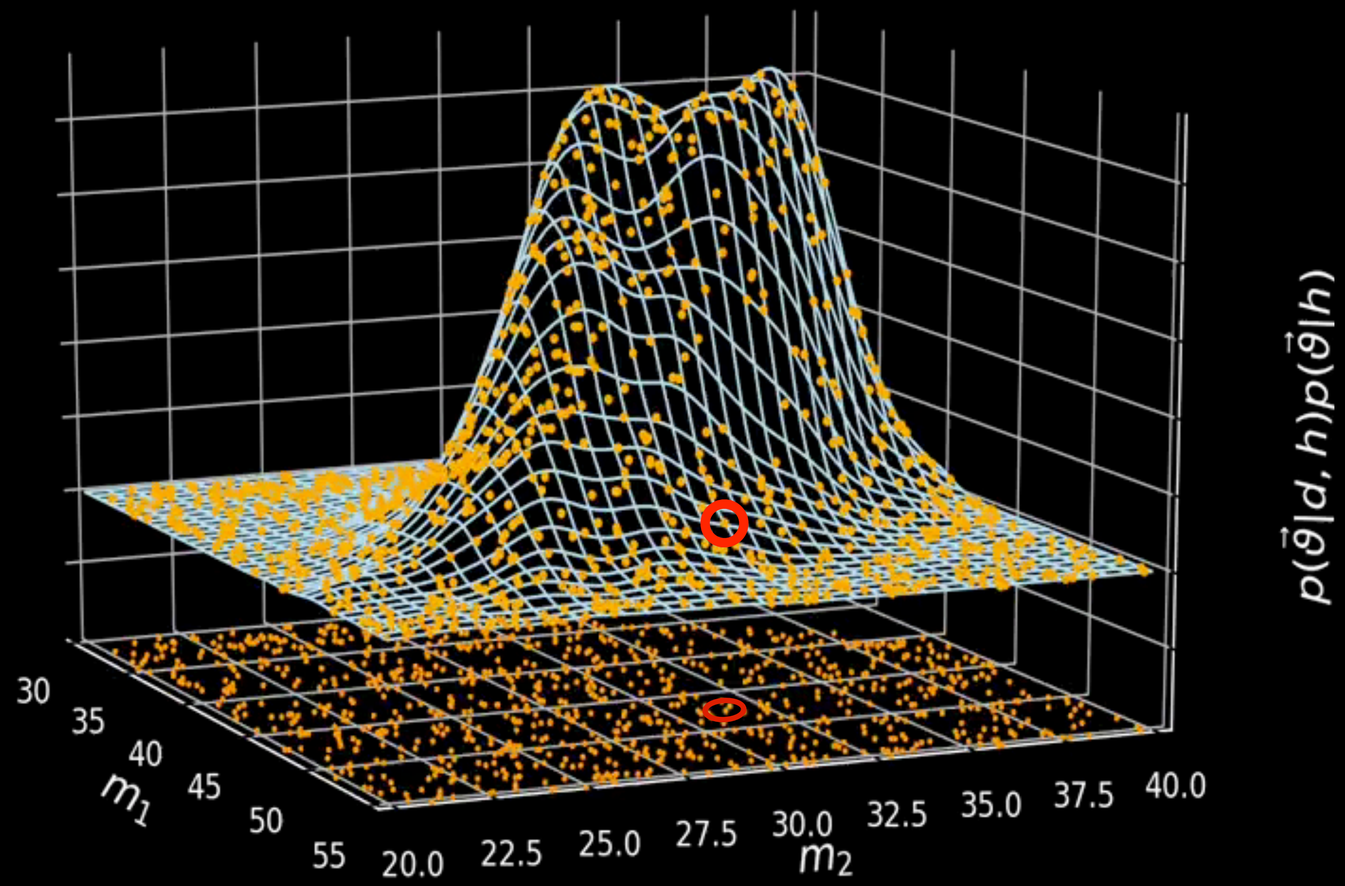
THANK YOU!

EXTRAS

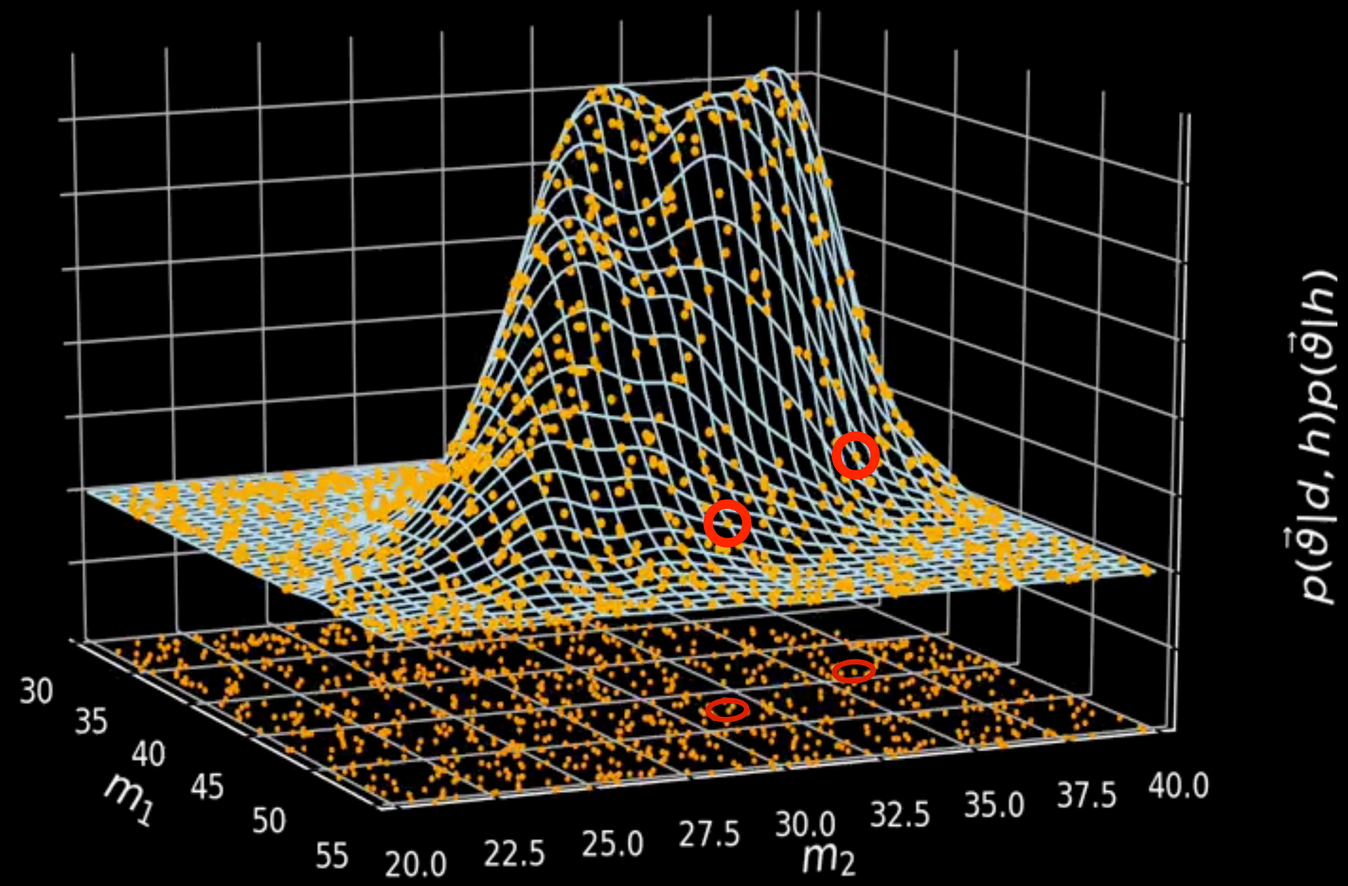
emcee uses a
“stretch” proposal



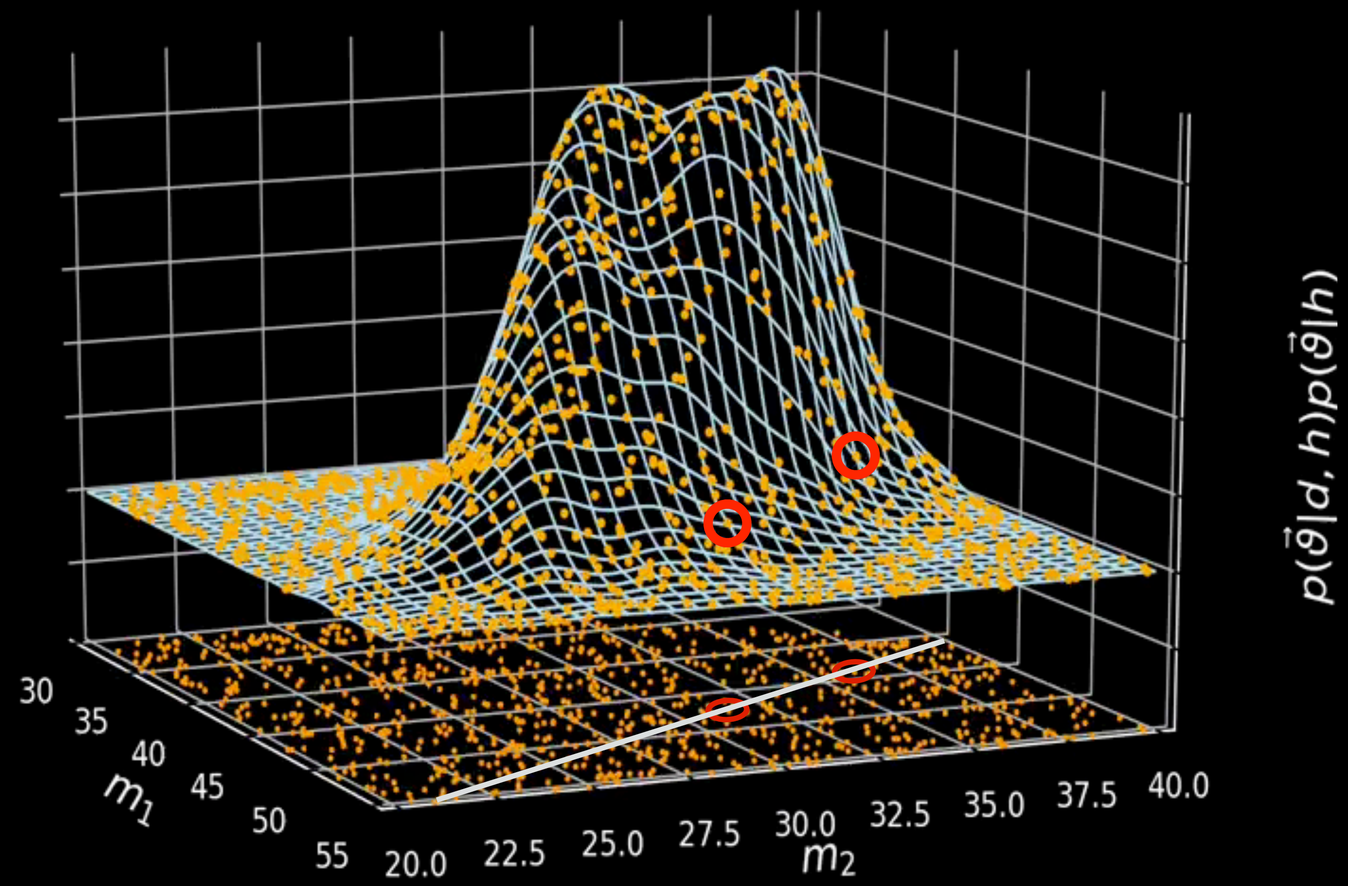
emcee uses a
“stretch” proposal



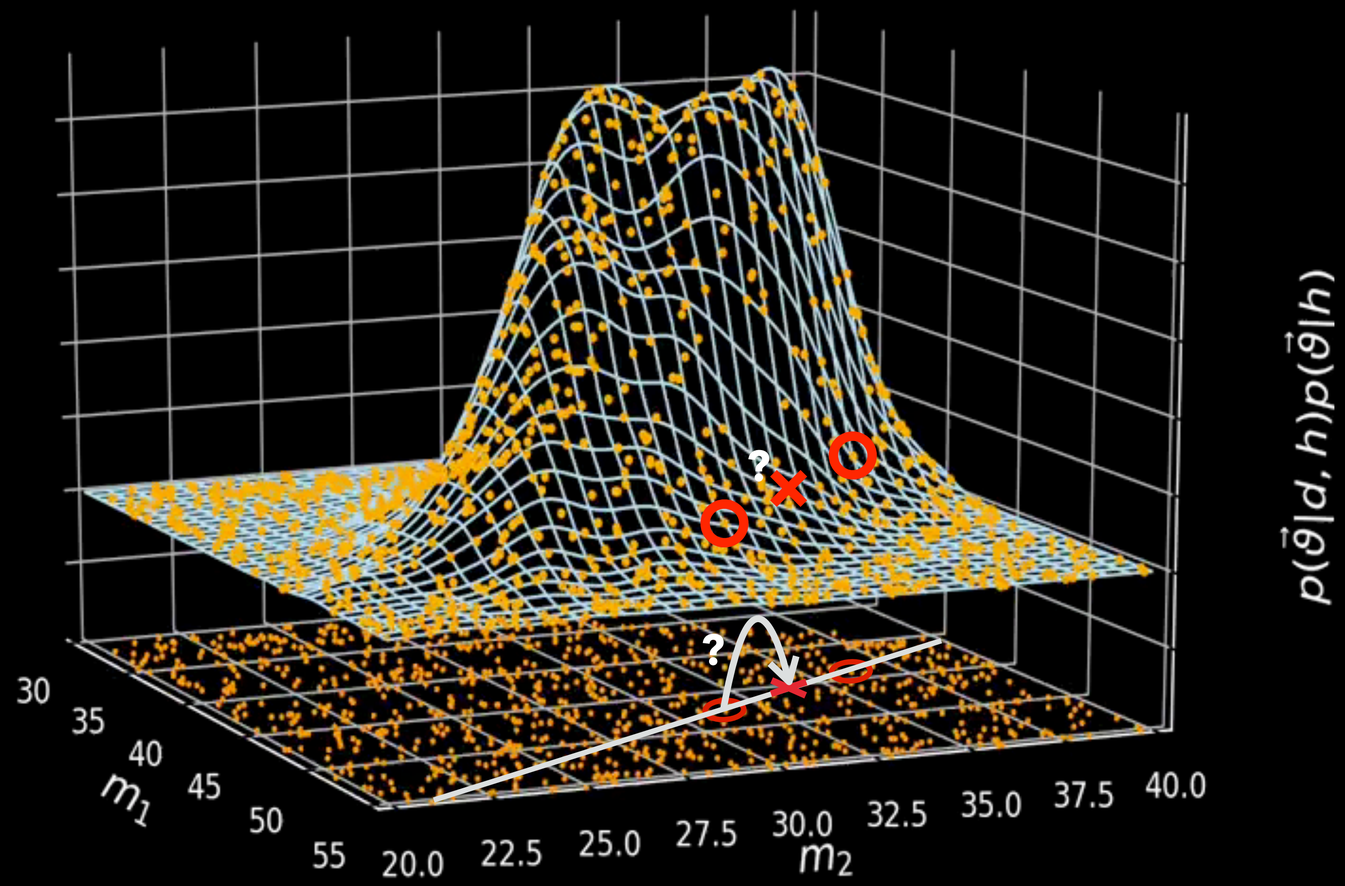
emcee uses a
“stretch” proposal



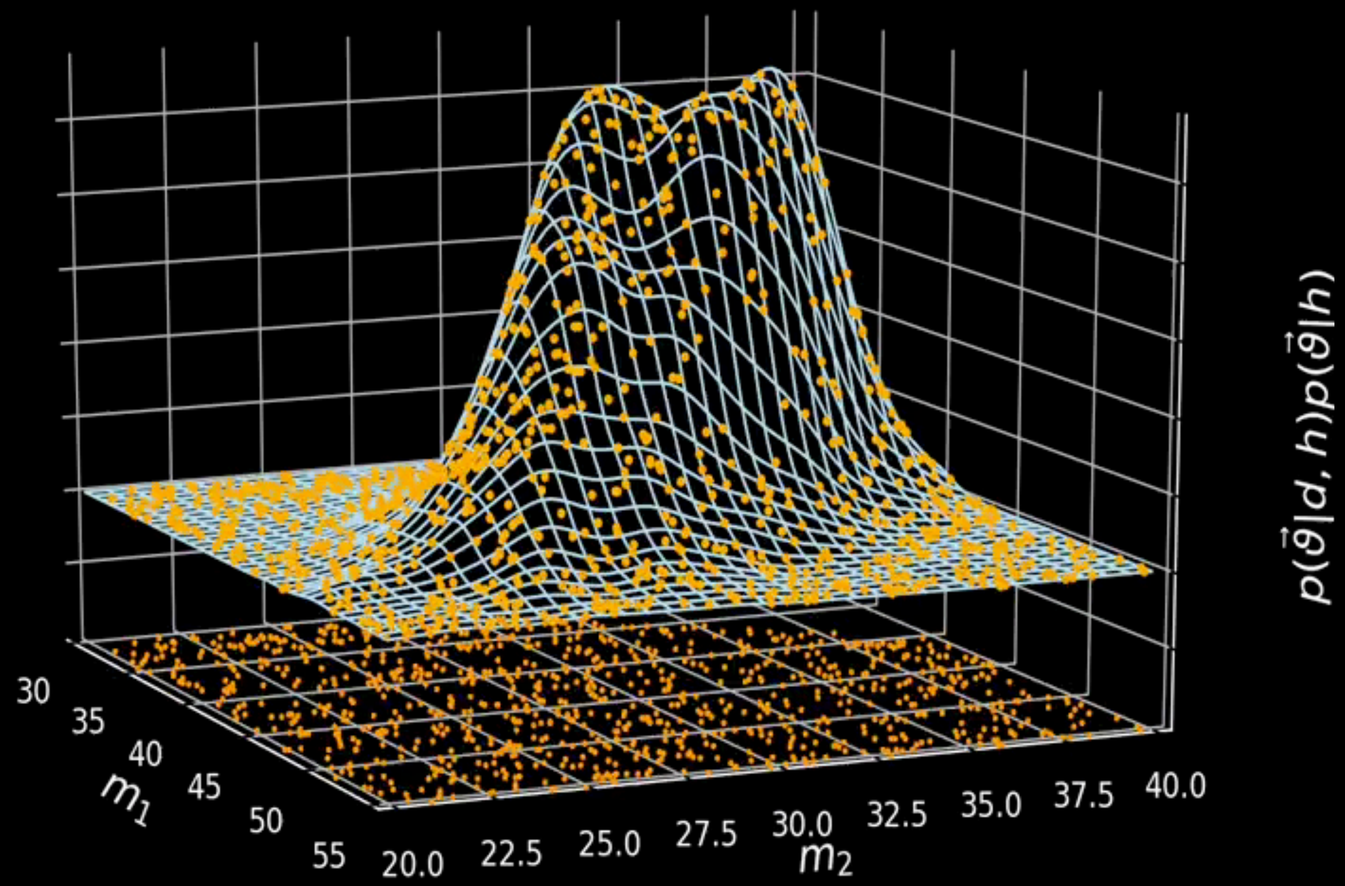
emcee uses a
“stretch” proposal



emcee uses a
“stretch” proposal



emcee uses a
“stretch” proposal



STANDARD LIKELIHOOD DERIVATION

Have some data: $\mathbf{s} = \{s_0, s_1, \cdots, s_N\}$

STANDARD LIKELIHOOD DERIVATION

Have some data: $\mathbf{s} = \{s_0, s_1, \cdots, s_N\}$

Noise hypothesis - assume Gaussian noise:

$$p(\mathbf{s}|n) = \frac{\exp\left[-\frac{1}{2}\mathbf{s}^\mathsf{T}\boldsymbol{\Sigma}^{-1}\mathbf{s}\right]}{\sqrt{(2\pi)^N \det \boldsymbol{\Sigma}}}$$

STANDARD LIKELIHOOD DERIVATION

Have some data: $\mathbf{s} = \{s_0, s_1, \cdots, s_N\}$

Noise hypothesis - assume Gaussian noise:

$$p(\mathbf{s}|n) = \frac{\exp\left[-\frac{1}{2}\mathbf{s}^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}\mathbf{s}\right]}{\sqrt{(2\pi)^N \det \boldsymbol{\Sigma}}}$$

Assume zero-mean, wide-sense stationary (WSS):

$$\boldsymbol{\Sigma} = \frac{1}{2}\begin{bmatrix}
R_{ss}[0] & R_{ss}[1] & \cdots & R_{ss}[\Delta_{\max}] & & & & & \\
R_{ss}[1] & R_{ss}[0] & \ddots & \ddots & \ddots & & & \mathbf{0} & \\
\vdots & R_{ss}[1] & \ddots & \ddots & \ddots & \ddots & & & \\
R_{ss}[\Delta_{\max}] & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & & \\
 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \\
 & & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & R_{ss}[\Delta_{\max}] \\
 & & & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\
 & \mathbf{0} & & & \ddots & \ddots & \ddots & \ddots & R_{ss}[1] \\
 & & & & & R_{ss}[\Delta_{\max}] & \cdots & R_{ss}[1] & R_{ss}[0]
\end{bmatrix}$$

STANDARD LIKELIHOOD DERIVATION

Have some data: $\mathbf{s} = \{s_0, s_1, \cdots, s_N\}$

Noise hypothesis - assume Gaussian noise:

$$p(\mathbf{s}|n) = \frac{\exp\left[-\frac{1}{2}\mathbf{s}^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}\mathbf{s}\right]}{\sqrt{(2\pi)^N \det \boldsymbol{\Sigma}}}$$

Assume zero-mean, wide-sense stationary (WSS):

$$\boldsymbol{\Sigma} \approx \frac{1}{2}\begin{bmatrix}
R_{ss}[0] & R_{ss}[1] & \cdots & R_{ss}[\Delta_{\max}] & & & R_{ss}[\Delta_{\max}] & \cdots & R_{ss}[1] \\
R_{ss}[1] & R_{ss}[0] & \ddots & \ddots & \ddots & & & \ddots & \vdots \\
\vdots & R_{ss}[1] & \ddots & \ddots & \ddots & \ddots & \mathbf{0} & & R_{ss}[\Delta_{\max}] \\
R_{ss}[\Delta_{\max}] & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & & \\
 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \\
 & & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & R_{ss}[\Delta_{\max}] \\
R_{ss}[\Delta_{\max}] & & \mathbf{0} & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\
\vdots & \ddots & & & \ddots & \ddots & \ddots & \ddots & R_{ss}[1] \\
R_{ss}[1] & \cdots & R_{ss}[\Delta_{\max}] & & & R_{ss}[\Delta_{\max}] & \cdots & R_{ss}[1] & R_{ss}[0]
\end{bmatrix}$$

STANDARD LIKELIHOOD DERIVATION

Eigenvectors: $$\mathbf{U}_{kp} = \frac{1}{\sqrt{N}} \exp\left[-2\pi i k p / N\right]$$

⇒ eigenvalues: $$\lambda_p = \frac{S_n[p]}{2\Delta t}$$

Eigendecomposition yields:

$$\mathbf{s}^\mathsf{T} \boldsymbol{\Sigma}^{-1} \mathbf{s} = 4\Delta f \sum_{p=1}^{N/2-1} \frac{\tilde{s}^*[p]\tilde{s}[p]}{S_n[p]} \equiv \langle \mathbf{s}, \mathbf{s} \rangle$$

Signal hypothesis:

$$\mathbf{s} = \mathbf{n} + \mathbf{h}(\boldsymbol{\vartheta})$$

$$\Rightarrow p(\mathbf{s}|\boldsymbol{\vartheta}, h) \propto \exp\left[-\frac{1}{2}\langle \mathbf{s} - \mathbf{h}(\boldsymbol{\vartheta}), \mathbf{s} - \mathbf{h}(\boldsymbol{\vartheta}) \rangle\right]$$