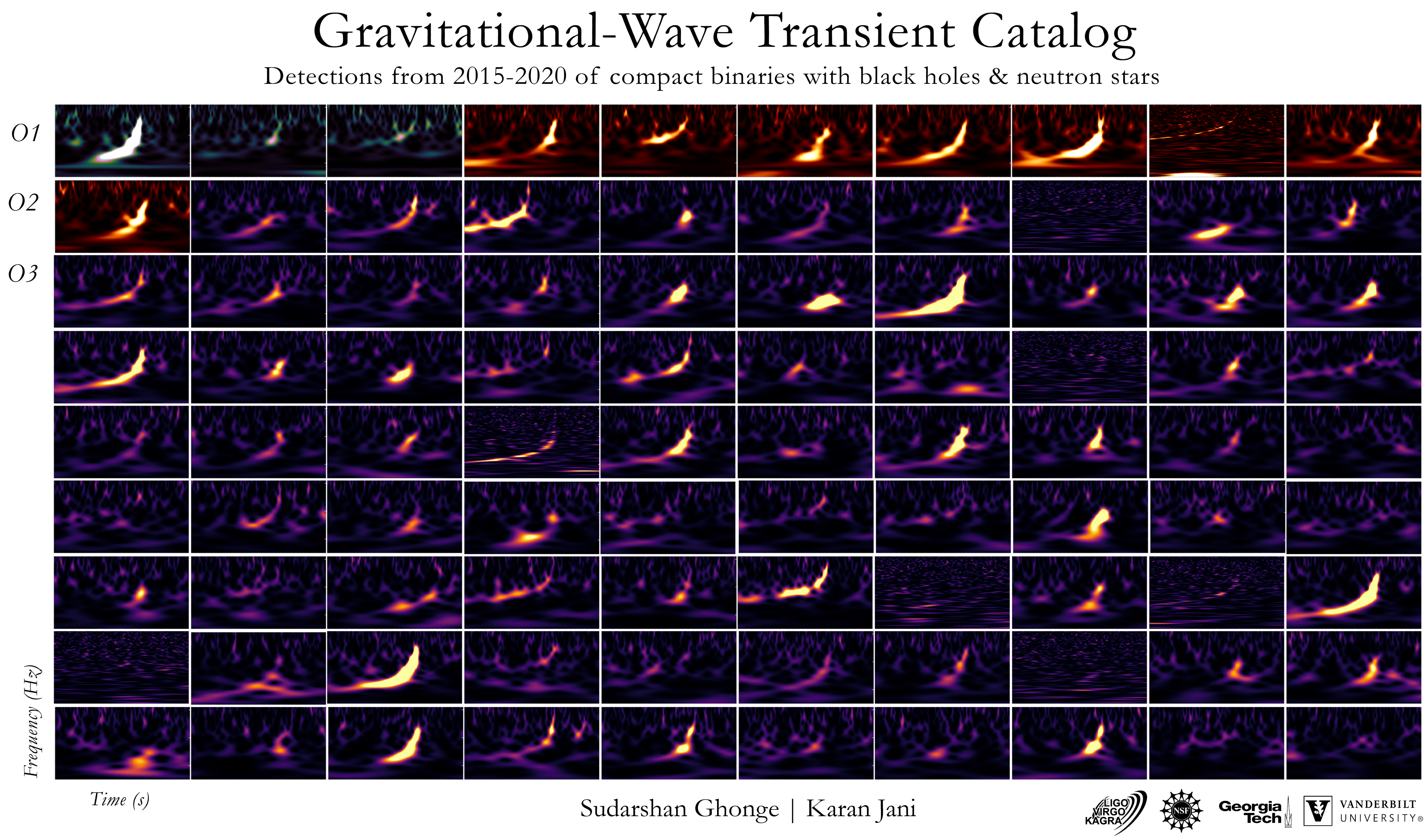
JOHNS HOPKINS UNIVERSITY

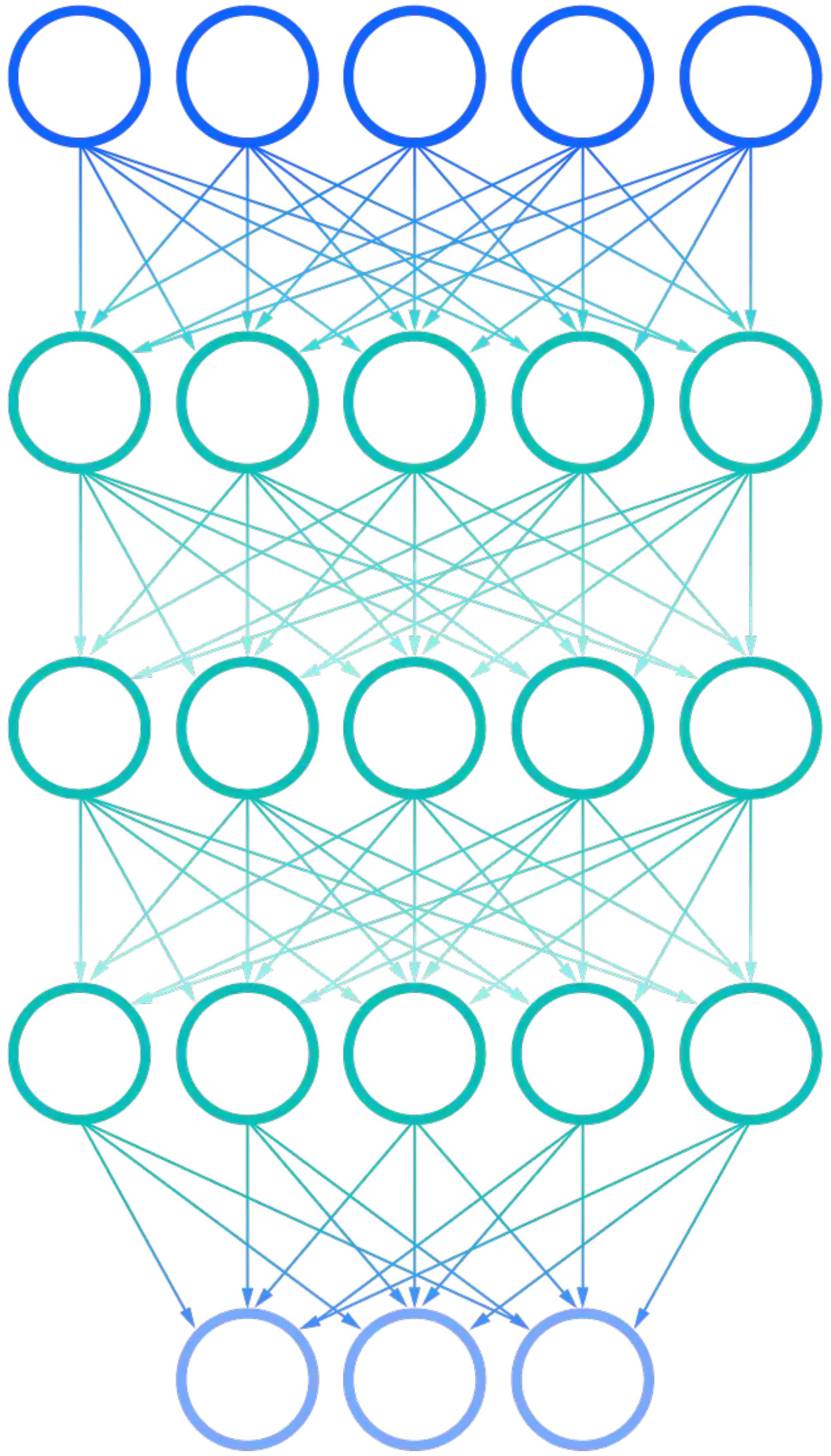
Enhancing Gravitational Wave Science with Differentiable Models and New Physics Searches

Thomas Edwards | King College London | 16th June 2023

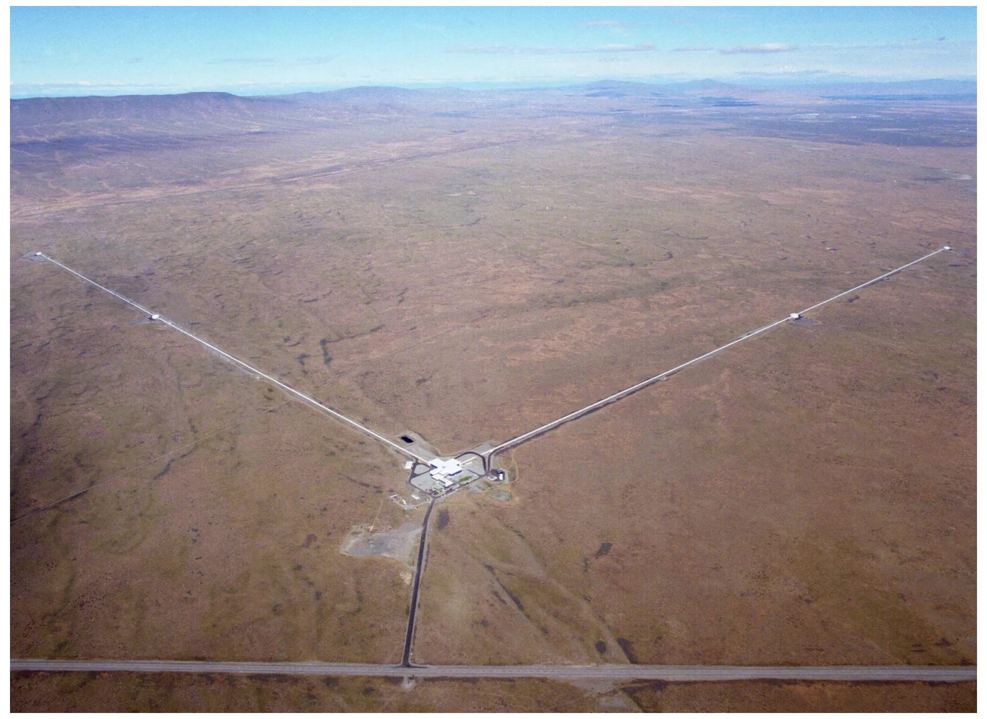
2302.05329
2302.05333
2306.00050
+ 2 upcoming papers

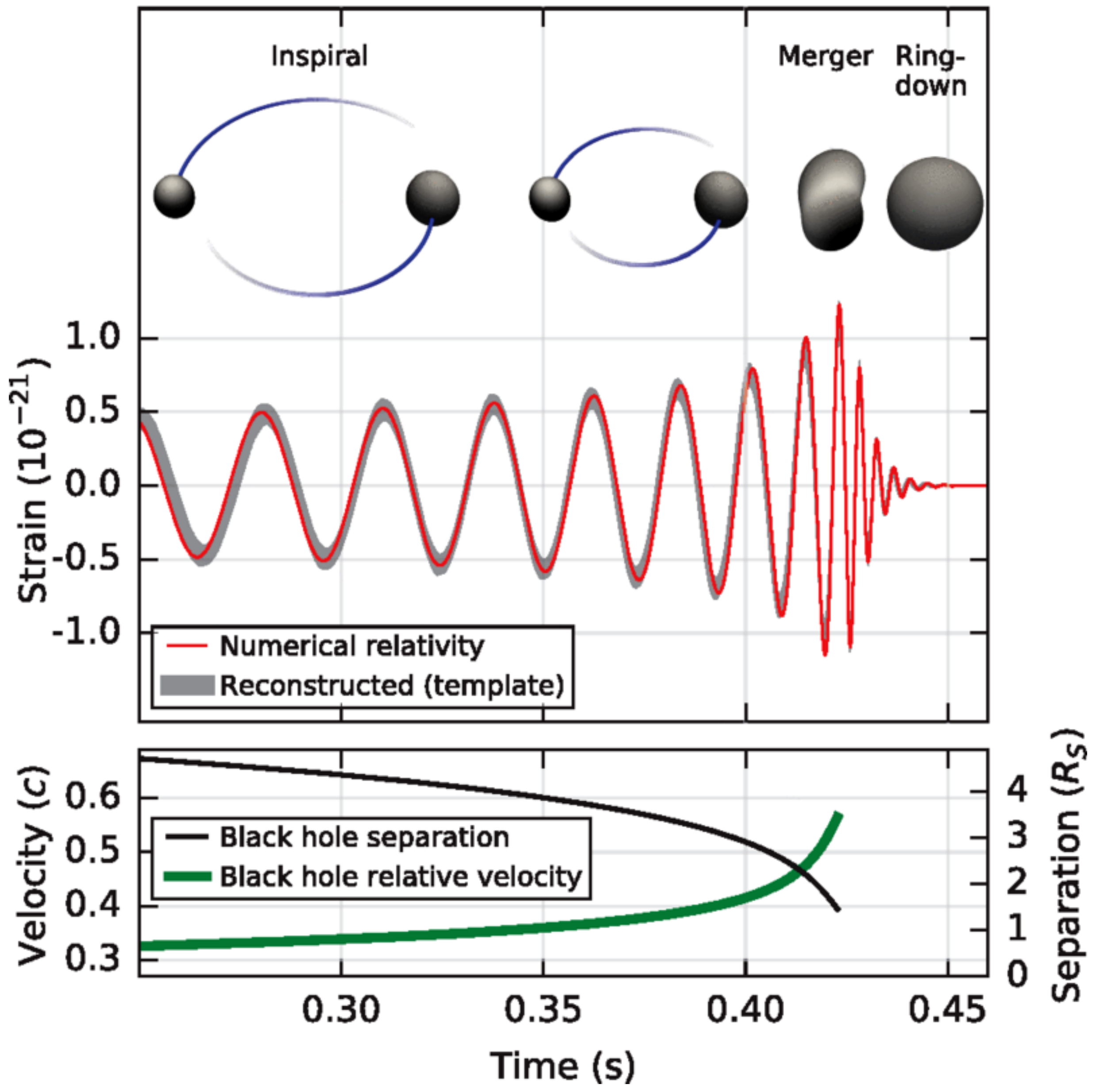
Gravitational Wave + ML Revolution



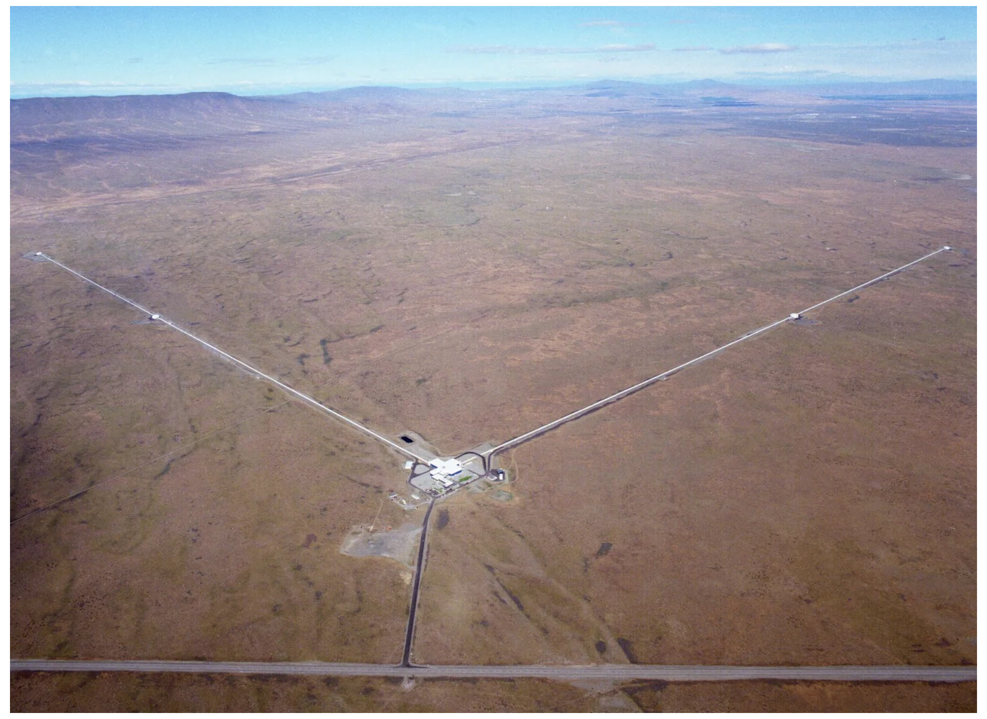


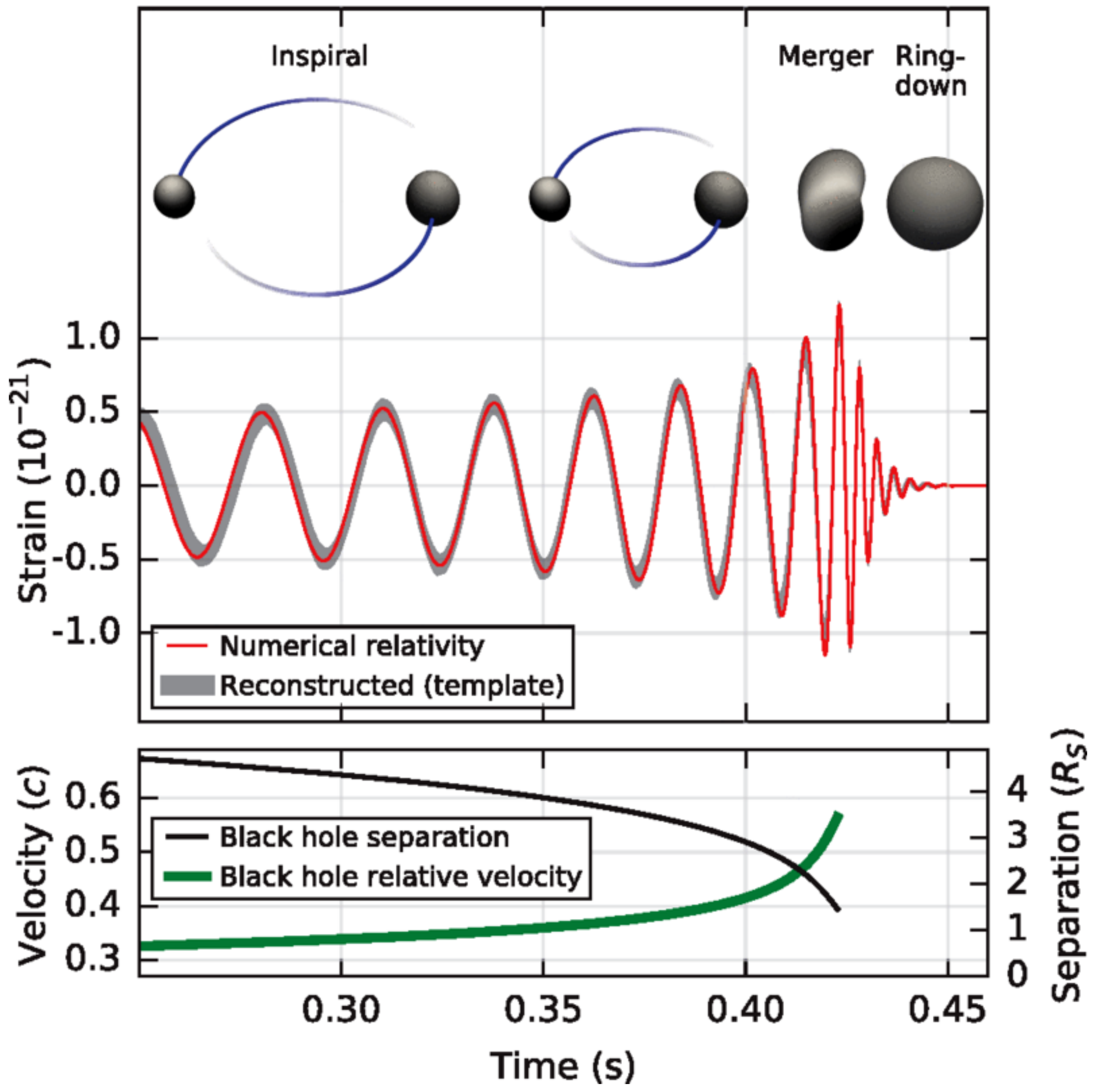
Gravitational Wave Analysis

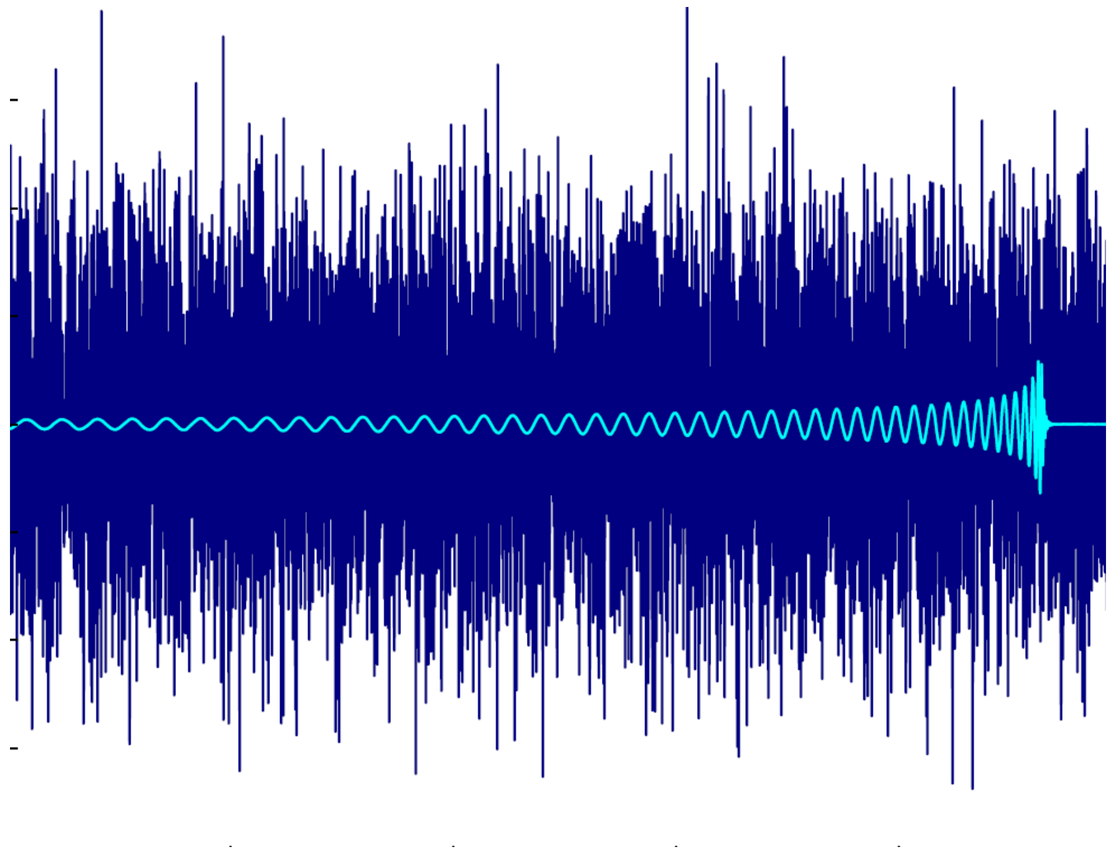




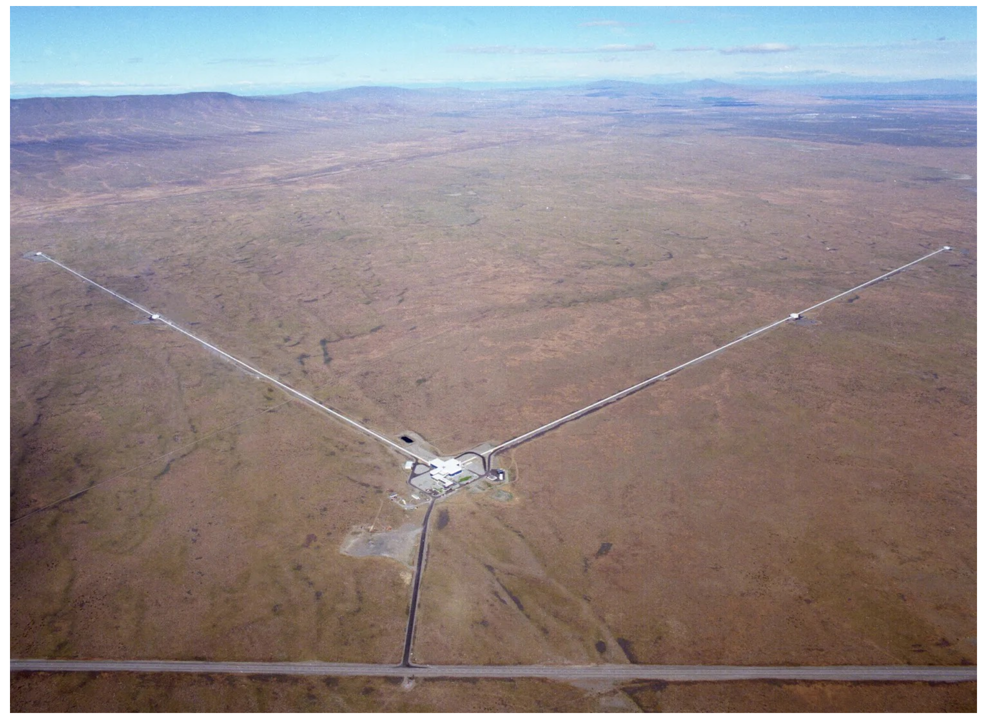
Gravitational Wave Analysis

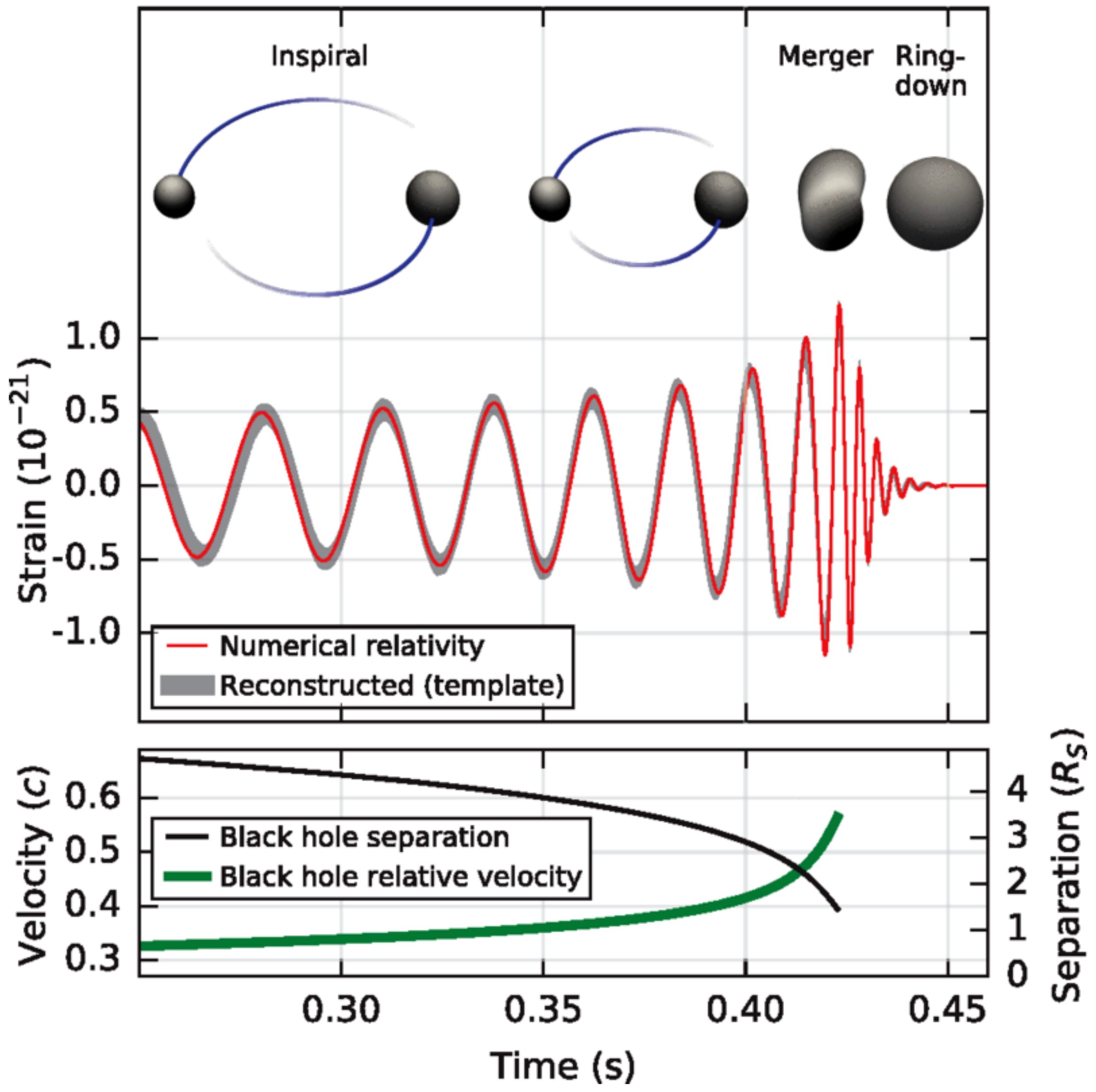


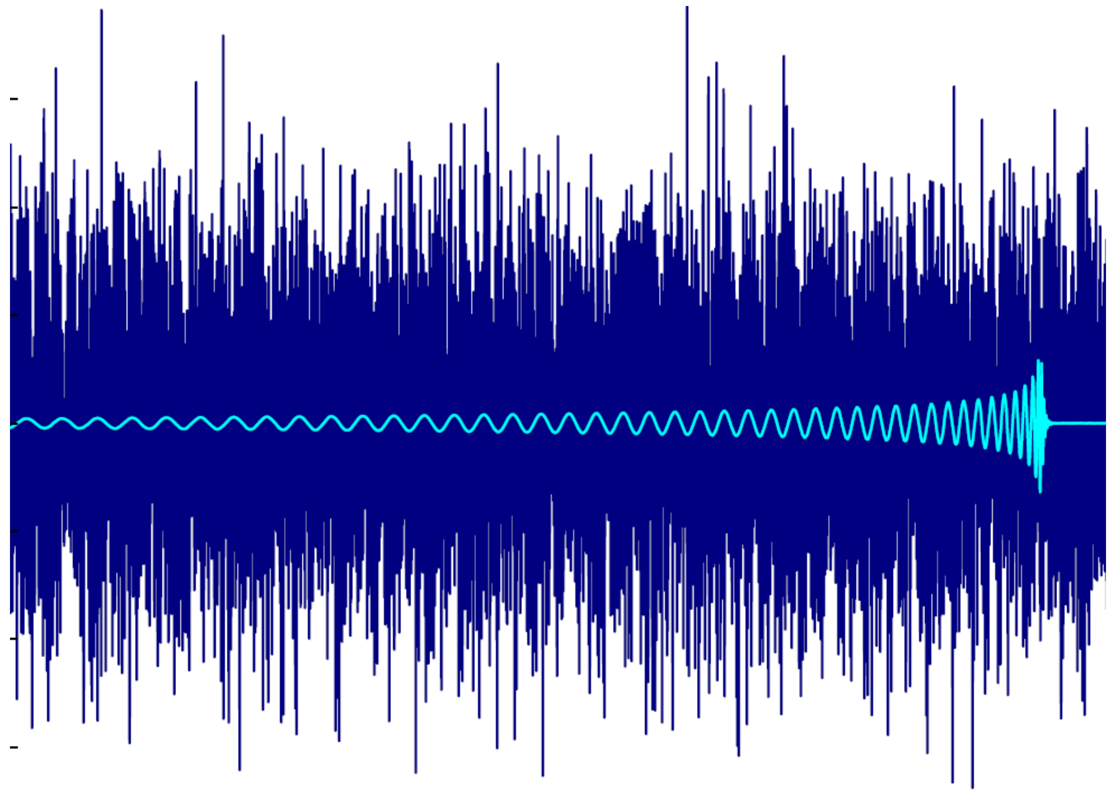


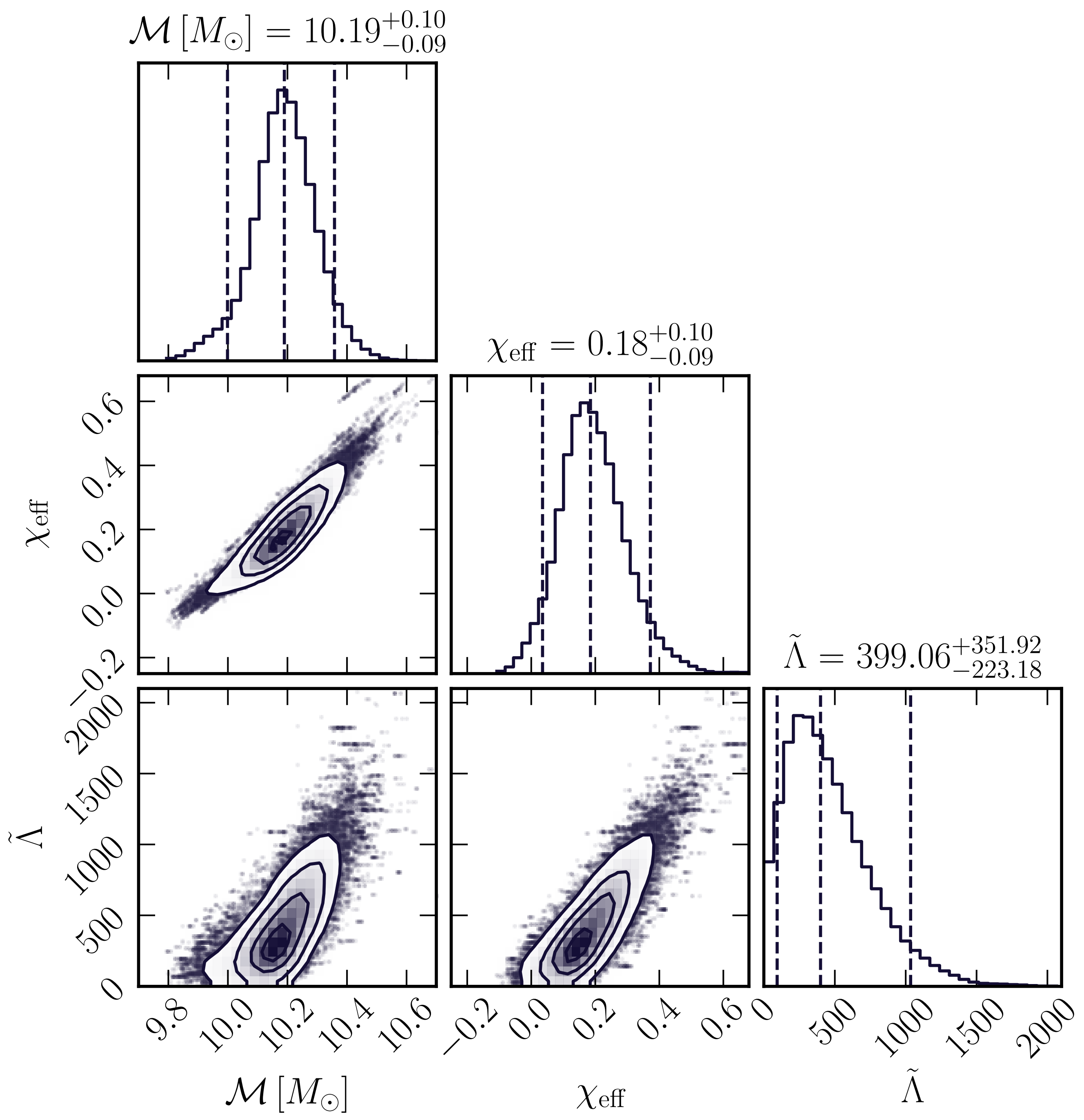


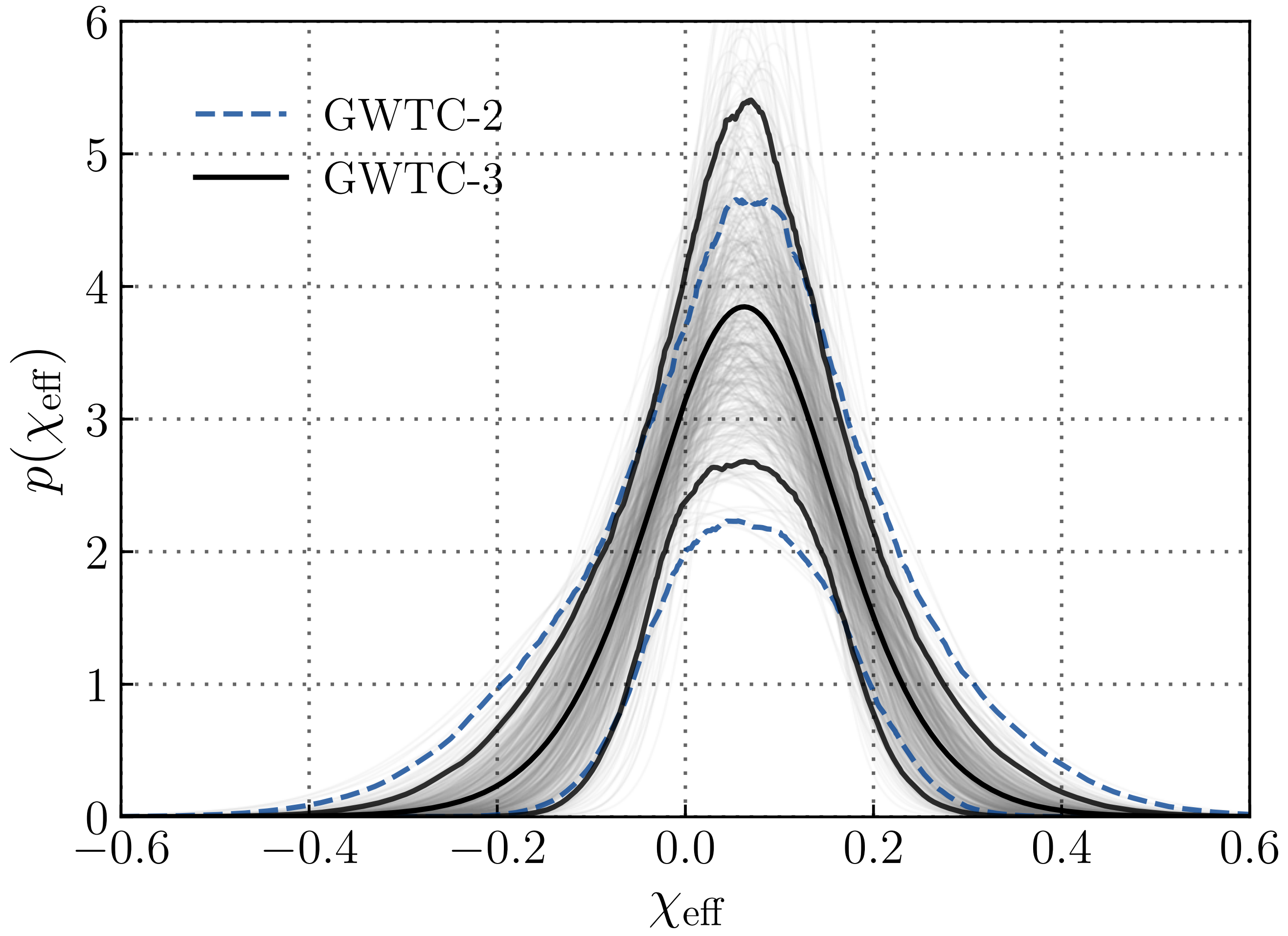
Gravitational Wave Analysis



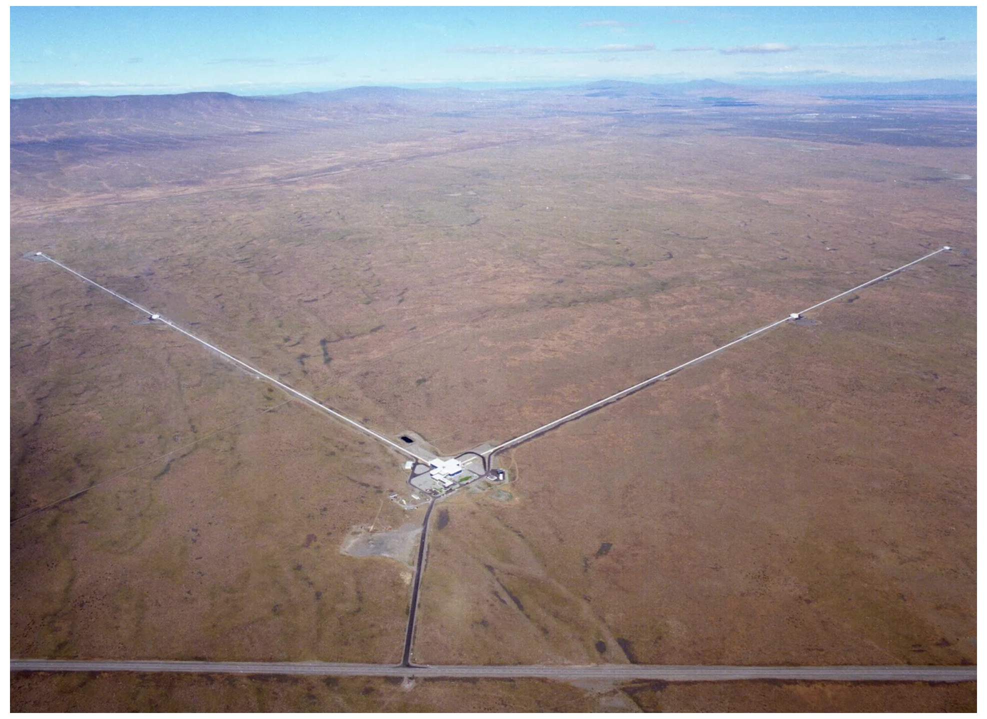


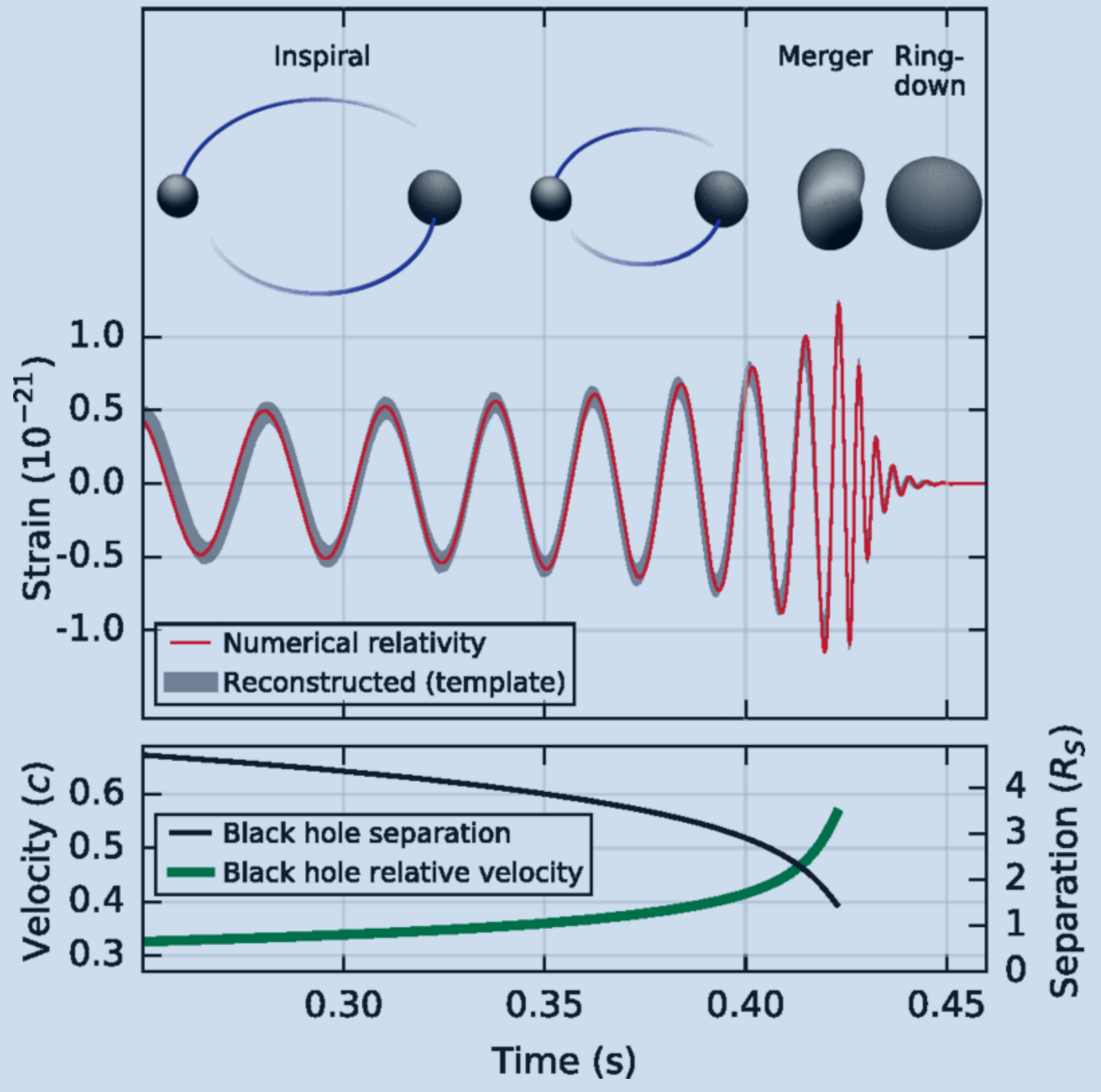


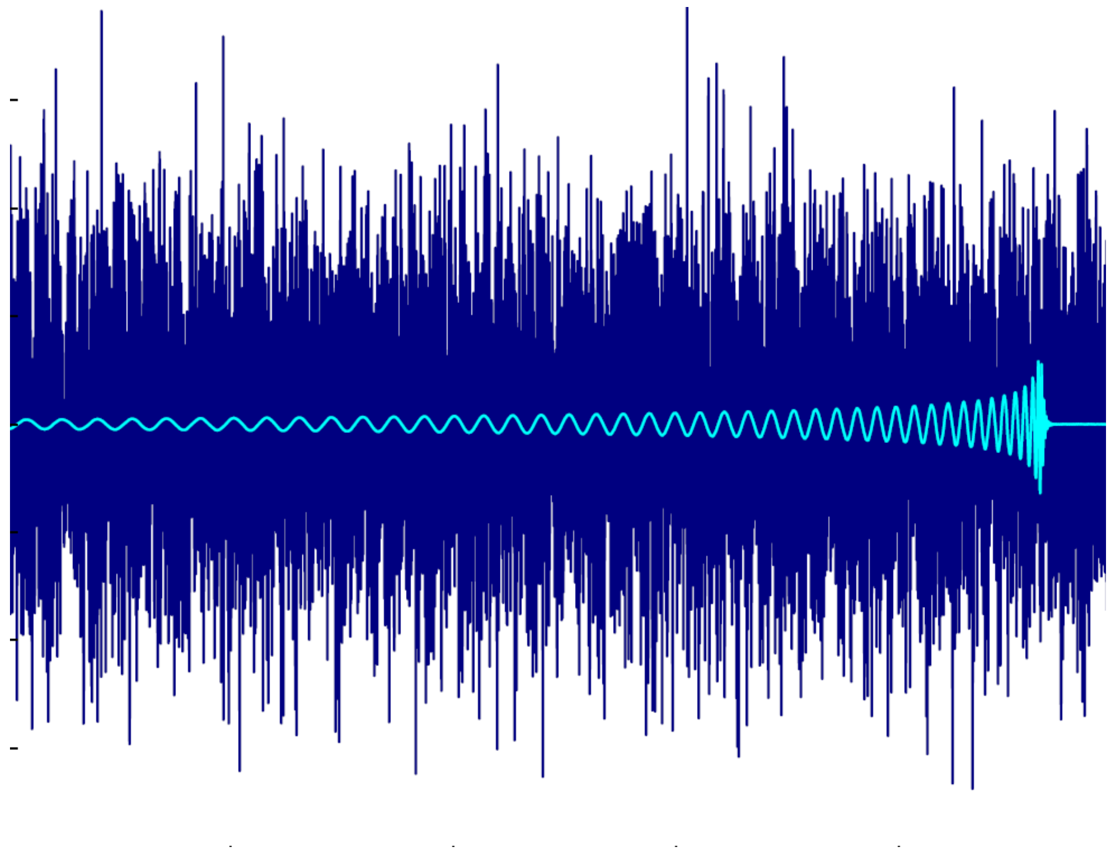


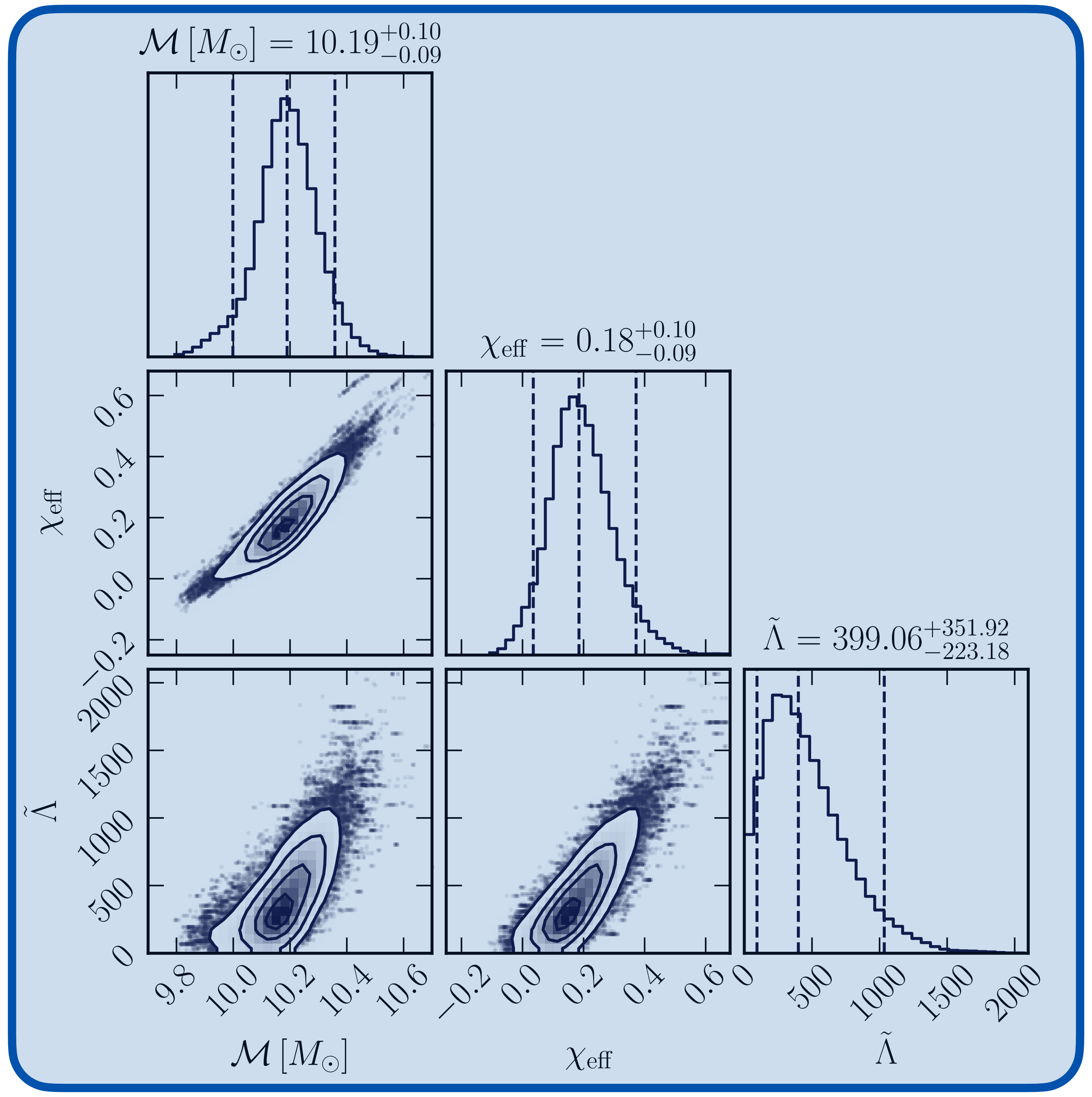


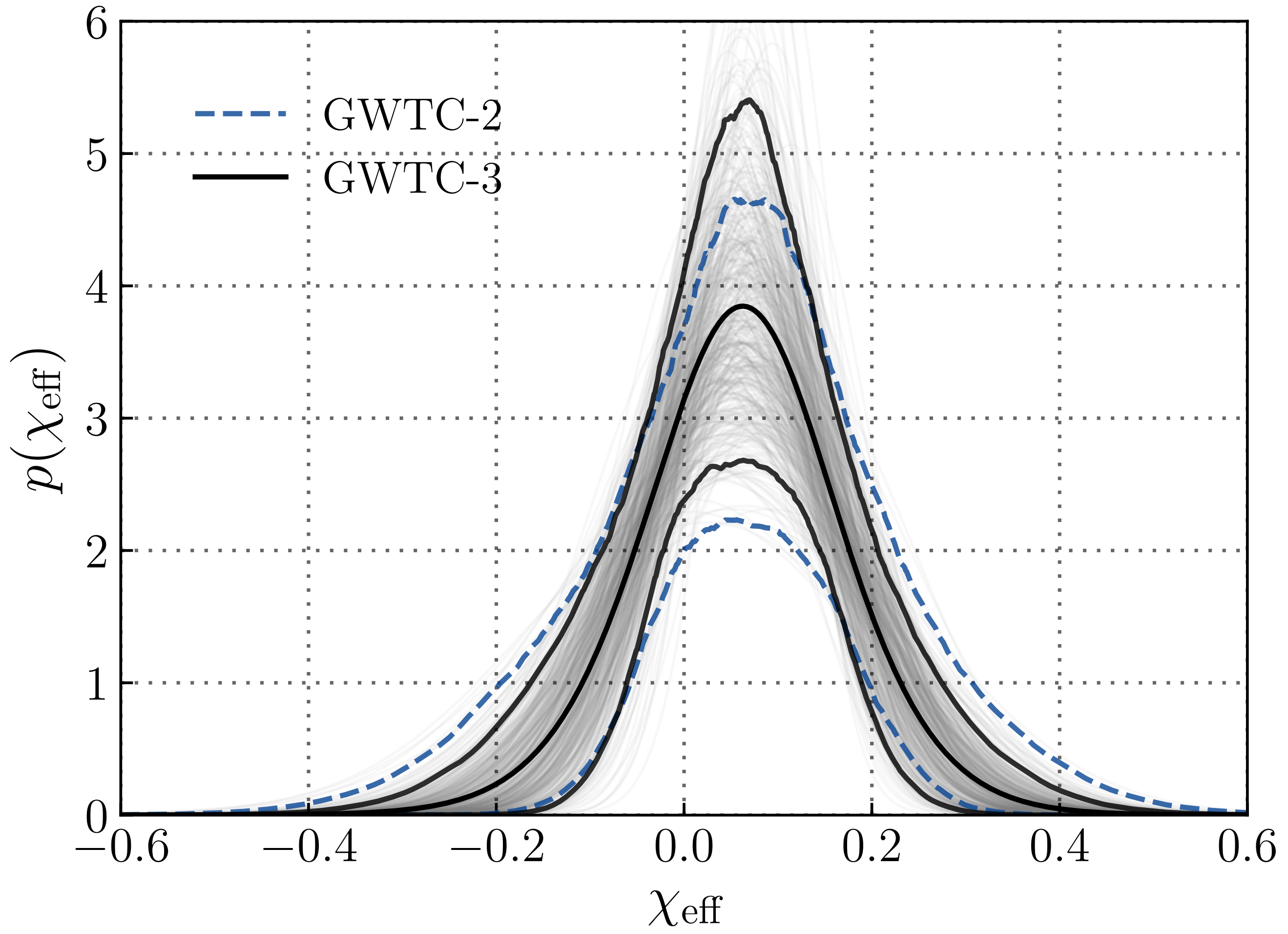
Gravitational Wave Analysis



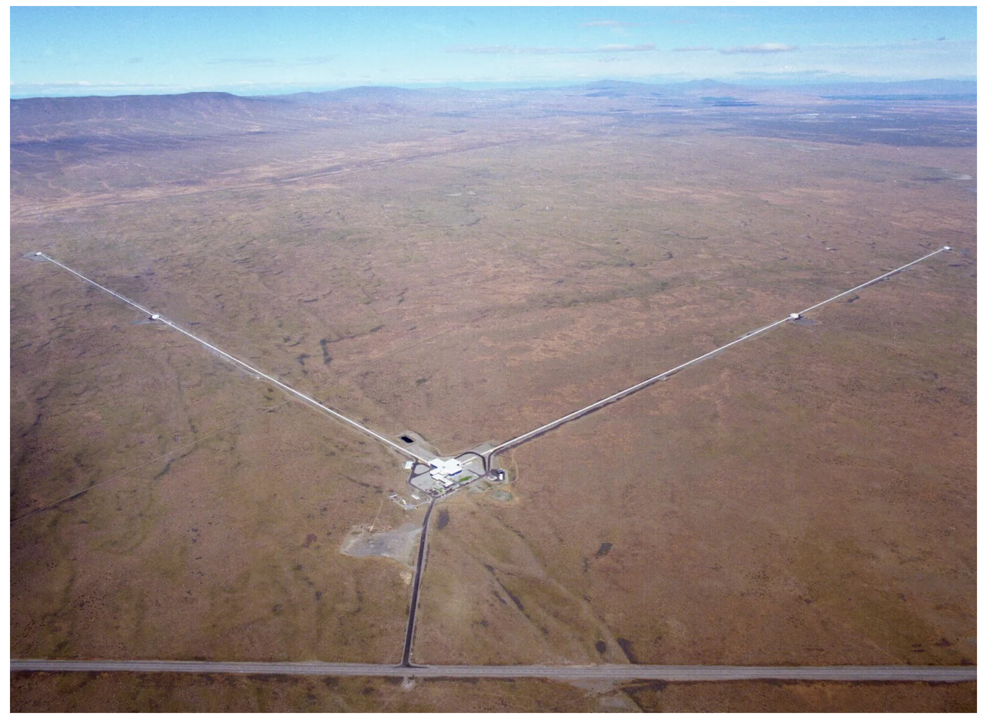


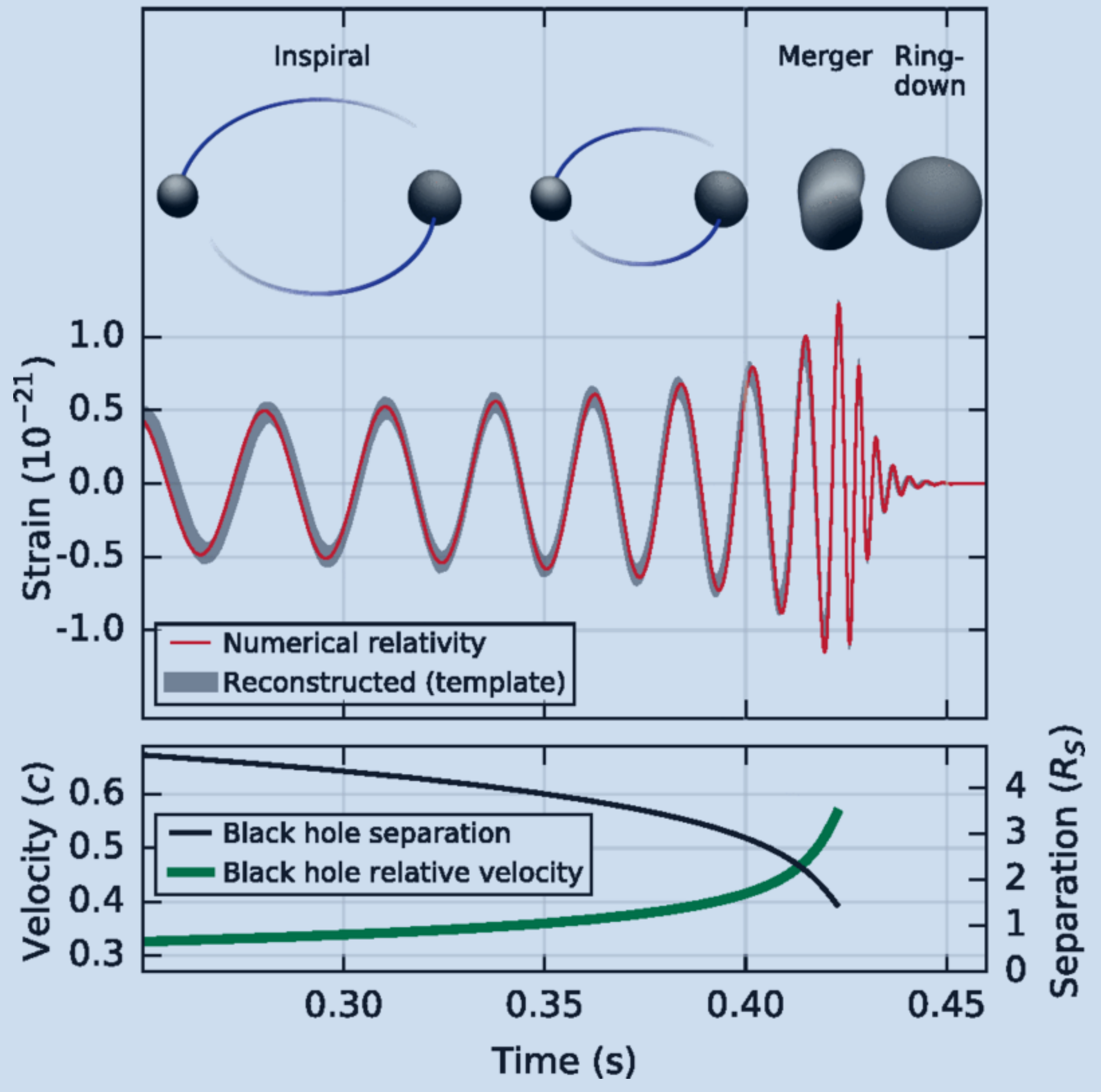


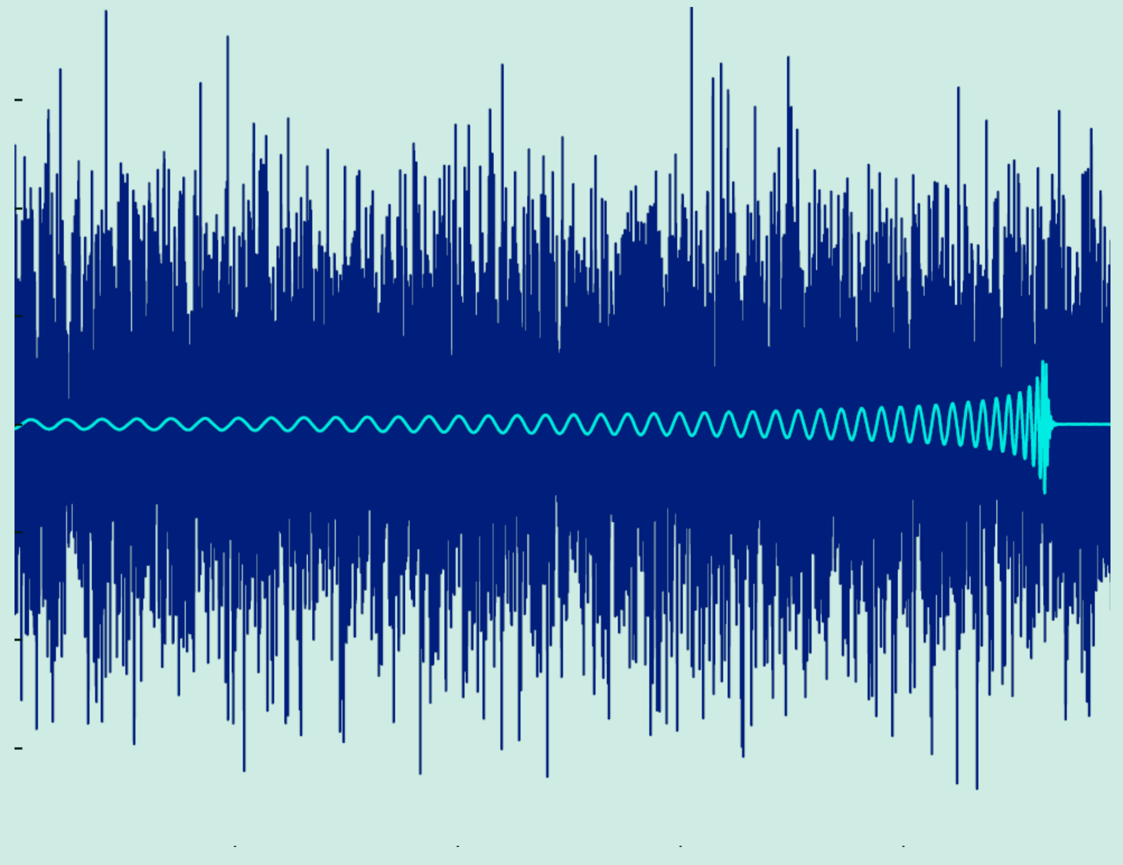


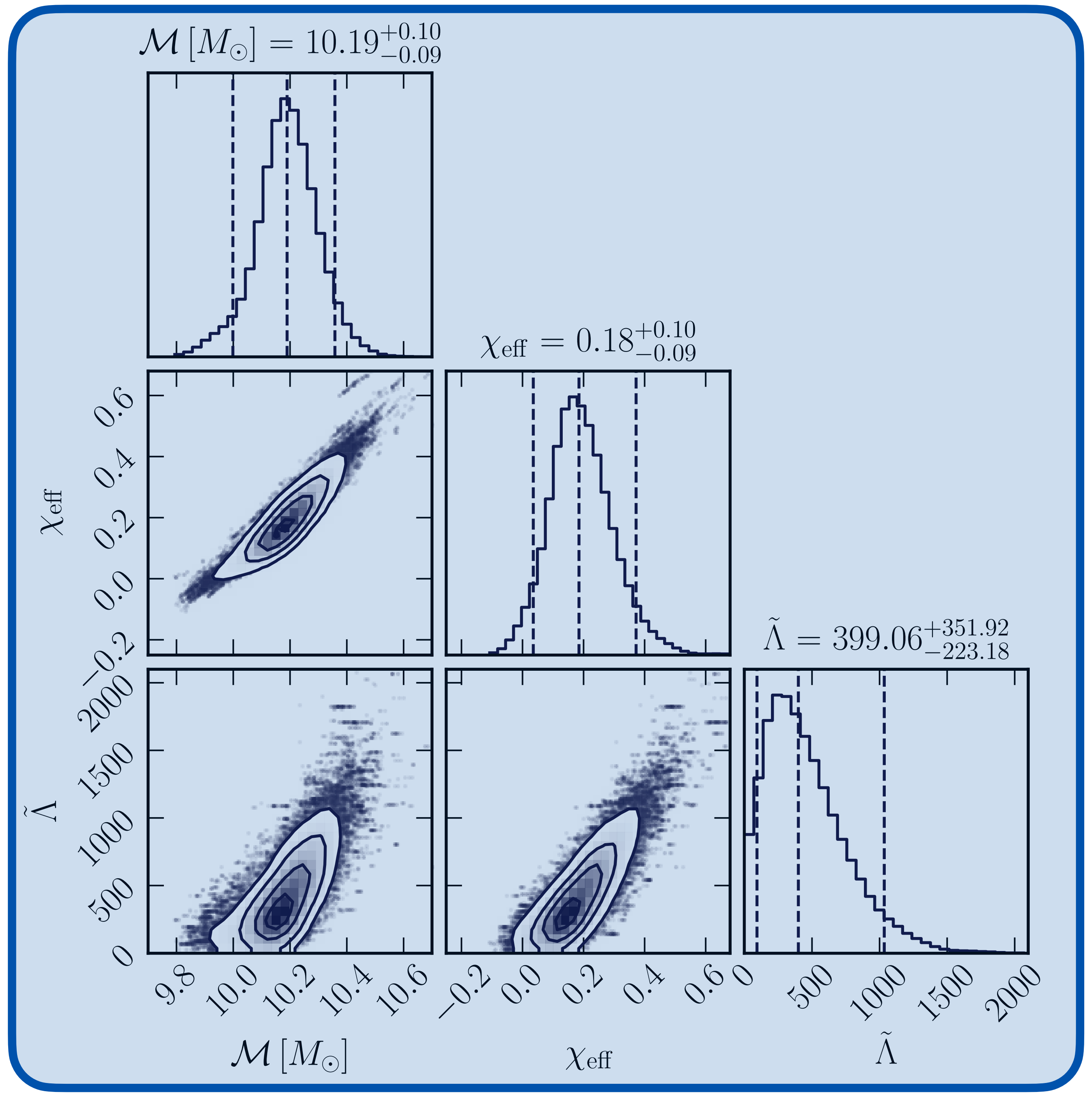


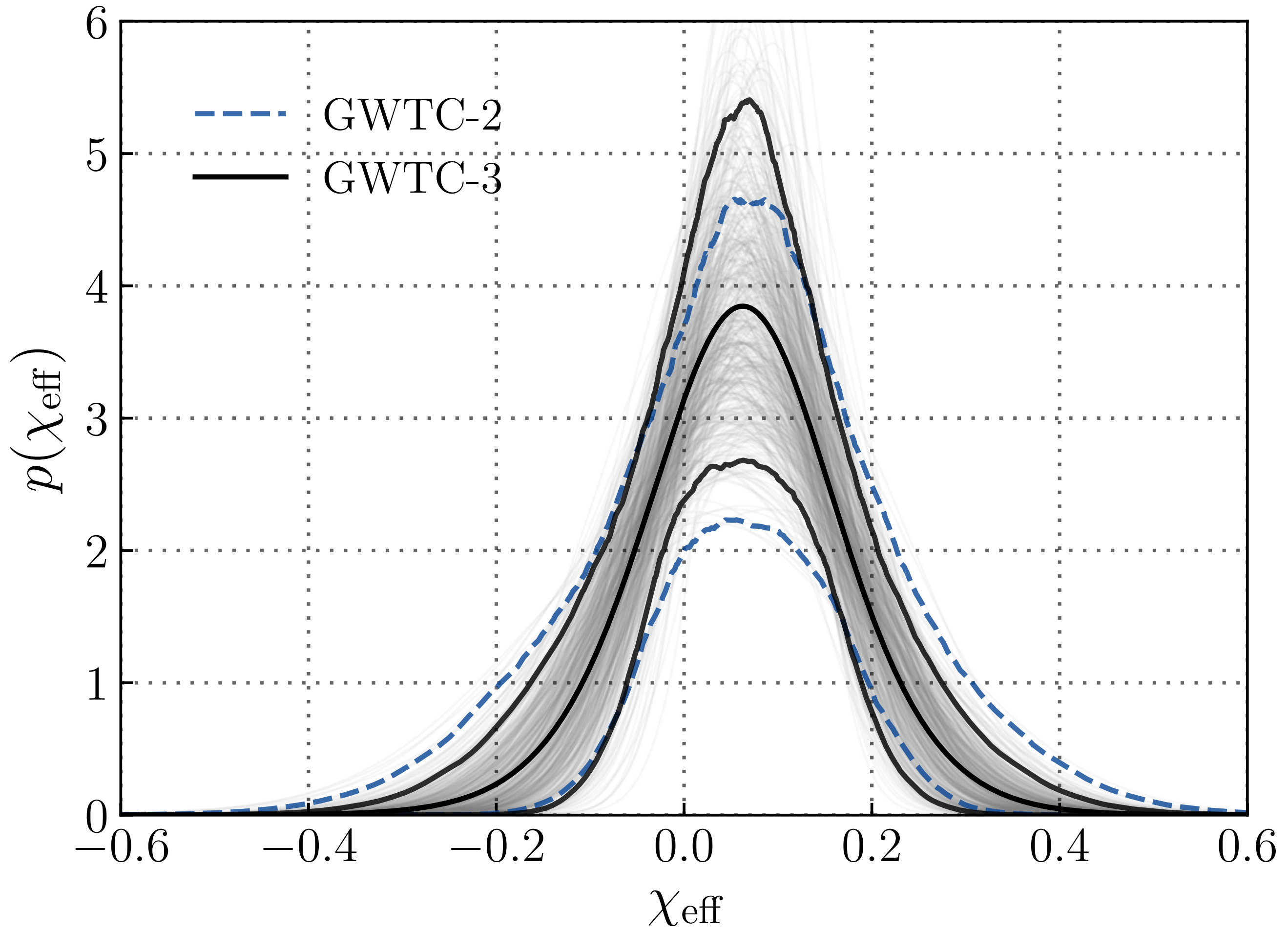
Gravitational Wave Analysis











Overview

How can automatically-differentiable models improve GW analysis tasks?


Kaze Wong


Max Isi

+ Kelvin K. H. Lam, Adam Coogan, James Alvey, and Daniel Foreman-Mackey

What could current searches be missing?


Horng Sheng Chia


Jay Wadekar


Aaron Zimmerman

Overview

How can automatically-differentiable models improve GW analysis tasks?

Kaze Wong **Max Isi**

+ Kelvin K. H. Lam, Adam Coogan, James Alvey, and Daniel Foreman-Mackey

What could current searches be missing?

Horng Sheng Chia Jay Wadekar Aaron Zimmerman

Symbolic and numerical differentiation

Symbolic and numerical differentiation

$$z = y\sin(x) + y^3$$

$$\frac{\partial z}{\partial x} = y\cos(x) \qquad \frac{\partial z}{\partial y} = \sin(x) + 3y^2$$

- Requires closed-form expressions
- Can lead to “expression swell”

Symbolic and numerical differentiation

$$z = y\sin(x) + y^3$$

$$\frac{\partial z}{\partial x} = y\cos(x) \qquad \frac{\partial z}{\partial y} = \sin(x) + 3y^2$$

- Requires closed-form expressions
- Can lead to “expression swell”

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

- Requires O(n) calls of the function where n is the number of parameters of the function
- Leads to numerical inaccuracies (rounding and truncation error)

What is Automatic Differentiation (AD)?

- Automatic differentiation is a family of methods which allows one to compute machine precision derivatives with little computational overhead for arbitrary computational programs
- Its foundation is that each mathematical step is itself differentiable and composable using the chain rule

$$\frac{\partial w}{\partial t} = \sum_{i}^{N} \frac{\partial w}{\partial u_i} \frac{\partial u_i}{\partial t}$$

Forward mode AD

- Forward mode breaks down a function into intermediate steps and simultaneously evaluates both the value of the intermediate variable and its derivative (also known as using dual numbers)
- Discussed extensively here
- Runs in ~n x O(f)

$$f(x_1, x_2) = \sin(x_1 x_2) + x_1 x_2$$

Forward mode AD

- Forward mode breaks down a function into intermediate steps and simultaneously evaluates both the value of the intermediate variable and its derivative (also known as using dual numbers)
- Discussed extensively here
- Runs in ~n x O(f)

$$f(x_1, x_2) = \sin(x_1 x_2) + x_1 x_2$$

$$\begin{aligned} x_1 &= ? \\ x_2 &= ? \\ a &= x_1 x_2 \\ b &= \sin(a) \\ c &= a + b \end{aligned}$$

Forward mode AD

- Forward mode breaks down a function into intermediate steps and simultaneously evaluates both the value of the intermediate variable and its derivative (also known as using dual numbers)
- Discussed extensively here
- Runs in ~n x O(f)

$$f(x_1, x_2) = \sin(x_1 x_2) + x_1 x_2$$

$$\begin{aligned} x_1 &= ? \\ x_2 &= ? \\ a &= x_1 x_2 \\ b &= \sin(a) \\ c &= a + b \end{aligned} \qquad \begin{aligned} dx_1 &= 1 \\ dx_2 &= 0 \\ da &= x_1 dx_2 + x_2 dx_1 \\ db &= \cos(a) da \\ dc &= da + db \end{aligned}$$

Forward mode AD

- Forward mode breaks down a function into intermediate steps and simultaneously evaluates both the value of the intermediate variable and its derivative (also known as using dual numbers)
- Discussed extensively here
- Runs in ~n x O(f)

$$f(x_1, x_2) = \sin(x_1 x_2) + x_1 x_2$$

$$\begin{aligned} x_1 &= ? & dx_1 &= 1 \\ x_2 &= ? & dx_2 &= 0 \\ a &= x_1 x_2 & da &= x_1 dx_2 + x_2 dx_1 \\ b &= \sin(a) & db &= \cos(a) da \\ c &= a + b & dc &= da + db \end{aligned}$$

$$f : \mathbb{R}^n \to \mathbb{R}^m, \quad \text{where} \quad n \ll m$$

Why JAX?

[https://github.com/google/jax]

Why JAX?

Runs on pure Python and numpy(ish) code

[https://github.com/google/jax]

Why JAX?

Runs on pure Python and numpy(ish) code

Use XLA compilation to speed up code substantially

[https://github.com/google/jax]

Why JAX?

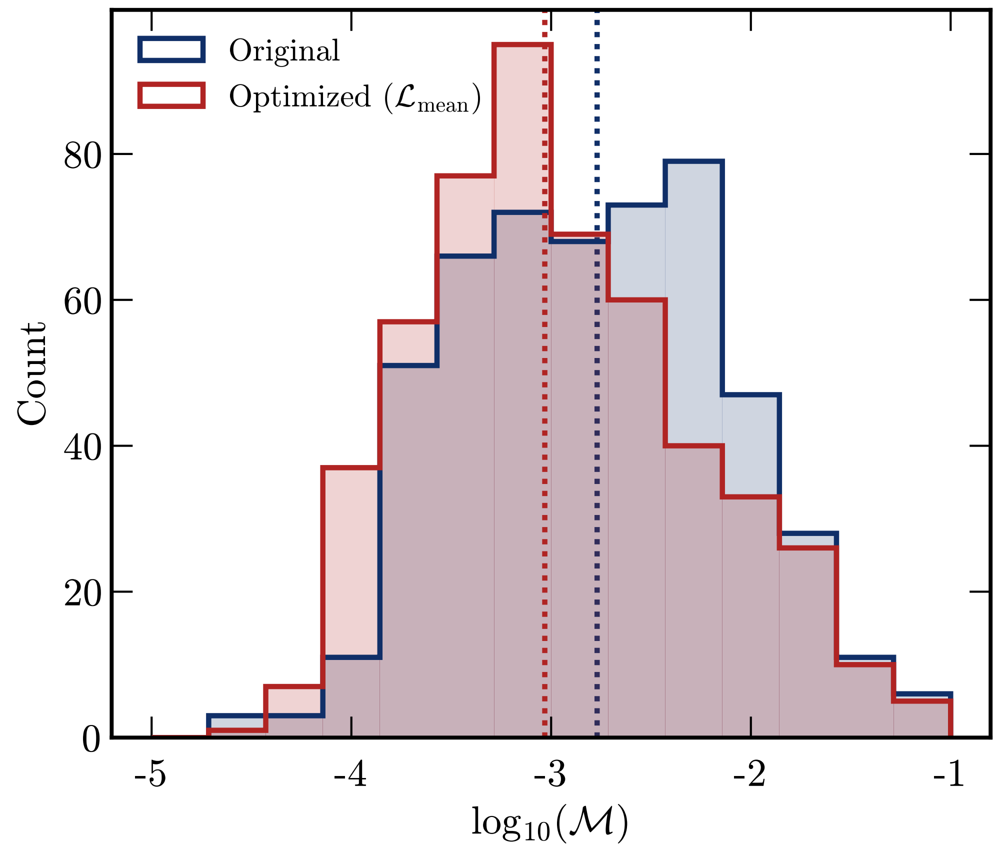
Runs on pure Python and numpy(ish) code

Use XLA compilation to speed up code substantially

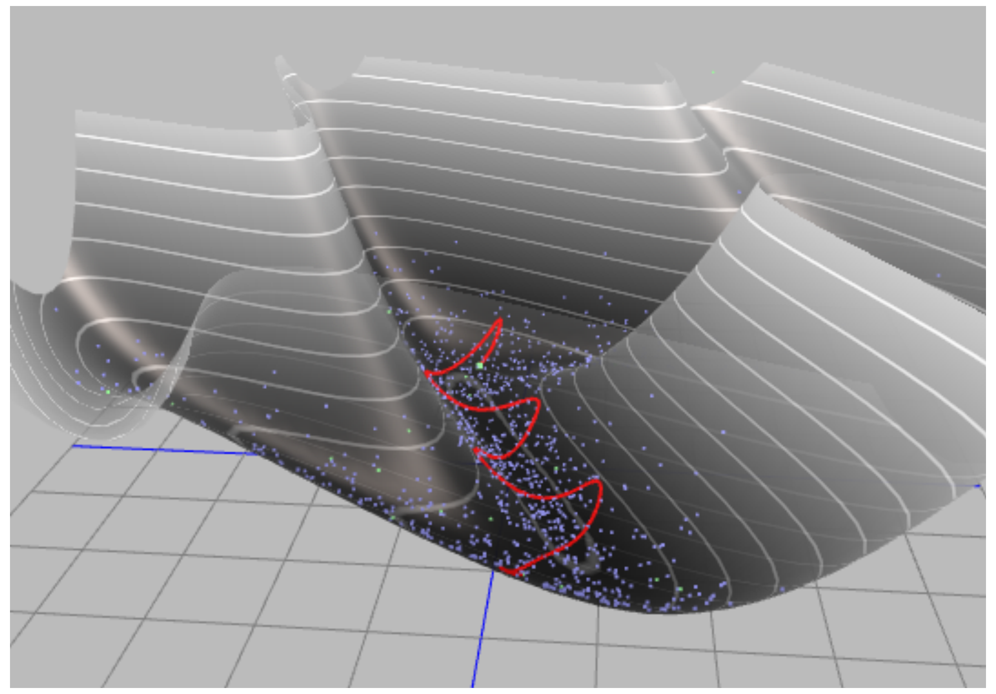
Scales to different computing architectures and multiple cores or a cluster

[https://github.com/google/jax]

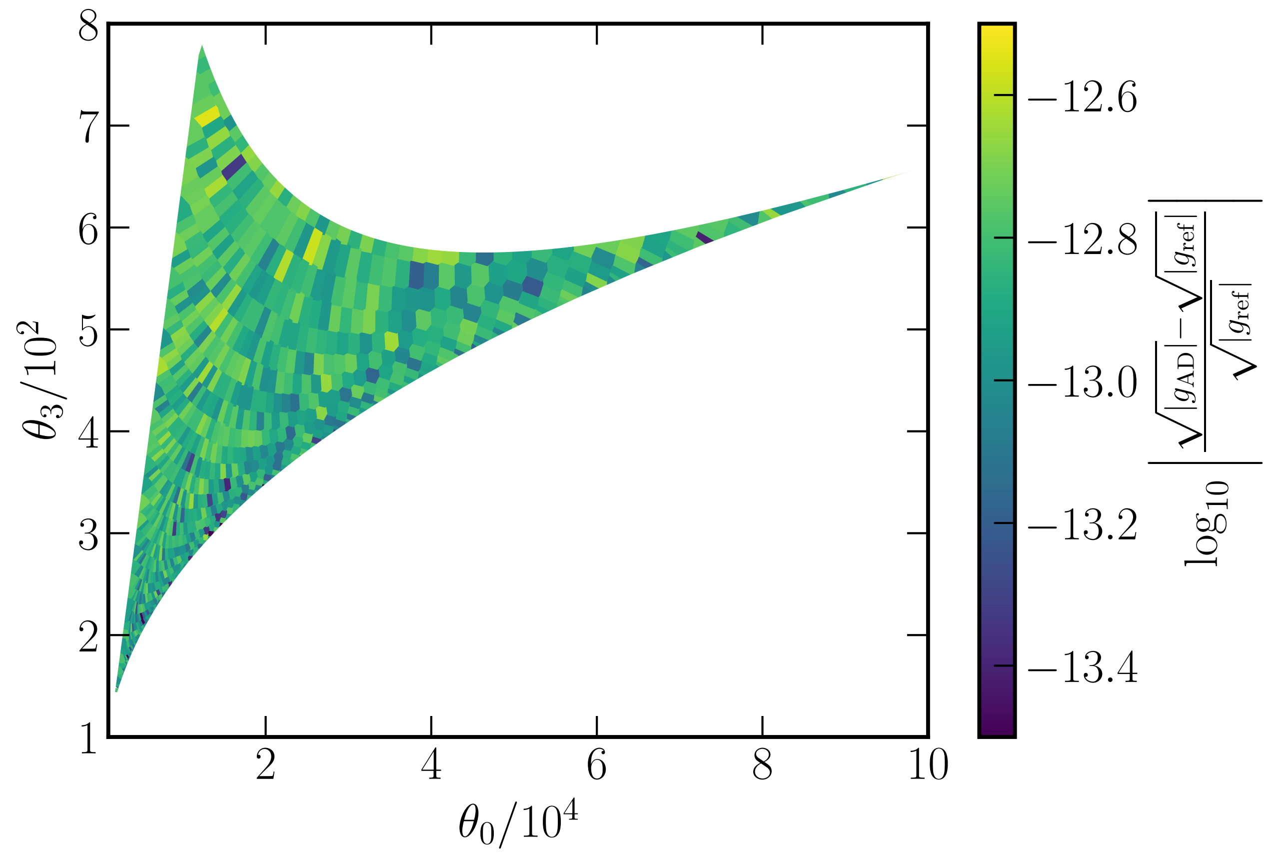
Ripple



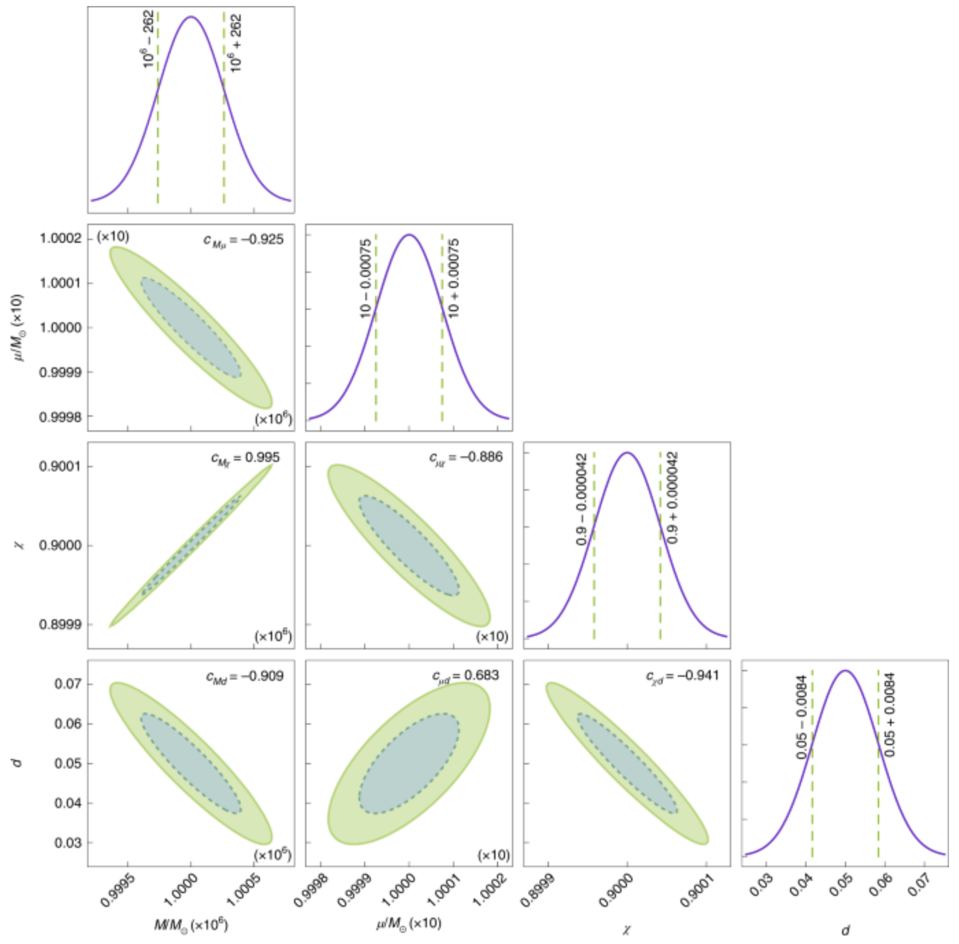
Waveform Optimization



Gradient-based Parameter Estimation



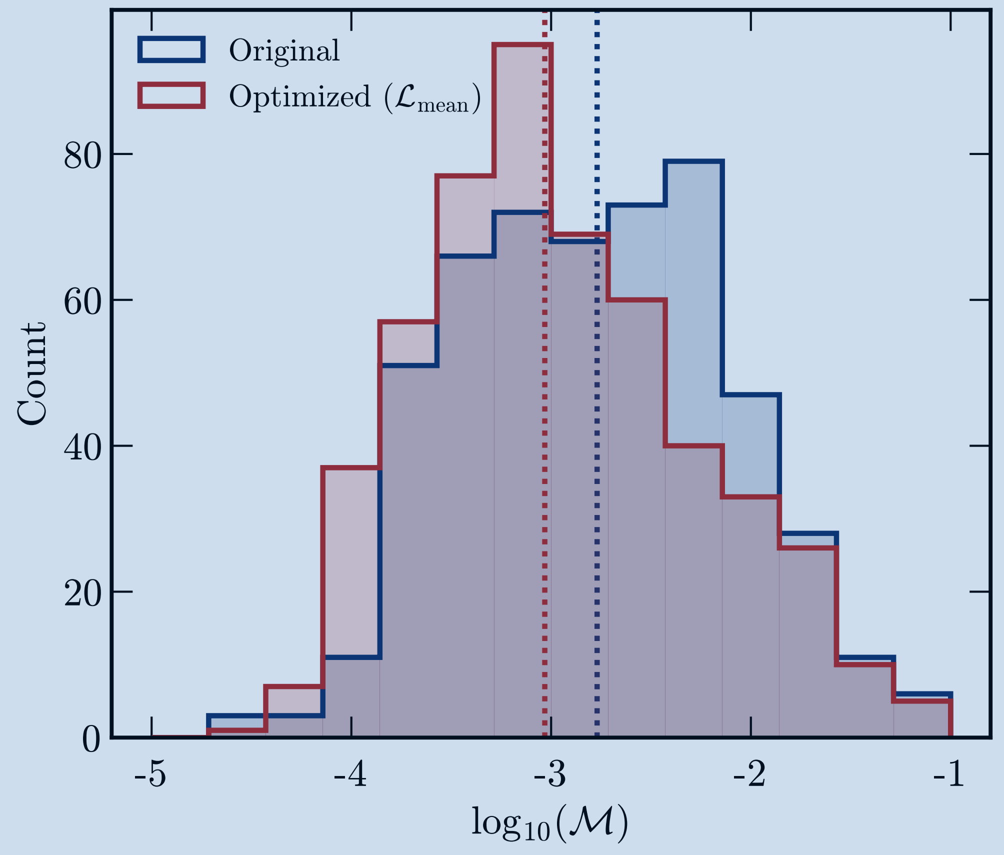
Template Bank Generation



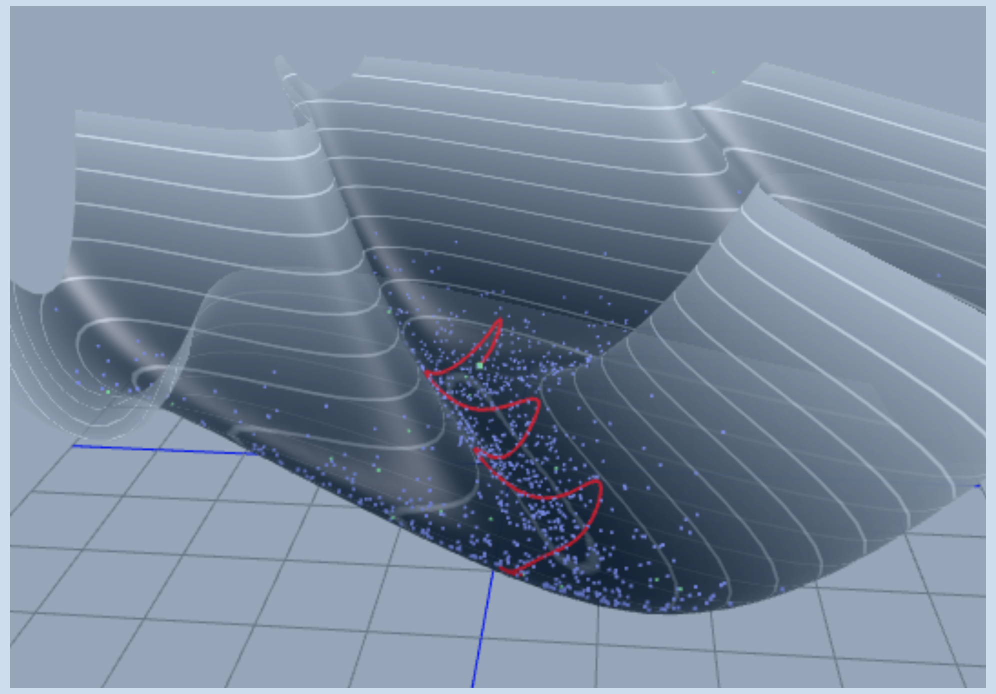
Fisher Analyses

[github.com/tedwards2412/ripple]

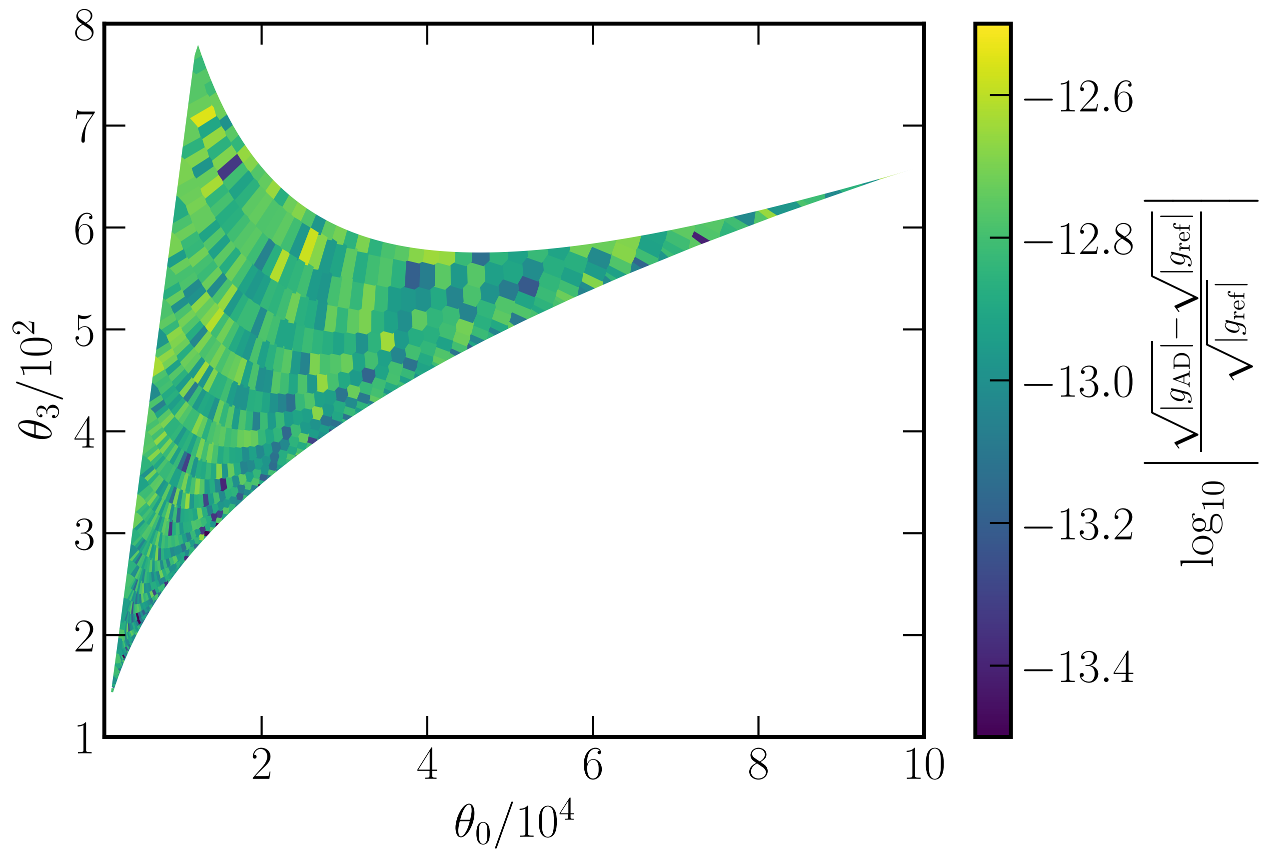
Ripple



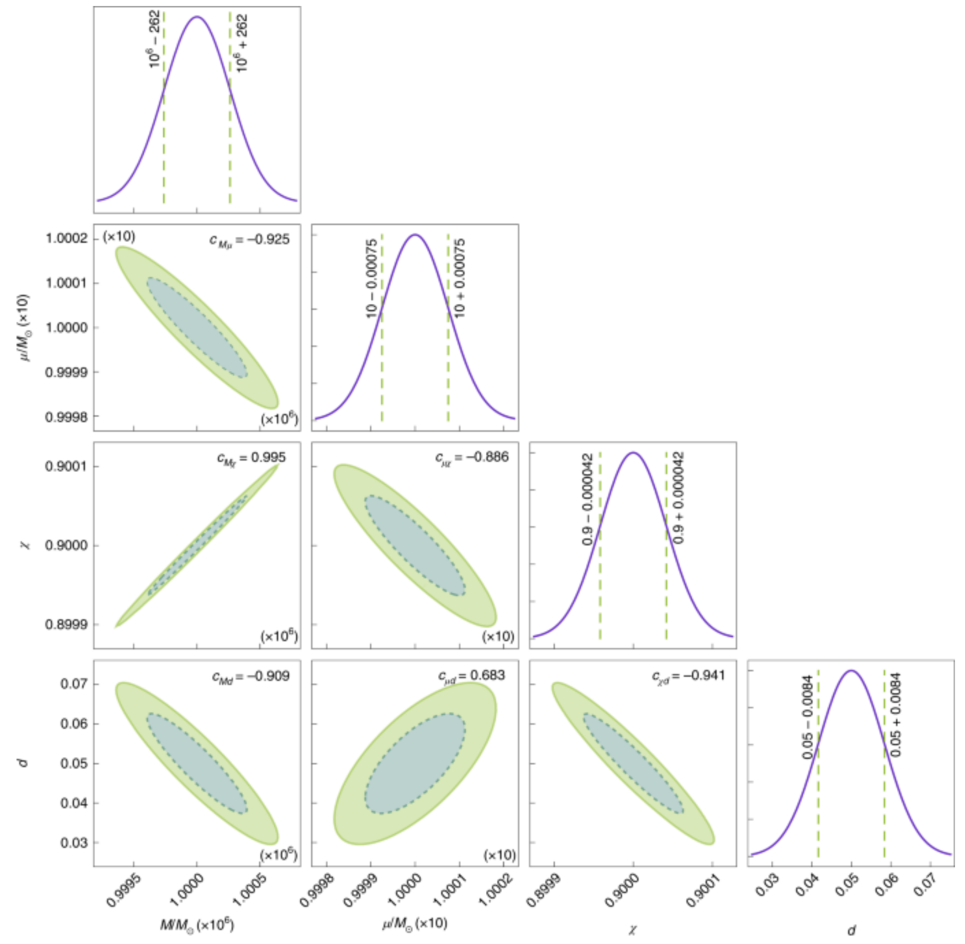
Waveform Optimization



Gradient-based Parameter Estimation



Template Bank Generation



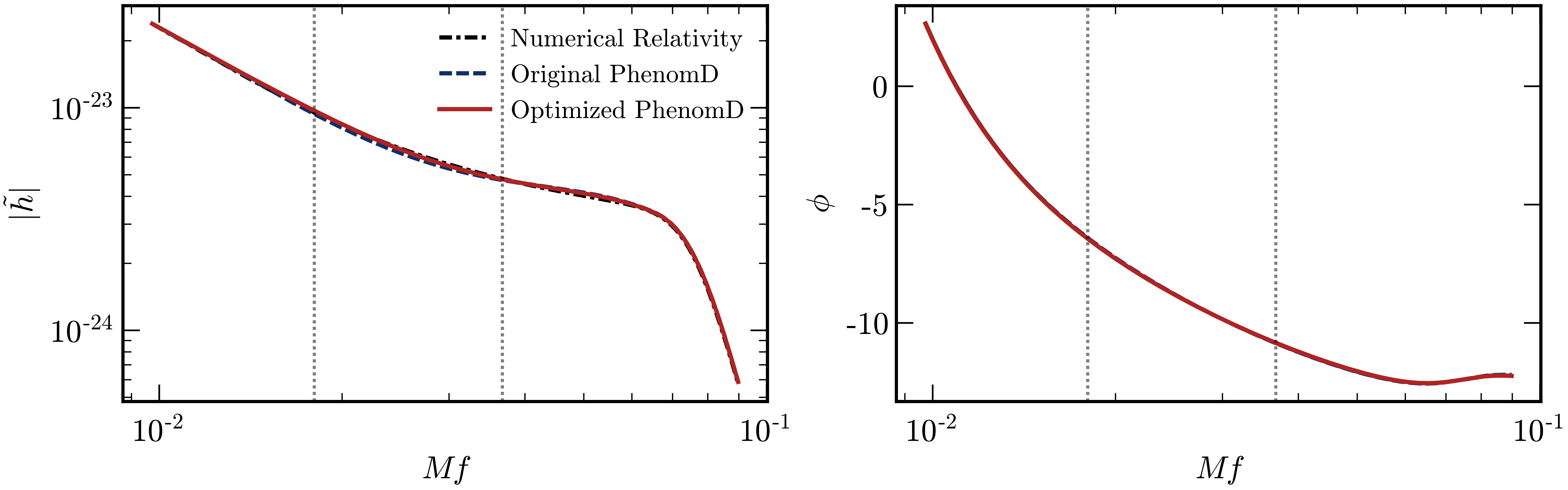
Fisher Analyses

[github.com/tedwards2412/ripple]

Optimizing Waveforms

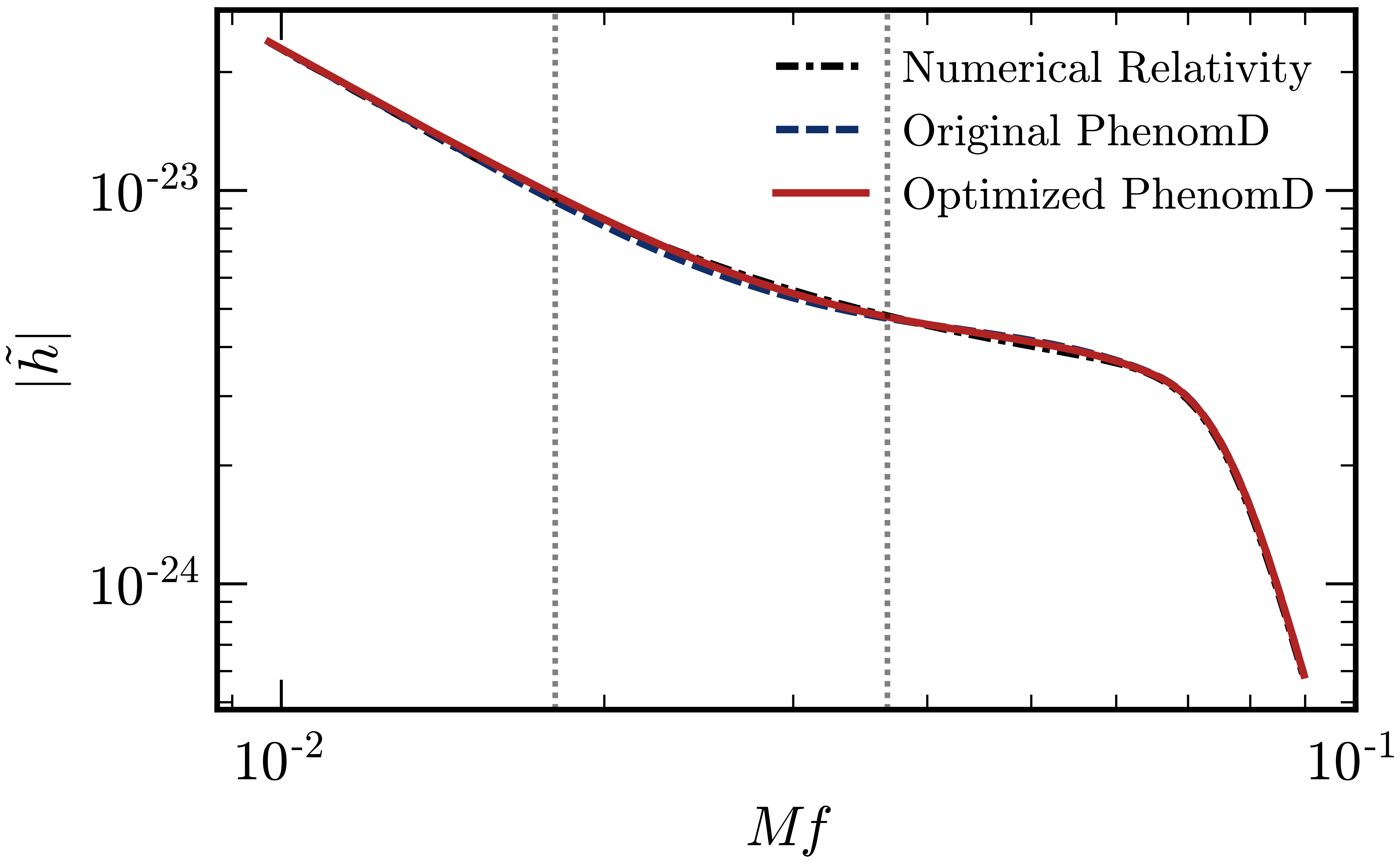
$$\tilde{h}(f;\boldsymbol{p}) = \mathcal{A}(\boldsymbol{p})e^{i\phi(f;\boldsymbol{p})}$$

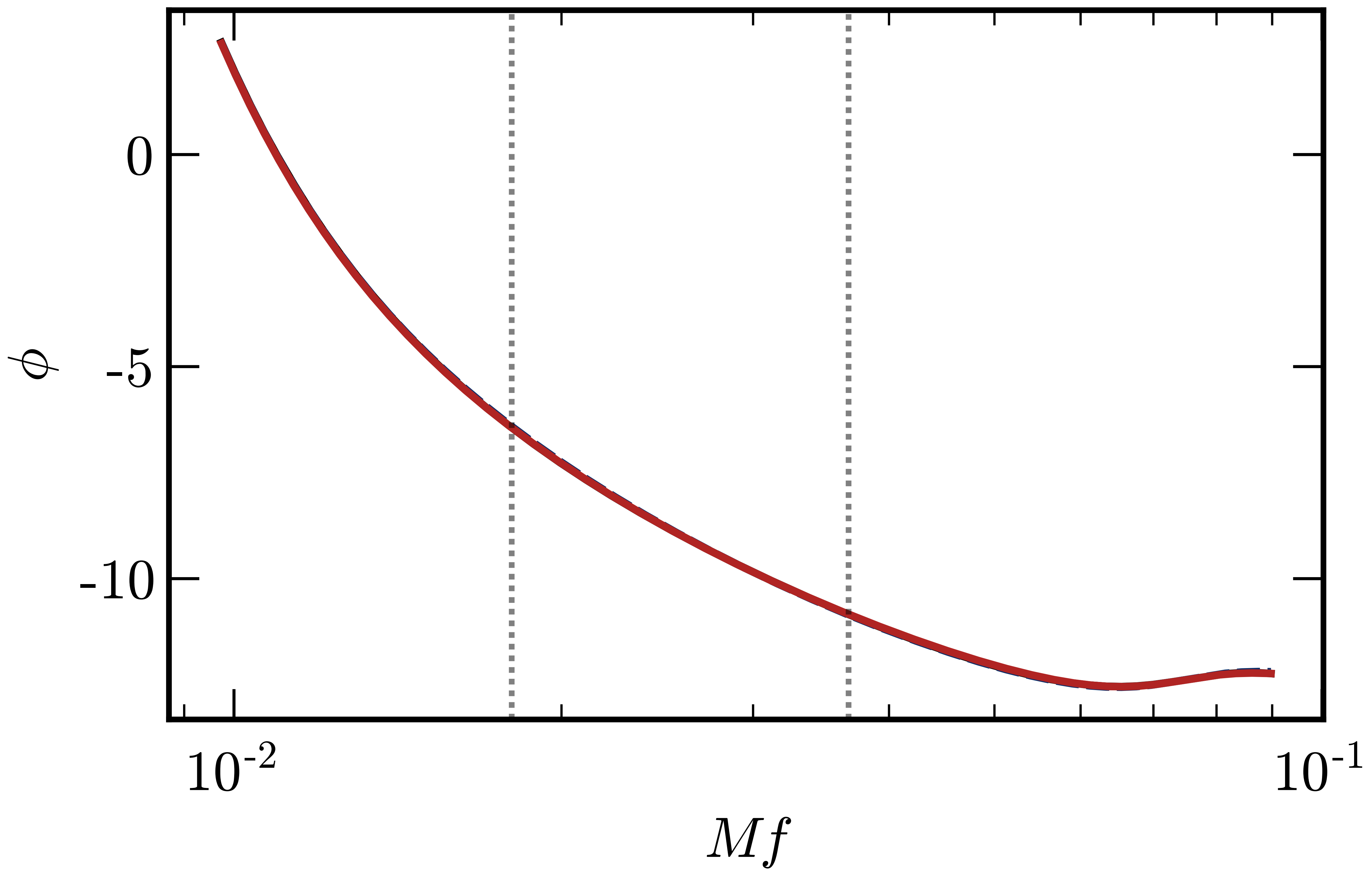
Optimizing Waveforms



$$\tilde{h}(f;\boldsymbol{p}) = \mathcal{A}(\boldsymbol{p})e^{i\phi(f;\boldsymbol{p})}$$

Optimizing Waveforms





$$\mathcal{A}(f;\boldsymbol{p},\boldsymbol{\lambda}) \propto \mathcal{A}_{\rm PN}(\boldsymbol{p},\boldsymbol{\lambda}) + \mathcal{A}_{\rm Int}(\boldsymbol{\lambda}) + \mathcal{A}_{\rm MR}(\boldsymbol{\lambda})$$

$$\phi(f;\boldsymbol{p},\boldsymbol{\lambda}) \propto \phi_{\rm PN}(\boldsymbol{p},\boldsymbol{\lambda}) + \phi_{\rm Int}(\boldsymbol{\lambda}) + \phi_{\rm MR}(\boldsymbol{\lambda})$$

Gradient Descent?

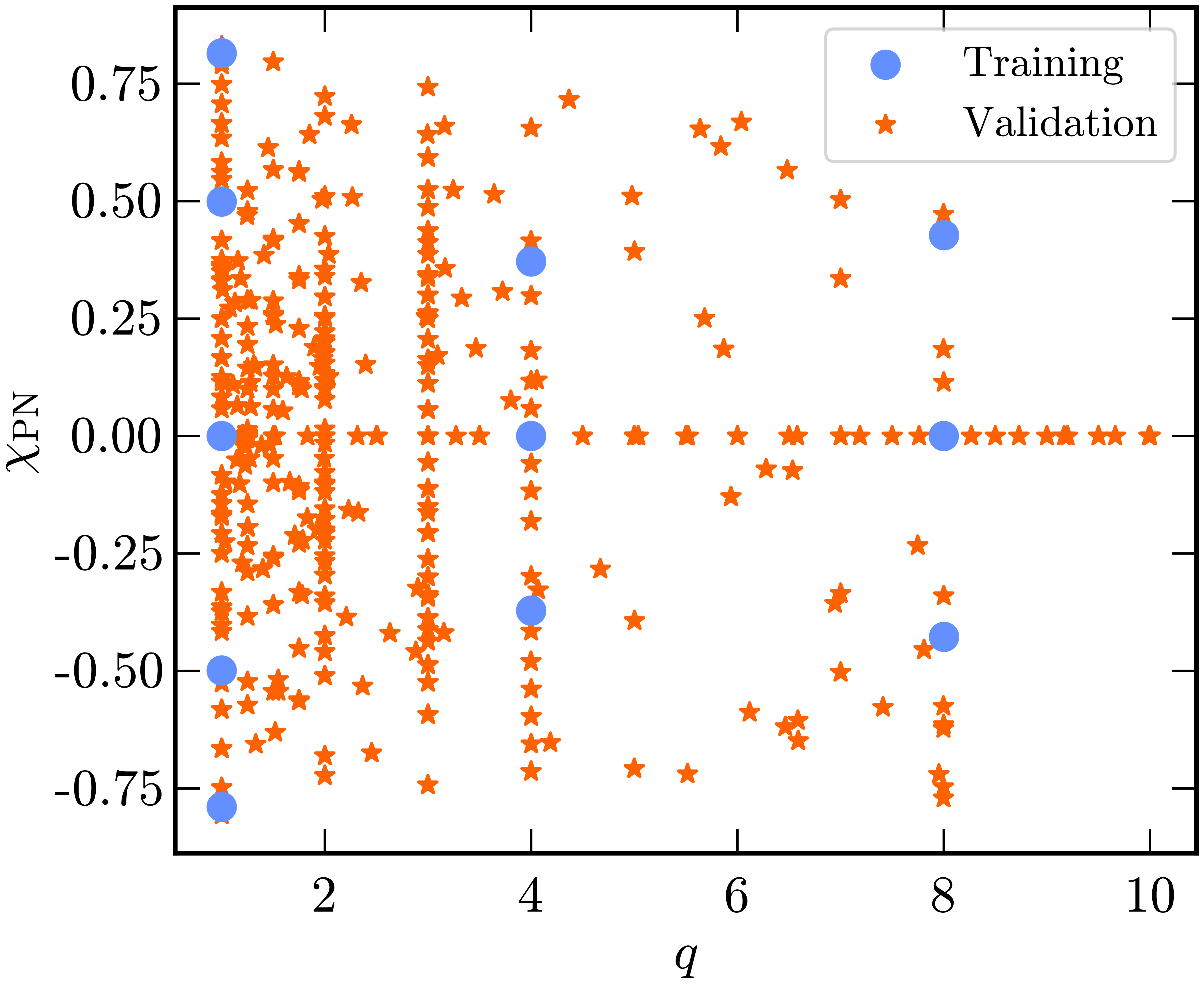
To use gradient descent we need to define a loss function, update rule, and stopping criteria

Gradient Descent?

To use gradient descent we need to define a loss function, update rule, and stopping criteria

Gradient Descent?

$$\mathcal{L}_{\text{mean}} = \frac{1}{N}\sum_{i=1}^{N}\mathcal{M}_i$$



To use gradient descent we need to define a loss function, update rule, and stopping criteria

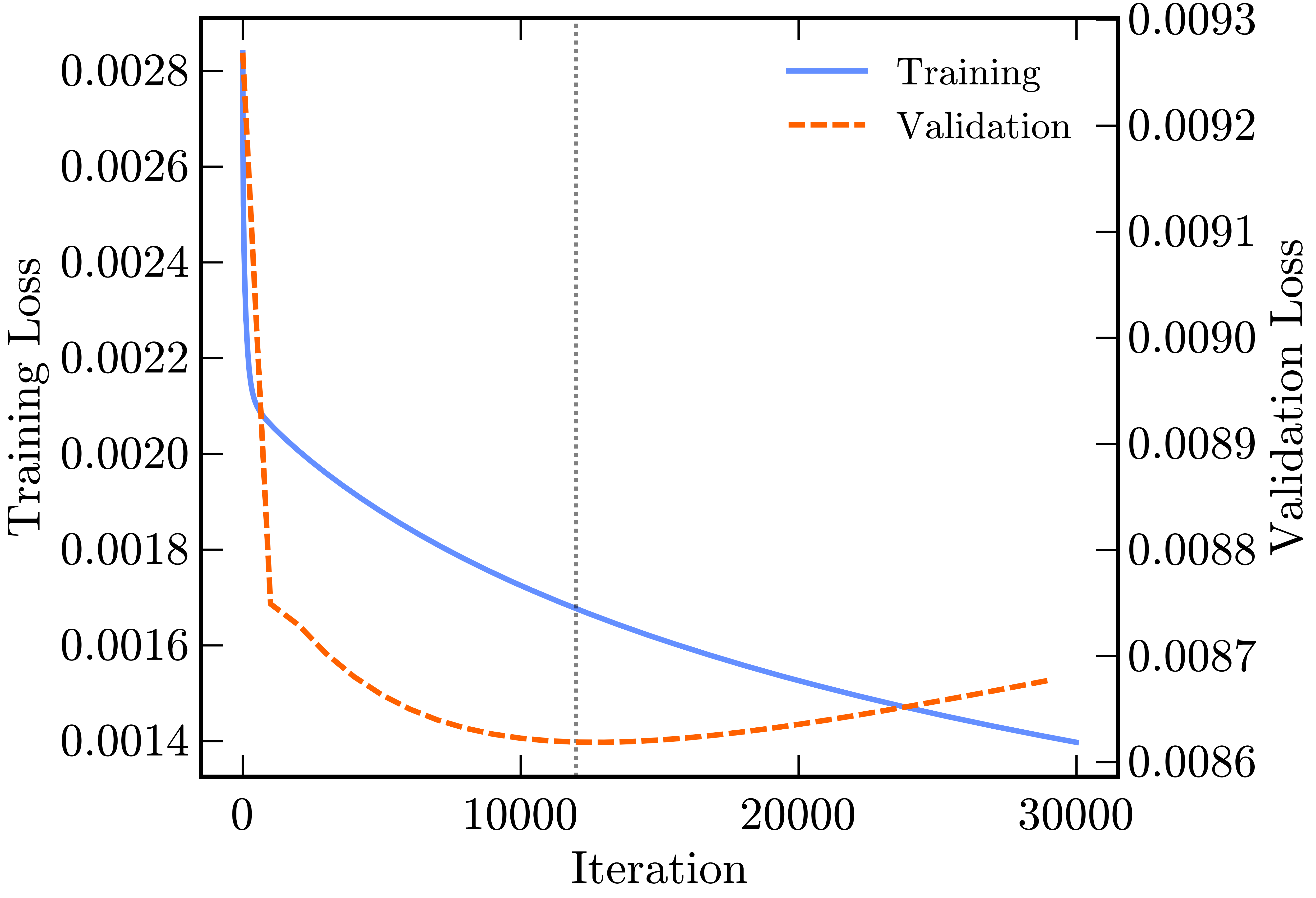
Gradient Descent?

To use gradient descent we need to define a loss function, update rule, and stopping criteria

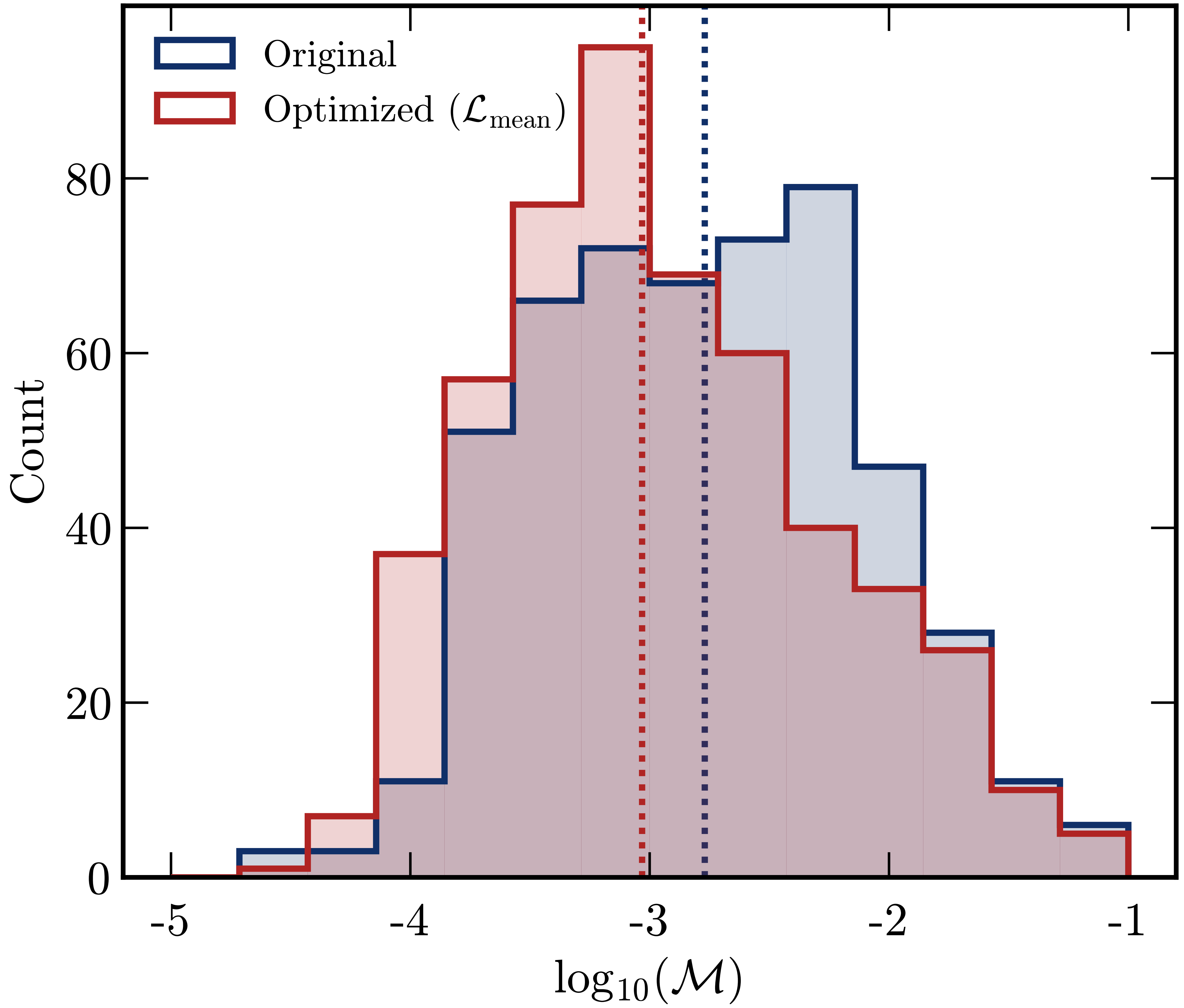
$$\lambda \to \lambda - \alpha \nabla \mathcal{L}$$

Gradient Descent?

To use gradient descent we need to define a loss function, update rule, and stopping criteria



Up to 50% Better Waveforms For Free



Parameter Estimation

$$p(\boldsymbol{\theta} \mid d) \propto p(d \mid \boldsymbol{\theta})\pi(\boldsymbol{\theta})$$

Parameter Estimation

$$p(\boldsymbol{\theta} \mid d) \propto \boxed{p(d \mid \boldsymbol{\theta})}\,\pi(\boldsymbol{\theta})$$

▢ = Likelihood

Parameter Estimation

$$p(\boldsymbol{\theta} \mid d) \propto p(d \mid \boldsymbol{\theta})\pi(\boldsymbol{\theta})$$

▢ = Likelihood

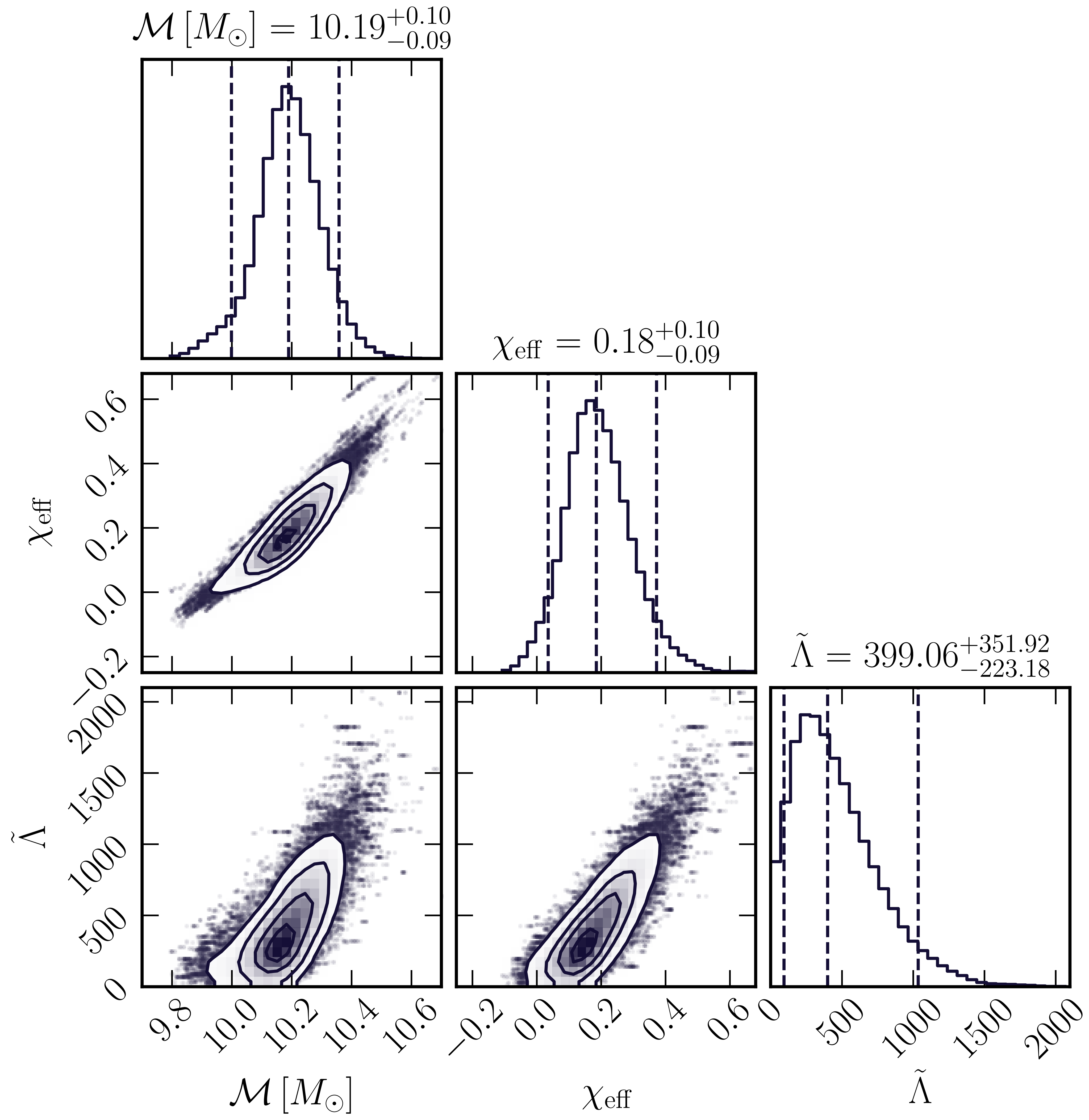
▢ = Prior

Parameter Estimation

$$p(\boldsymbol{\theta} \mid d) \propto p(d \mid \boldsymbol{\theta})\pi(\boldsymbol{\theta})$$

▢ = Likelihood

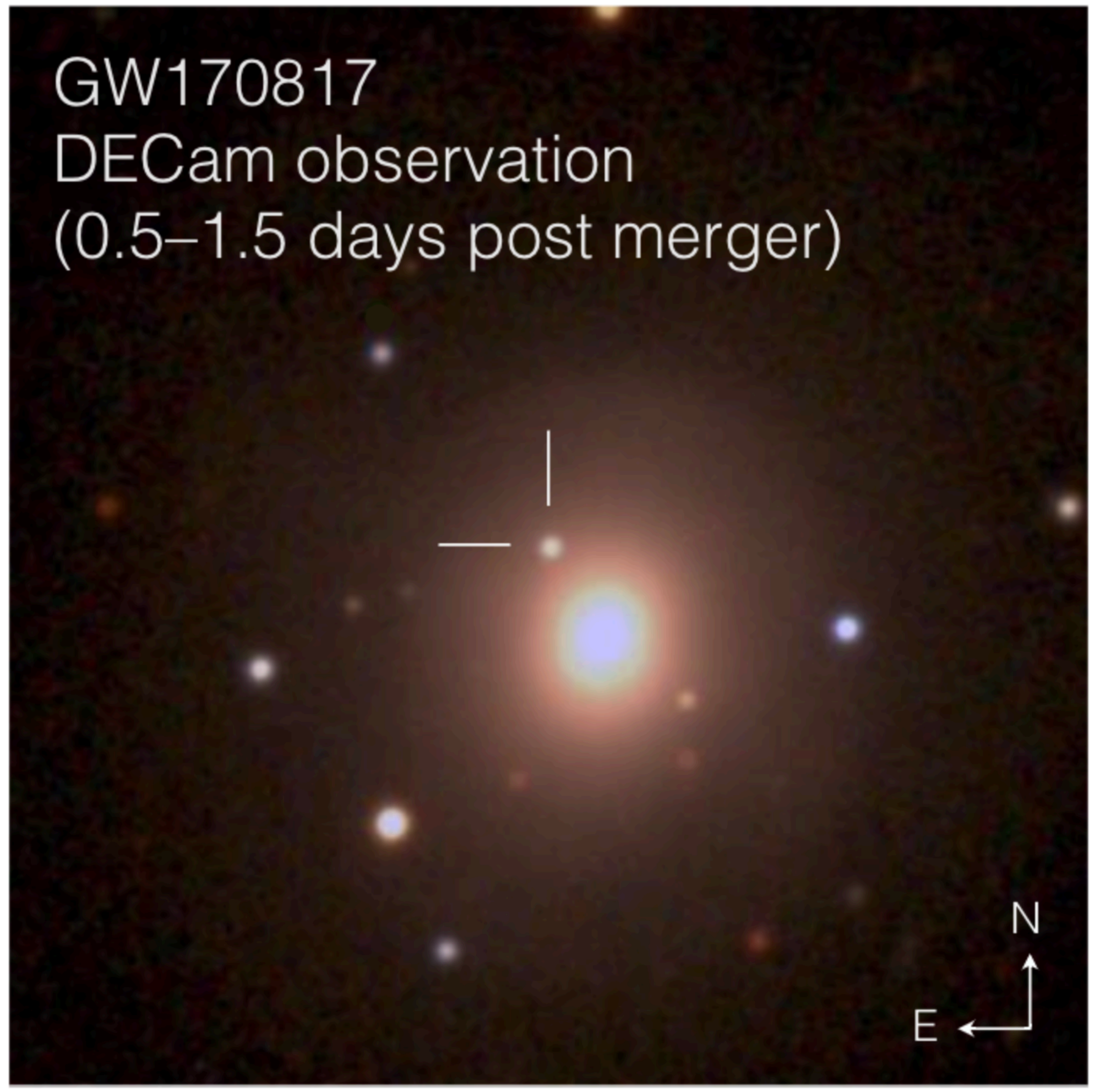
▢ = Prior



Why Accelerate PE?

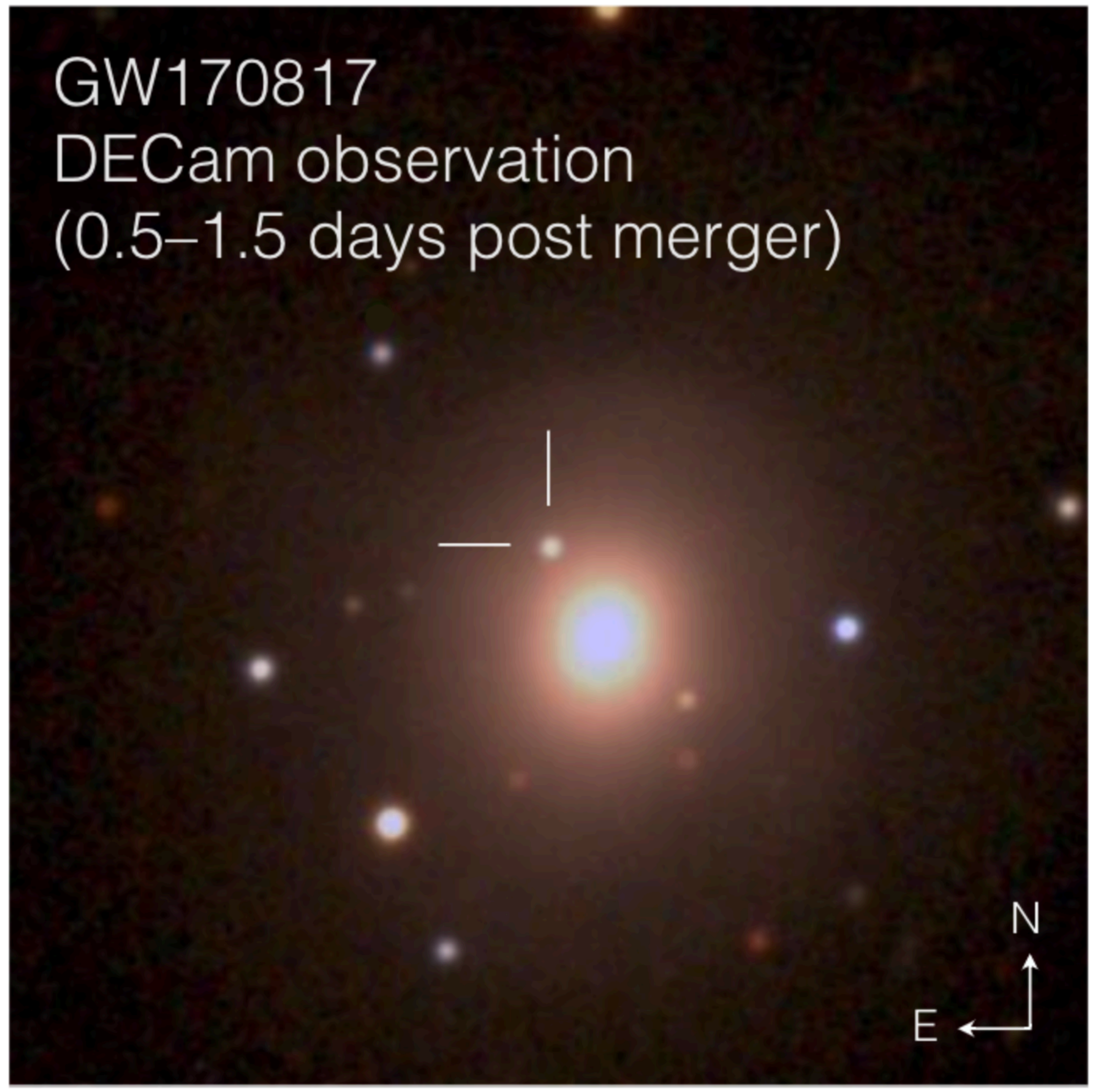
Why Accelerate PE?

Low-latency follow up

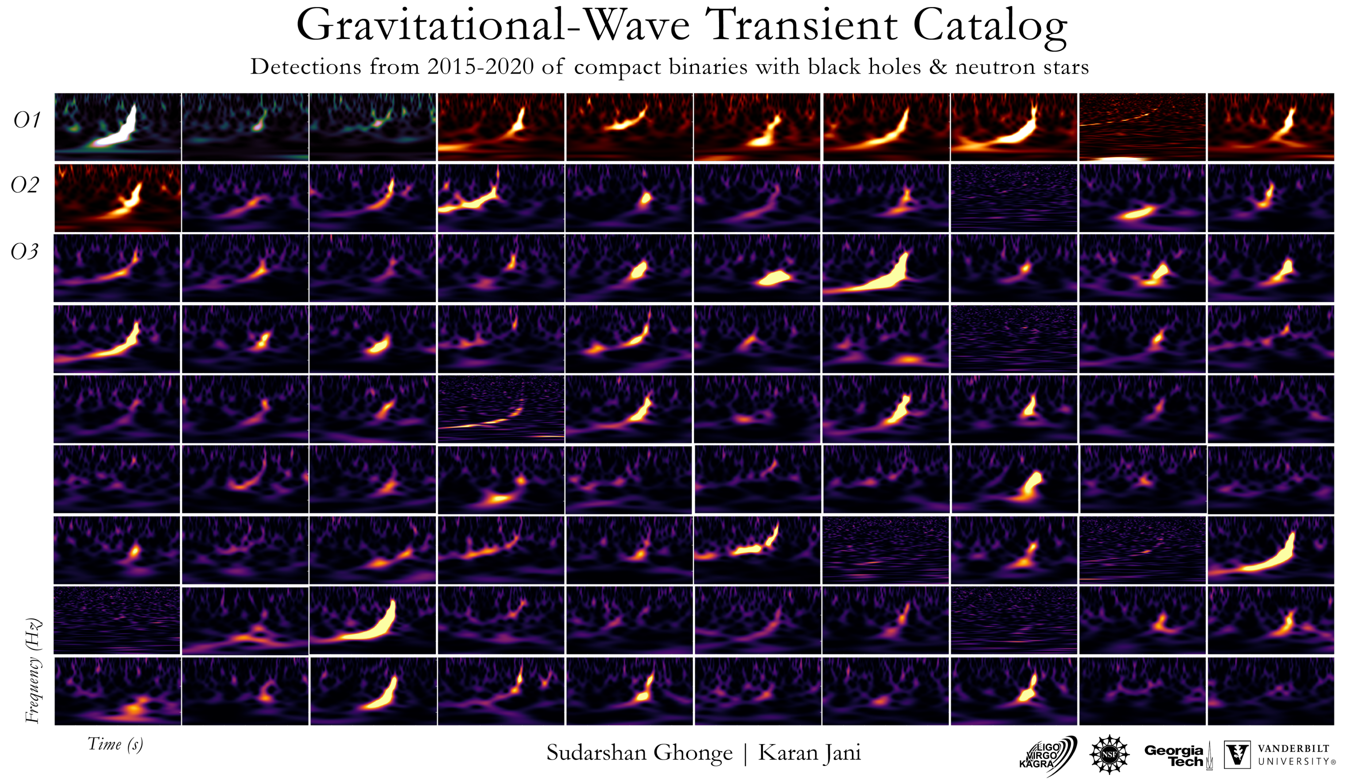


Why Accelerate PE?

Low-latency follow up

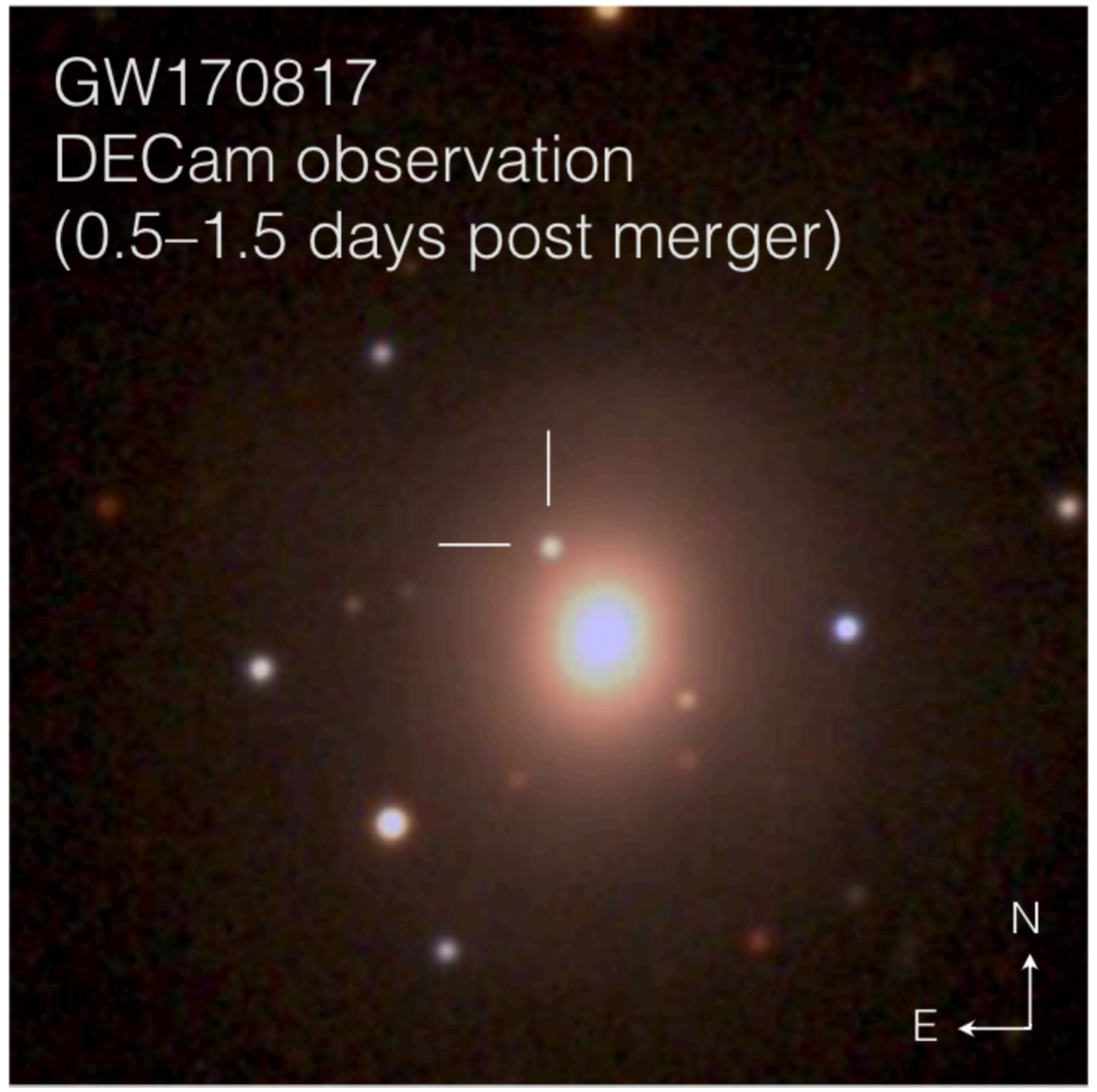


Overall compute

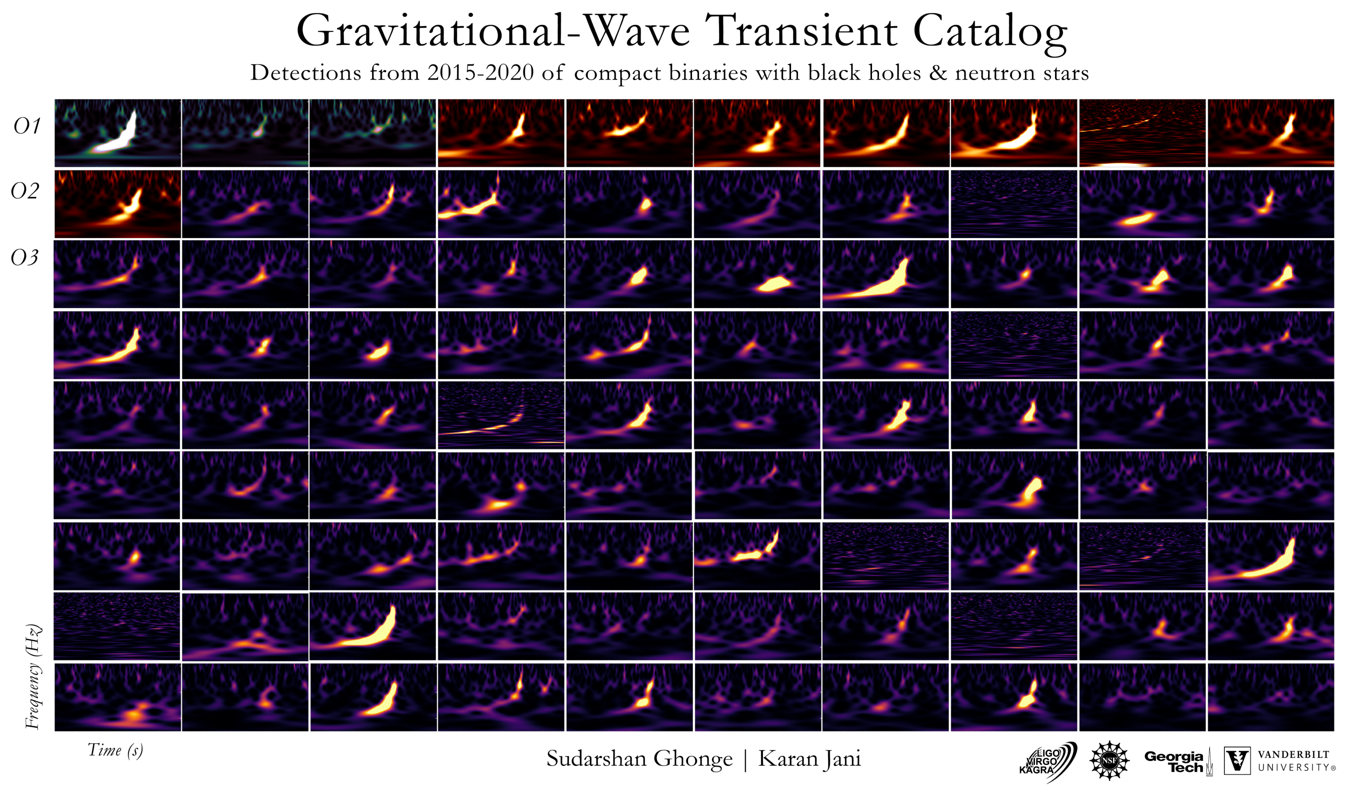


Why Accelerate PE?

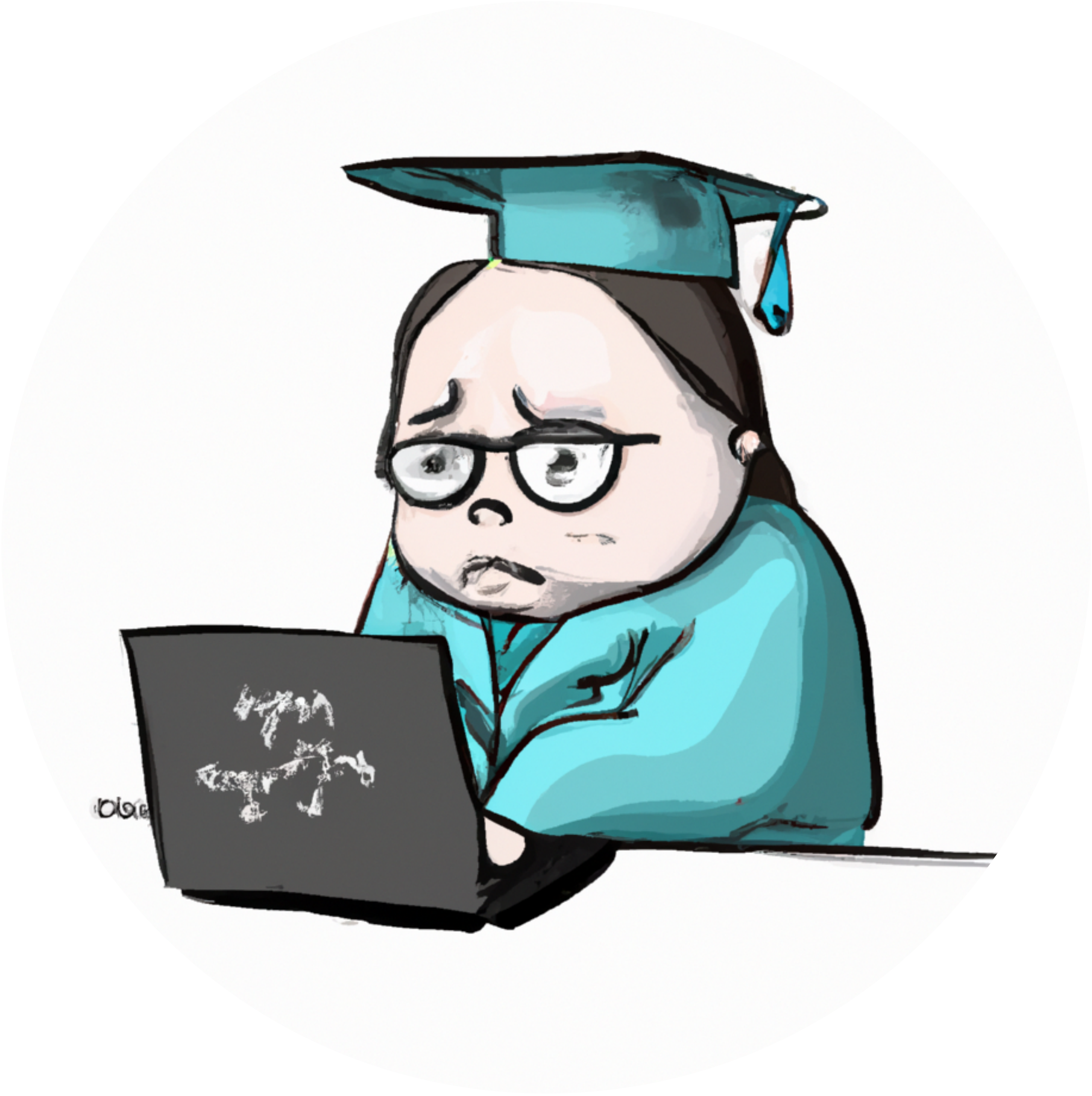
Low-latency follow up



Overall compute



Development cycle



flowMC

$$A\left(x^{\prime}, x\right)=\min \left(1, \frac{P\left(x^{\prime}\right)}{P(x)} \frac{g\left(x \mid x^{\prime}\right)}{g\left(x^{\prime} \mid x\right)}\right)$$

[github.com/kazewong/flowMC]

flowMC

$$A\left(x^{\prime}, x\right)=\min \left(1, \frac{P\left(x^{\prime}\right)}{P(x)} \frac{g\left(x \mid x^{\prime}\right)}{g\left(x^{\prime} \mid x\right)}\right)$$

[github.com/kazewong/flowMC]

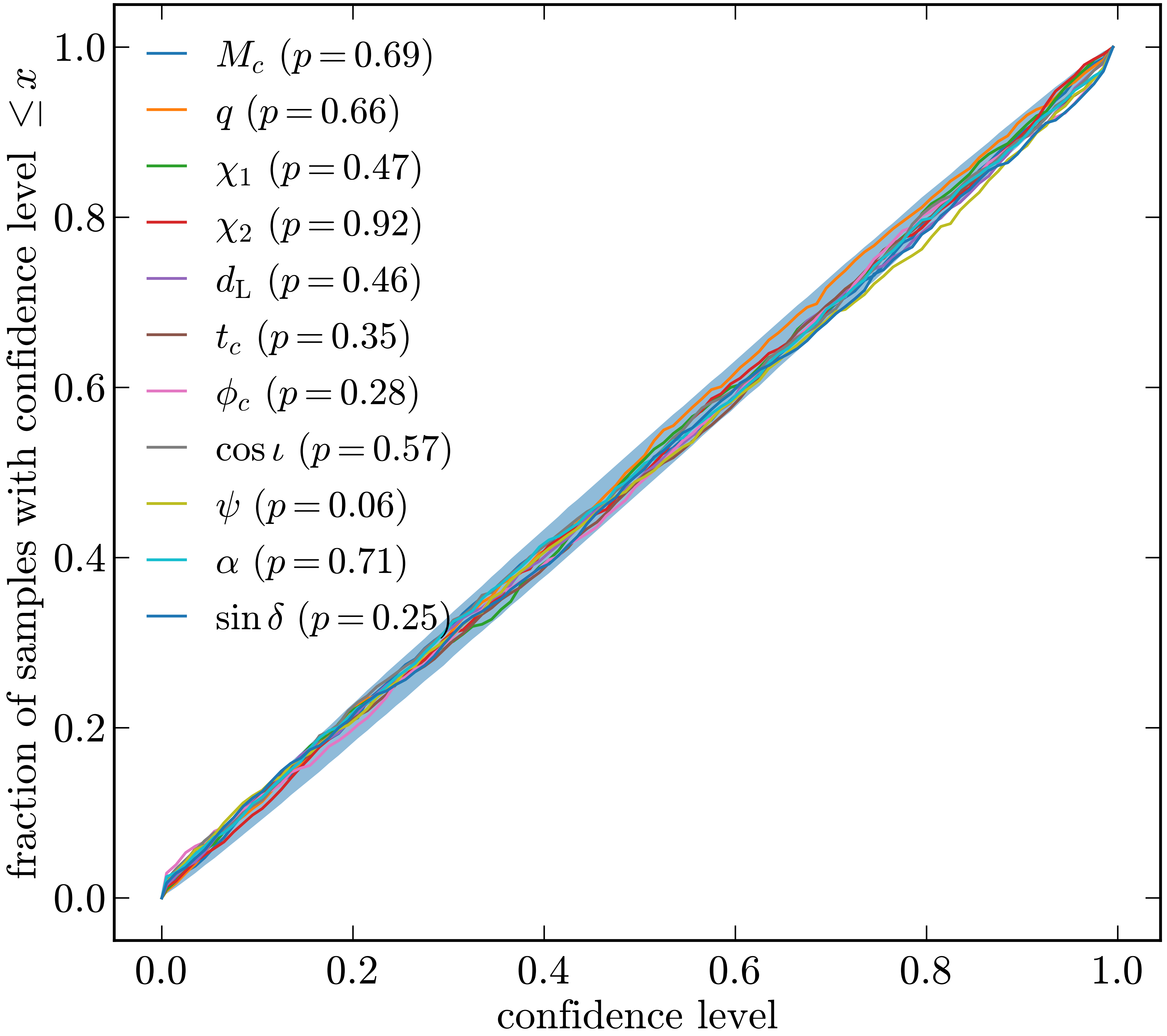
flowMC

$$A\left(x', x\right) = \min\left(1, \frac{P\left(x'\right)}{P(x)} \frac{g\left(x \mid x'\right)}{g\left(x' \mid x\right)}\right)$$

[github.com/kazewong/flowMC]

Accuracy and Speed

For both BNS and BBH, we achieve converged results in **~1 min** on an A100 GPU



[Kaze Wong, Max Isi, **TE**: 2302.05333]

Overview

How can automatically-differentiable models improve GW analysis tasks?

Kaze Wong

Max Isi

+ Kelvin K. H. Lam, Adam Coogan, James Alvey, and Daniel Foreman-Mackey

What could current searches be missing?

Horng Sheng Chia

Jay Wadekar

Aaron Zimmerman

Overview

How can automatically-differentiable models improve GW analysis tasks?

Kaze Wong

Max Isi

+ Kelvin K. H. Lam, Adam Coogan, James Alvey, and Daniel Foreman-Mackey

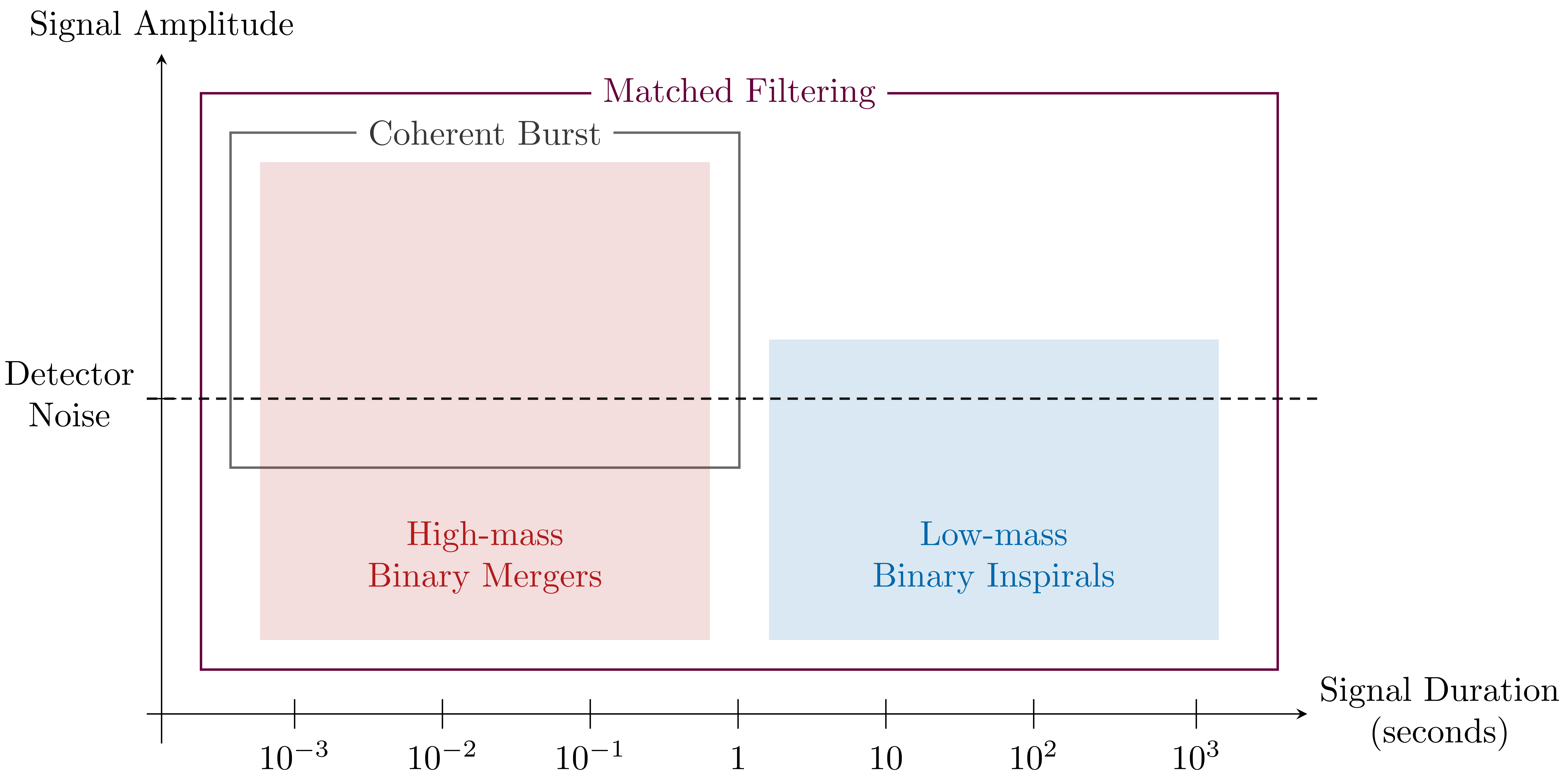
What could current searches be missing?

Horng Sheng Chia

Jay Wadekar

Aaron Zimmerman

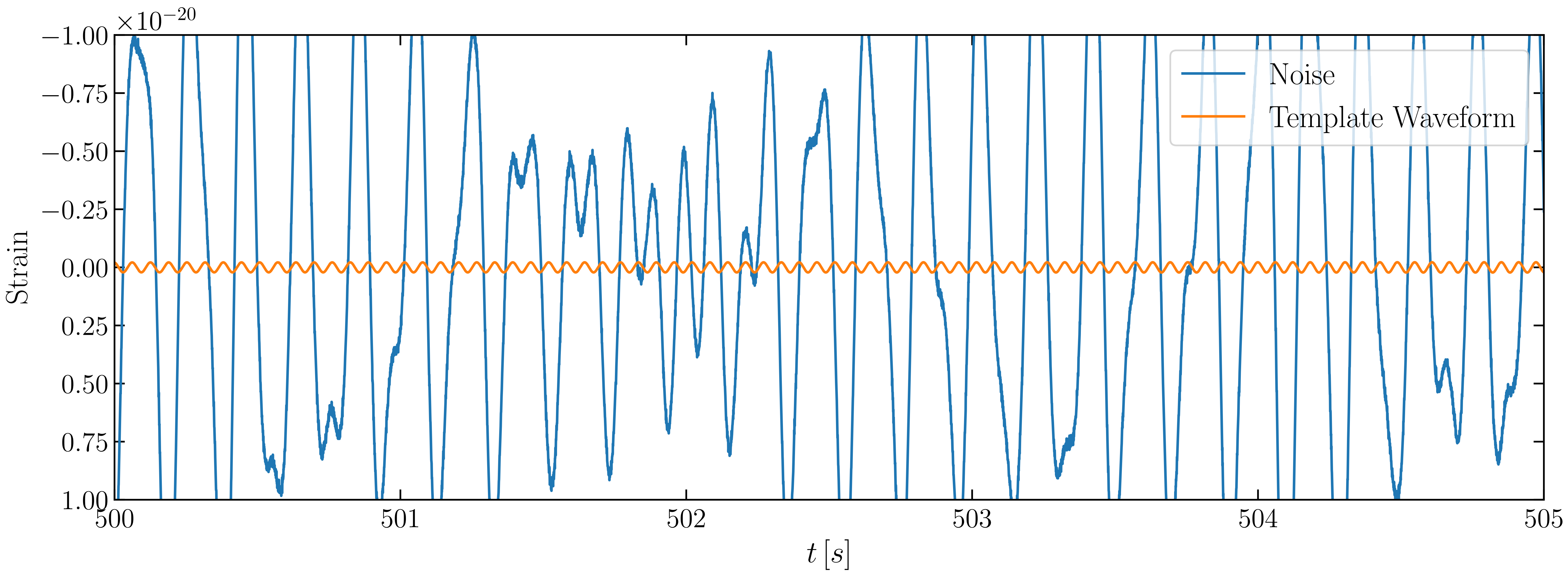
Current Searches



[**TE**, Chia (JCAP): 2004.06729]

Matched Filtering

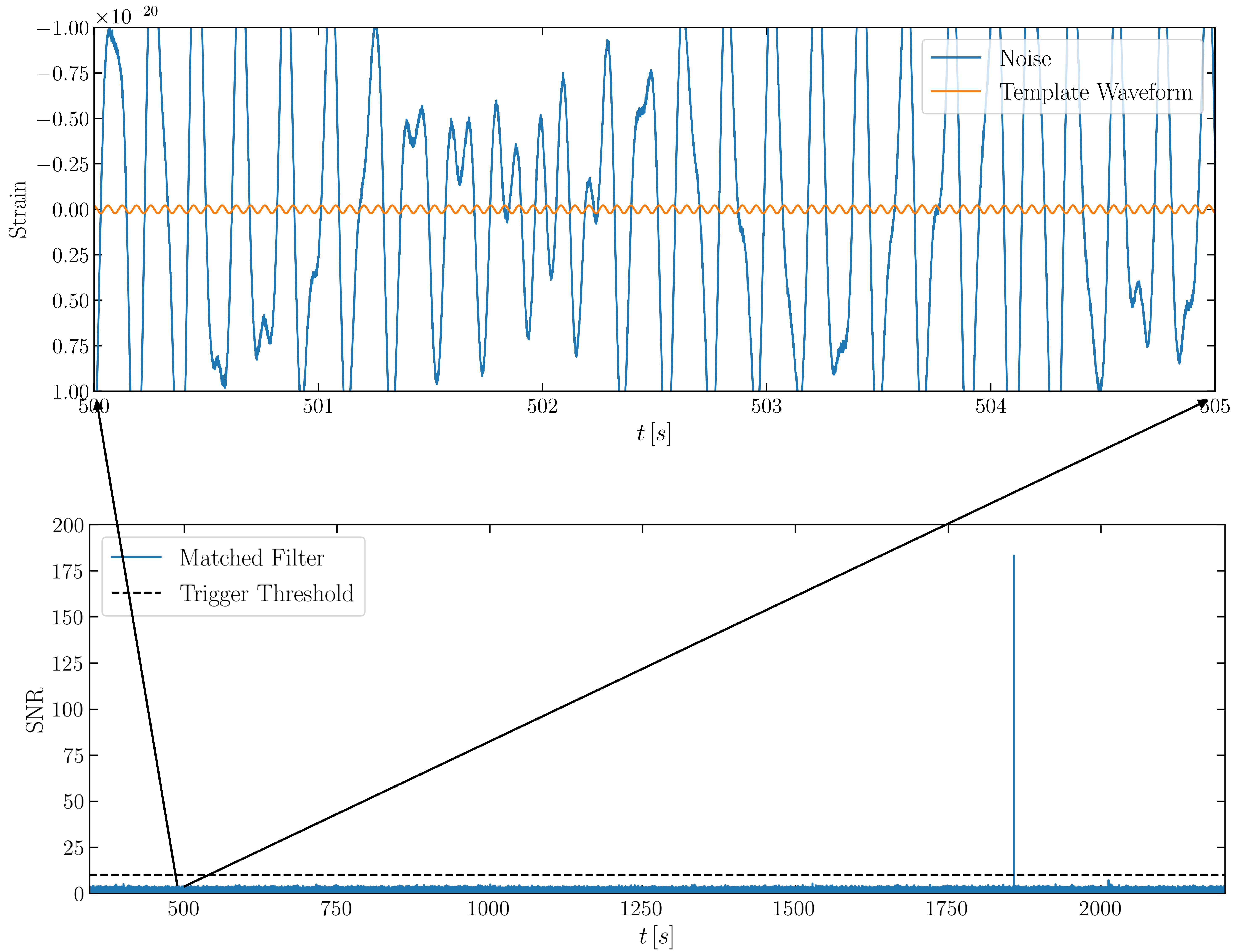
Matched Filtering is the optimum technique for extracting known signals from **stationary Gaussian** noise



Matched Filtering

Matched Filtering is the optimum technique for extracting known signals from **stationary Gaussian** noise

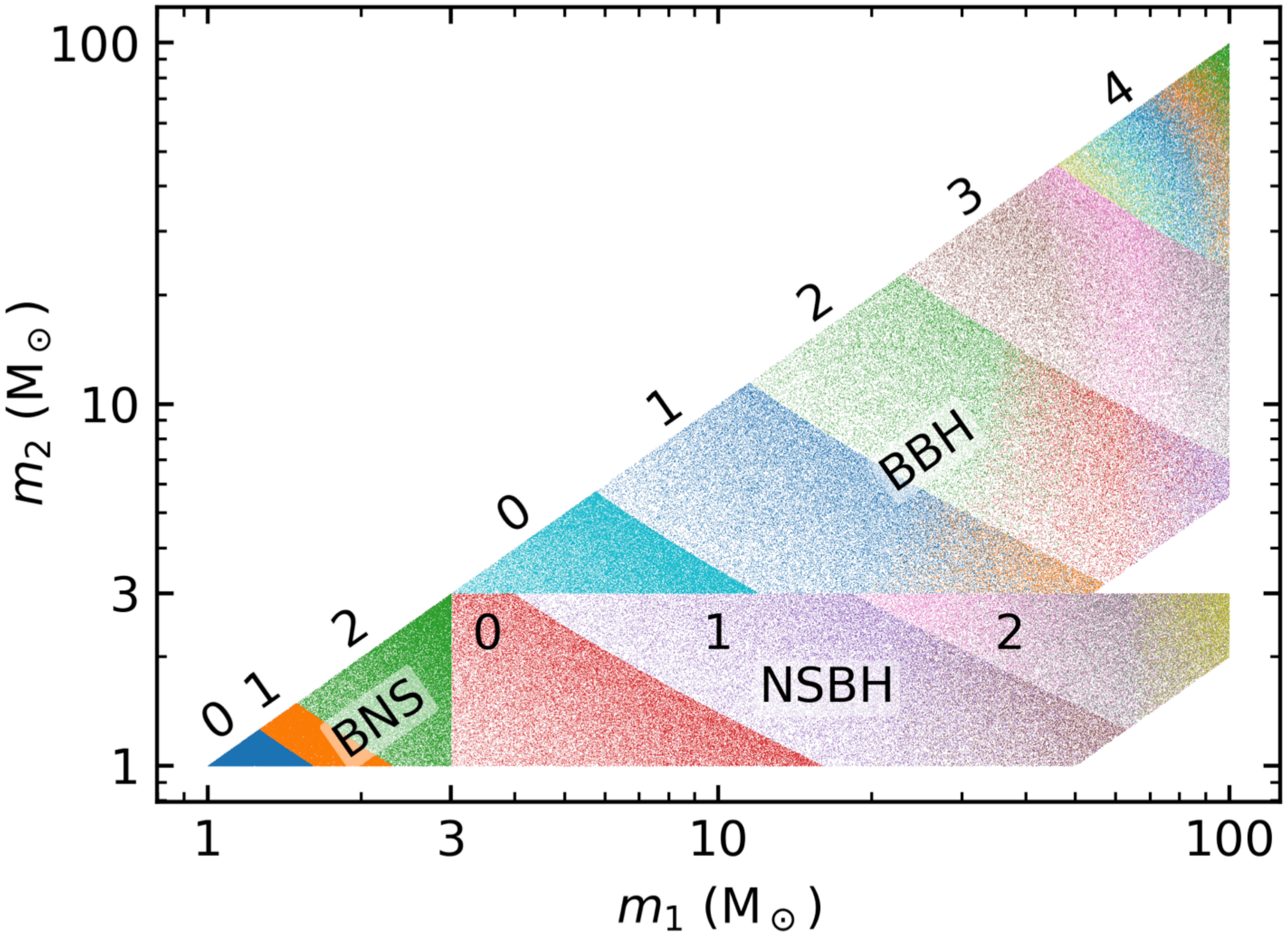
$$(h_1|h_2) \equiv 4\,\mathrm{Re}\int_0^\infty \mathrm{d}f\,\frac{\tilde{h}_1(f)\tilde{h}_2^*(f)}{S_n(f)}$$



Effectualness/fitting-factor

The effectualness is the percentage of SNR retained by a template bank or model

[Roulet et al.: 1904.01683]



$$\varepsilon\left(h_{\text{finite-size}}\right) \equiv \max_{t_c, \phi_c, \boldsymbol{p}_{\text{bbh}}} \left[h_{\text{finite-size}} \mid h\left(\boldsymbol{p}_{\text{bbh}}\right)\right]$$

Finite-Size Effects

$$\tilde{h}(f;\boldsymbol{p}) = \mathcal{A}(\boldsymbol{p})e^{i\phi(f;\boldsymbol{p})}$$

$$\mathcal{A}(f;\boldsymbol{p},\boldsymbol{\lambda}) \propto \mathcal{A}_{\mathrm{PN}}(\boldsymbol{p}) + \mathcal{A}_{\mathrm{Int}}(\boldsymbol{\lambda}) + \mathcal{A}_{\mathrm{MR}}(\boldsymbol{\lambda})$$

$$\phi(f;\boldsymbol{p},\boldsymbol{\lambda}) \propto \phi_{\mathrm{PN}}(\boldsymbol{p}) + \phi_{\mathrm{Int}}(\boldsymbol{\lambda}) + \phi_{\mathrm{MR}}(\boldsymbol{\lambda})$$

Finite-Size Effects

$$\tilde{h}(f;\boldsymbol{p}) = \mathcal{A}(\boldsymbol{p})e^{i\phi(f;\boldsymbol{p})}$$

$$\mathcal{A}(f;\boldsymbol{p},\boldsymbol{\lambda}) \propto \mathcal{A}_{\mathrm{PN}}(\boldsymbol{p}) + \mathcal{A}_{\mathrm{Int}}(\boldsymbol{\lambda}) + \mathcal{A}_{\mathrm{MR}}(\boldsymbol{\lambda})$$

$$\phi(f;\boldsymbol{p},\boldsymbol{\lambda}) \propto \phi_{\mathrm{PN}}(\boldsymbol{p}) + \phi_{\mathrm{Int}}(\boldsymbol{\lambda}) + \phi_{\mathrm{MR}}(\boldsymbol{\lambda})$$

Finite-Size Effects

$$\tilde{h}(f;\boldsymbol{p}) = \mathcal{A}(\boldsymbol{p})e^{i\phi(f;\boldsymbol{p})}$$

$$\mathcal{A}(f;\boldsymbol{p},\boldsymbol{\lambda}) \propto \mathcal{A}_{\rm PN}(\boldsymbol{p}) + \mathcal{A}_{\rm Int}(\boldsymbol{\lambda}) + \mathcal{A}_{\rm MR}(\boldsymbol{\lambda})$$

$$\phi(f;\boldsymbol{p},\boldsymbol{\lambda}) \propto \phi_{\rm PN}(\boldsymbol{p}) + \phi_{\rm Int}(\boldsymbol{\lambda}) + \phi_{\rm MR}(\boldsymbol{\lambda})$$

$$\phi(f;\boldsymbol{p}) = 2\pi f t_c - \phi_c - \frac{\pi}{4} + \frac{3}{128\nu v^5}\left(\phi_{\rm point-particle} + \phi_{\rm finite-size}\right)$$

Finite-Size Effects

$$\tilde{h}(f;\boldsymbol{p}) = \mathcal{A}(\boldsymbol{p}) e^{i\phi(f;\boldsymbol{p})}$$

$$\phi(f;\boldsymbol{p}) = 2\pi f t_c - \phi_c - \frac{\pi}{4} + \frac{3}{128\nu v^5}\left(\phi_{\text{point-particle}} + \phi_{\text{finite-size}}\right)$$

Finite-Size Effects

$$\tilde{h}(f;\boldsymbol{p}) = \mathcal{A}(\boldsymbol{p})e^{i\phi(f;\boldsymbol{p})}$$

$$\phi(f;\boldsymbol{p}) = 2\pi f t_c - \phi_c - \frac{\pi}{4} + \frac{3}{128\nu v^5}\left(\phi_{\text{point-particle}} + \phi_{\text{finite-size}}\right)$$

$$\phi_{\text{point-particle}} \propto \sum_n \phi_n v^n \qquad \phi_{\text{finite-size}} \supset -50\sum_{i=1}^{2}\left(\frac{m_i}{M}\right)^2 \kappa_i \chi_i^2 v^4 - \frac{39\tilde{\Lambda}}{2}v^{10}$$

Finite-Size Effects

$$\tilde{h}(f;\boldsymbol{p}) = \mathcal{A}(\boldsymbol{p})e^{i\phi(f;\boldsymbol{p})}$$

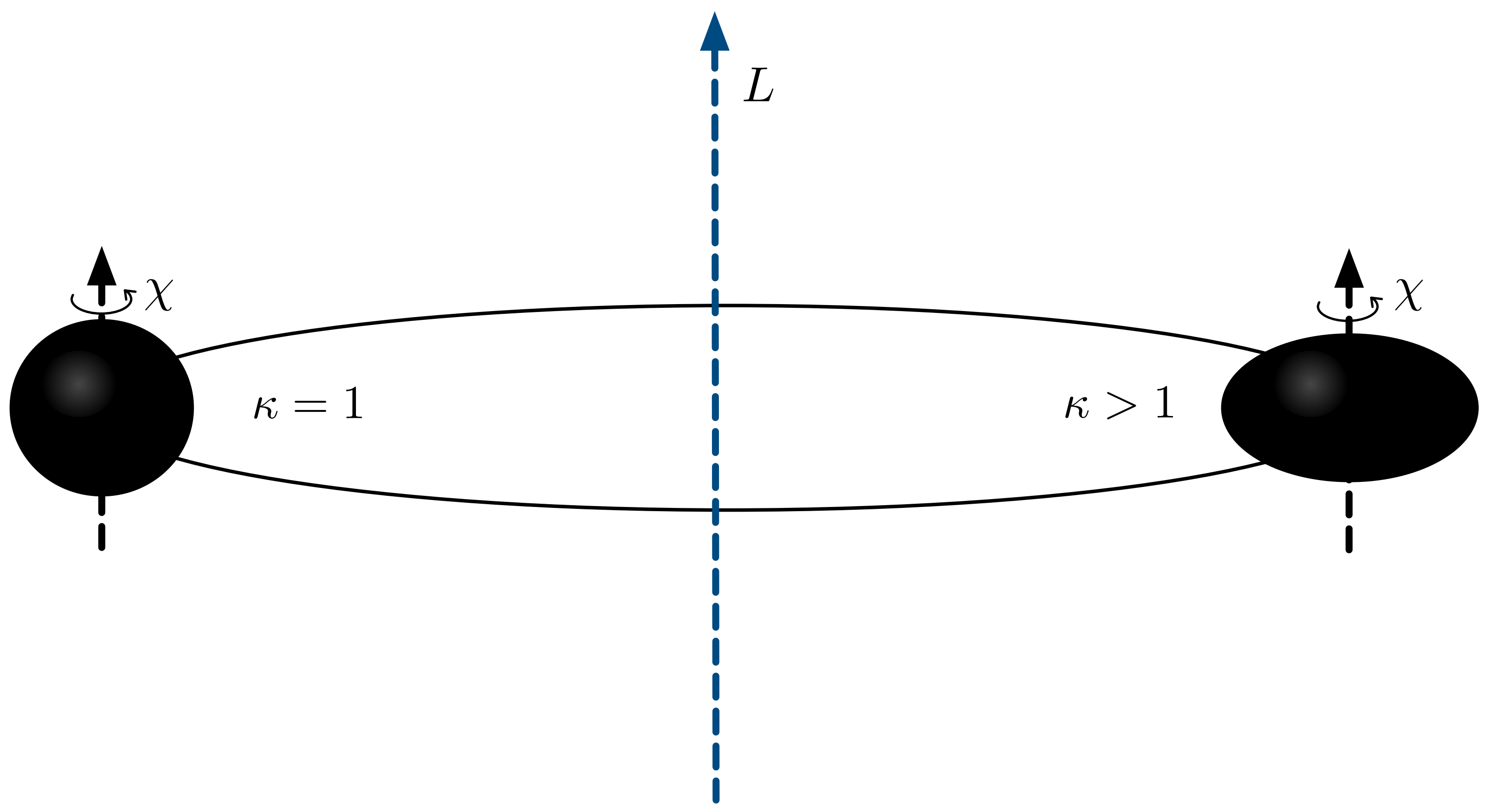
$$\phi_{\mathrm{finite-size}} \supset -50\sum_{i=1}^{2}\left(\frac{m_i}{M}\right)^2 \kappa_i\,\chi_i^2\,v^4 - \frac{39\,\tilde{\Lambda}}{2}v^{10}$$

Finite-Size Effects

$$\tilde{h}(f;\boldsymbol{p}) = \mathcal{A}(\boldsymbol{p})e^{i\phi(f;\boldsymbol{p})}$$

$$\phi_{\rm finite-size} \supset -50\sum_{i=1}^{2}\left(\frac{m_i}{M}\right)^2 \kappa_i \chi_i^2 v^4 - \frac{39\tilde{\Lambda}}{2}v^{10}$$

Spin Induced Quadruple

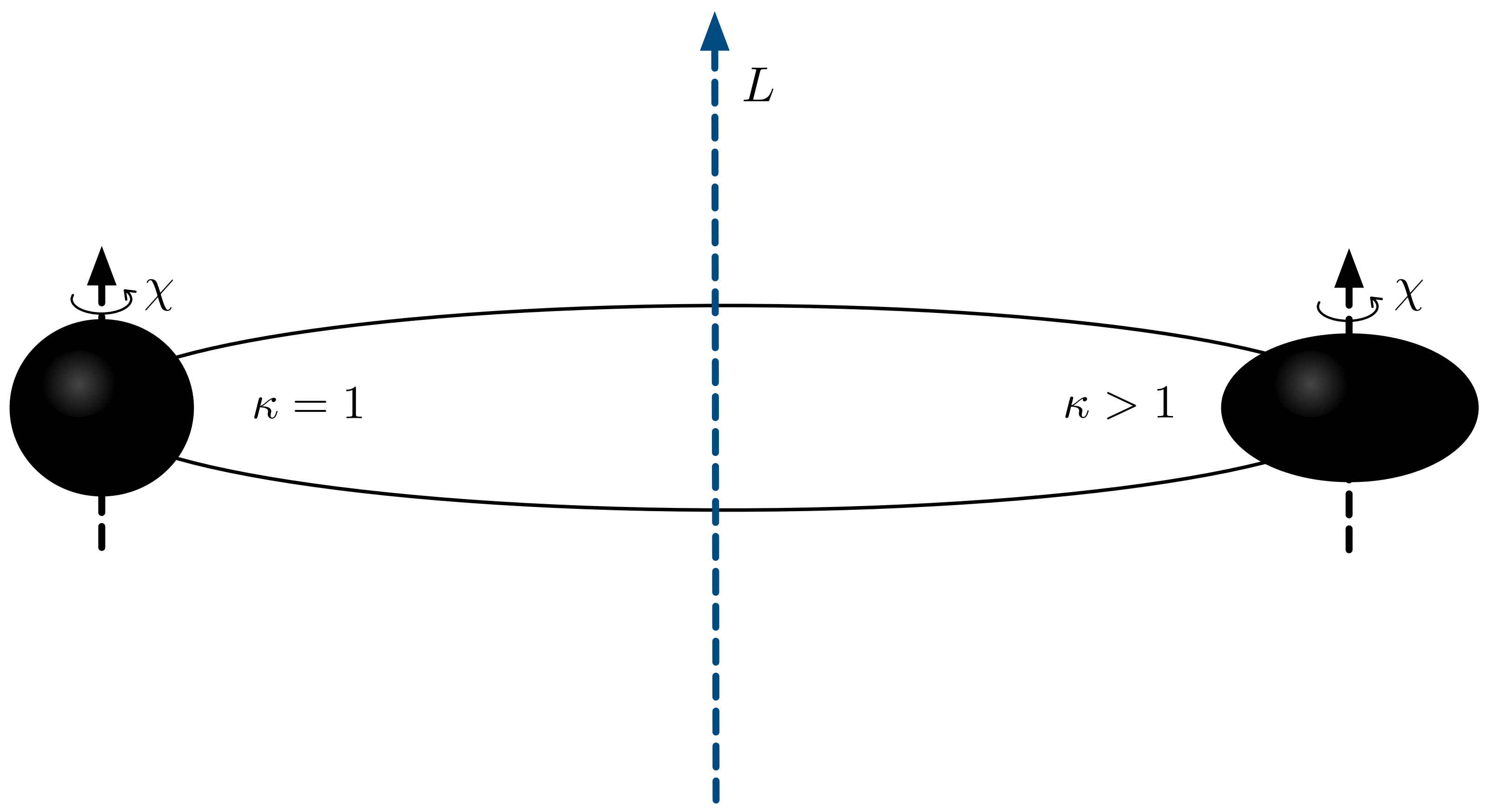


Finite-Size Effects

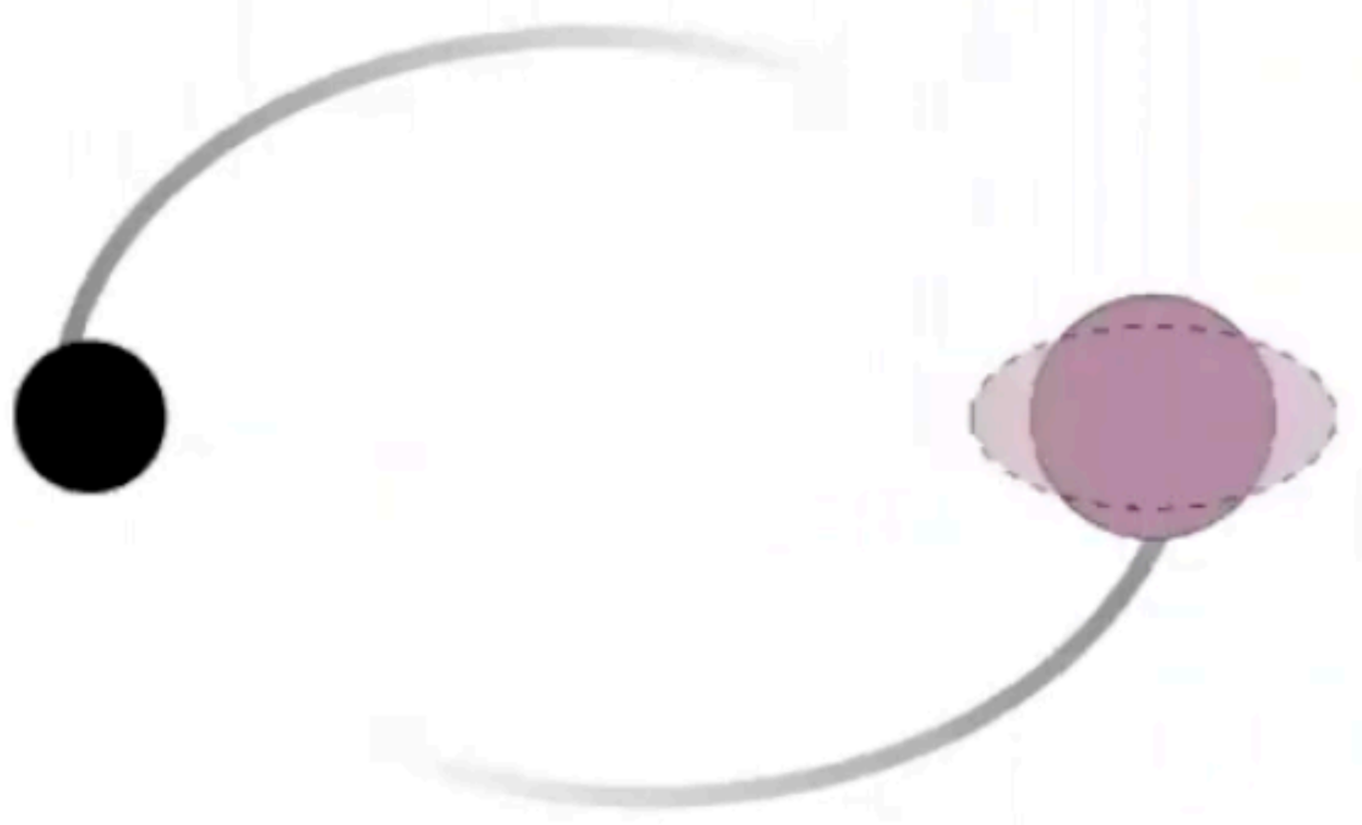
$$\tilde{h}(f;\boldsymbol{p}) = \mathcal{A}(\boldsymbol{p})e^{i\phi(f;\boldsymbol{p})}$$

$$\phi_{\mathrm{finite-size}} \supset -50\sum_{i=1}^{2}\left(\frac{m_i}{M}\right)^2 \kappa_i\,\chi_i^2\,v^4 - \frac{39\,\tilde{\Lambda}}{2}v^{10}$$

Spin Induced Quadruple



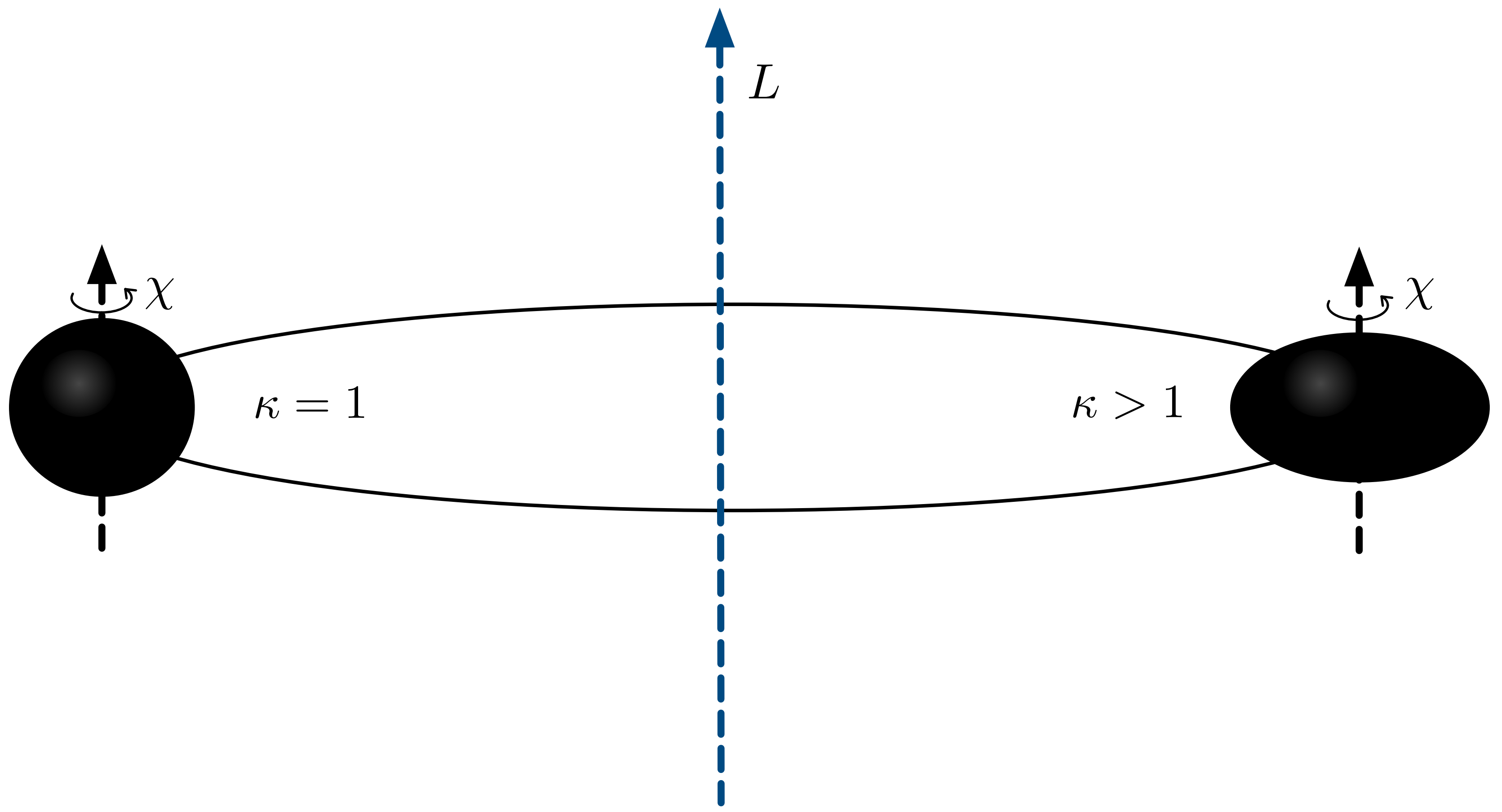
Tidal Love Number

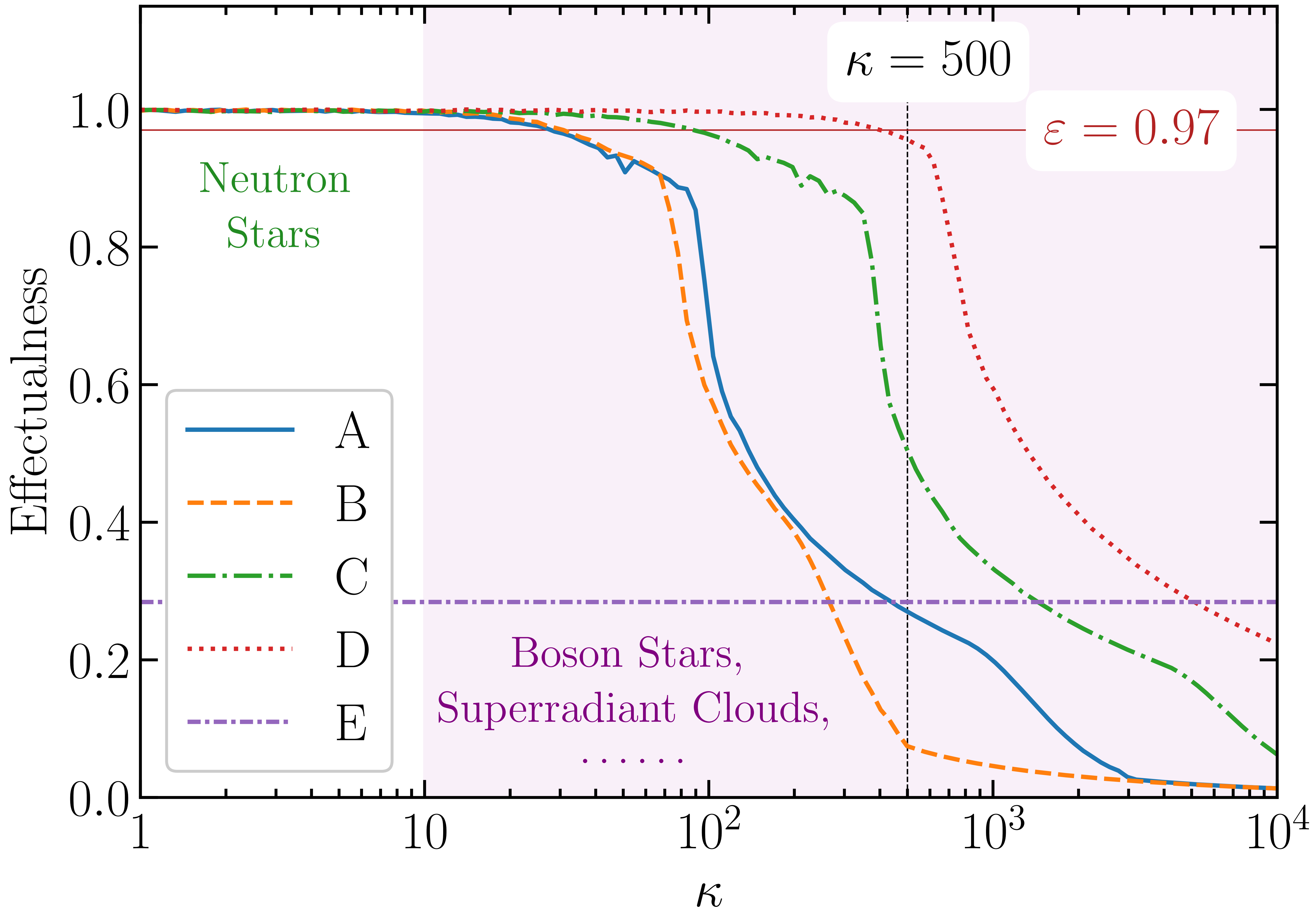


Finite-Size Effects

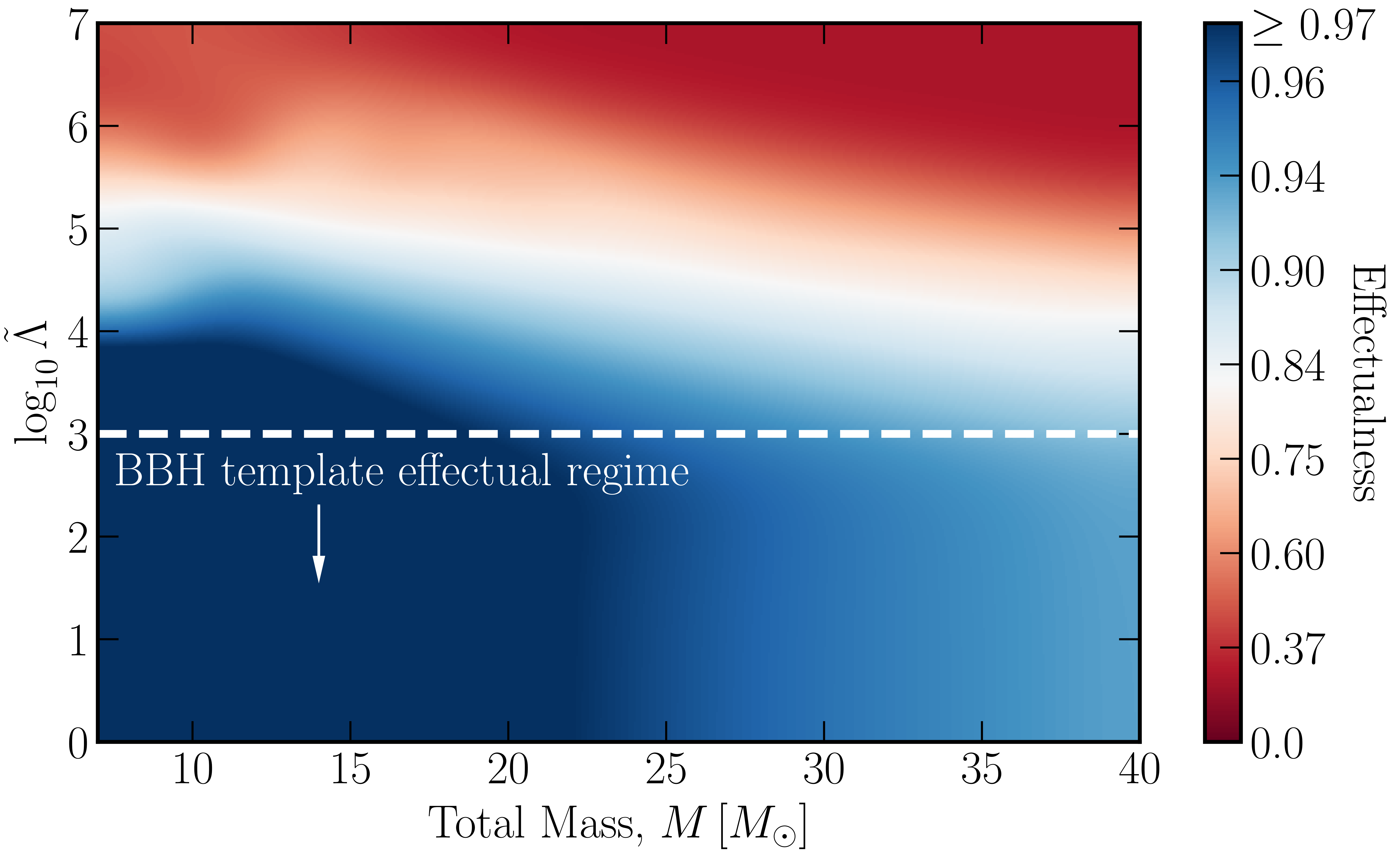
Spin-Induced Quadrupoles

Kappa controls the axisymmetric response of an object to spin *i.e.*, how oblate it becomes





Tidal Love Numbers



[Horng Sheng Chia, **TE**, Digvijay Wadekar, Aaron Zimmerman et al.: 2306.00050]

Why would Love Numbers be Large?

$$\Lambda = \frac{2}{3}k\left(\frac{r}{m}\right)^5$$

[Kent Yagi, Nicolas Yunes: 1303.1528]

Why would Love Numbers be Large?

$$\Lambda = \frac{2}{3} \boxed{k} \left(\frac{r}{m} \right)^5$$

▢ = O(1) coefficient

[Kent Yagi, Nicolas Yunes: 1303.1528]

Why would Love Numbers be Large?

$$\Lambda = \frac{2}{3} k \left(\frac{r}{m} \right)^5$$

▢ = O(1) coefficient

▢ = Object's radius

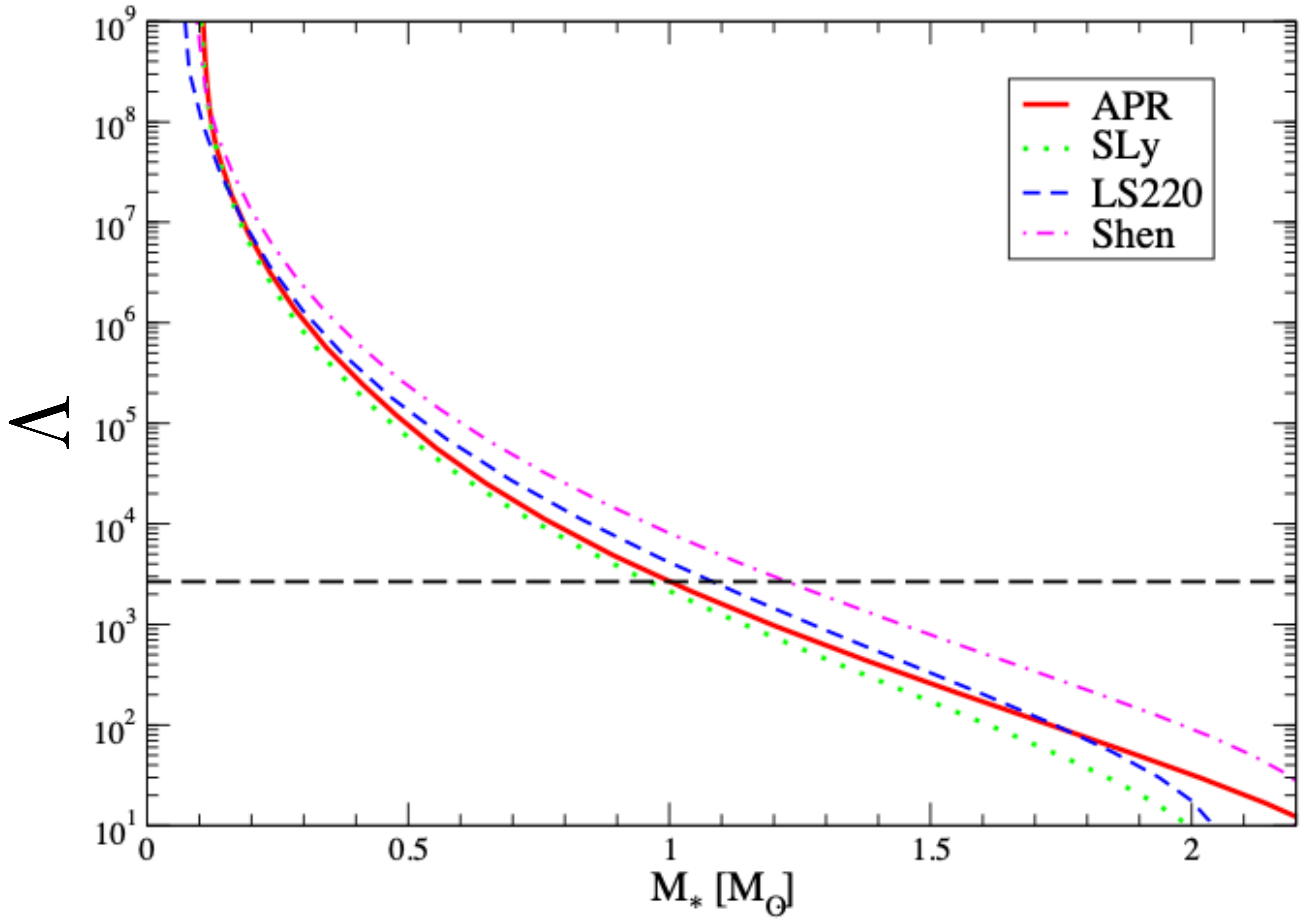
[Kent Yagi, Nicolas Yunes: 1303.1528]

Why would Love Numbers be Large?

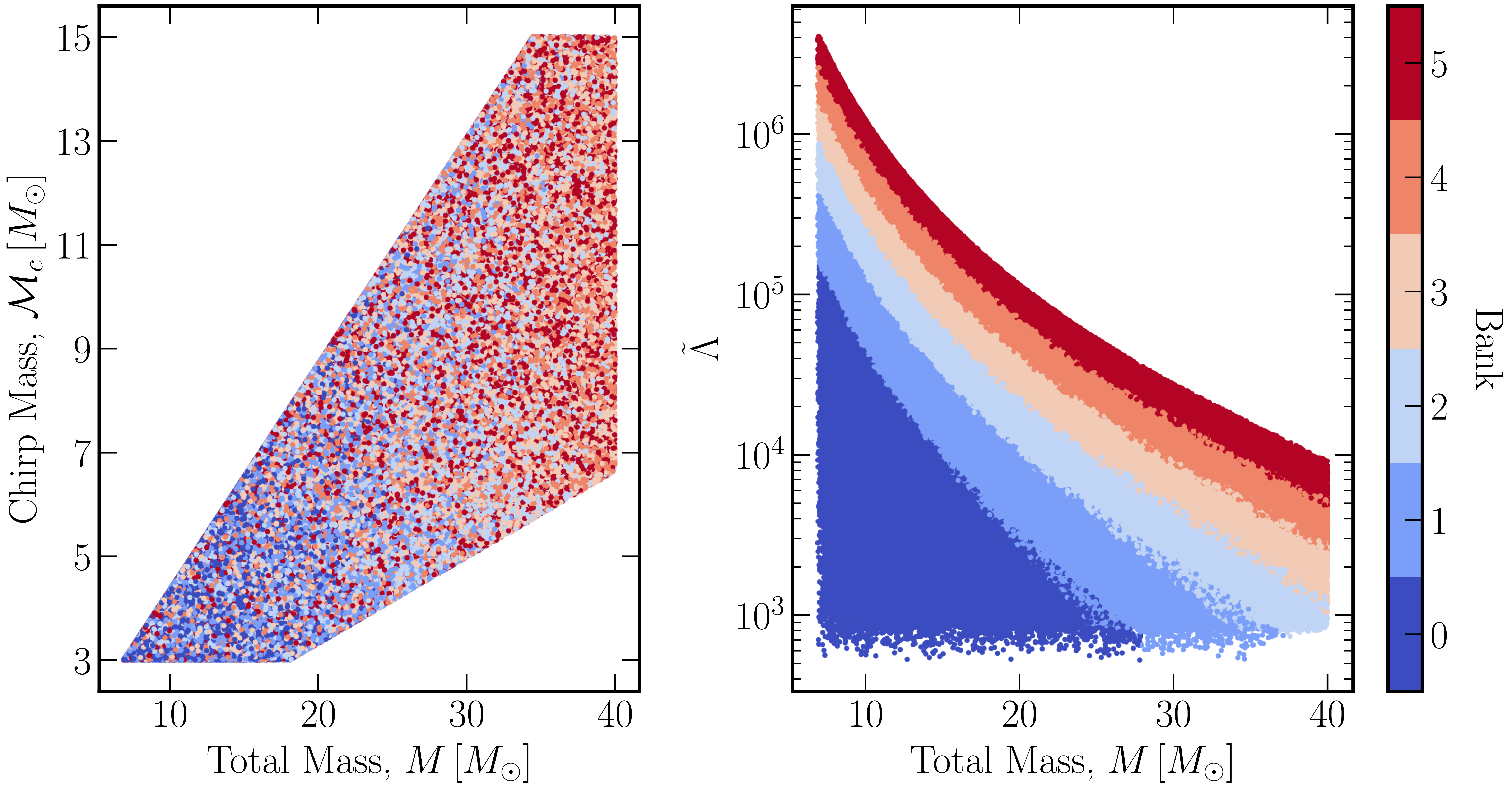
$$\Lambda = \frac{2}{3} k \left(\frac{r}{m} \right)^5$$

▢ = O(1) coefficient

▢ = Object's radius



Template Bank



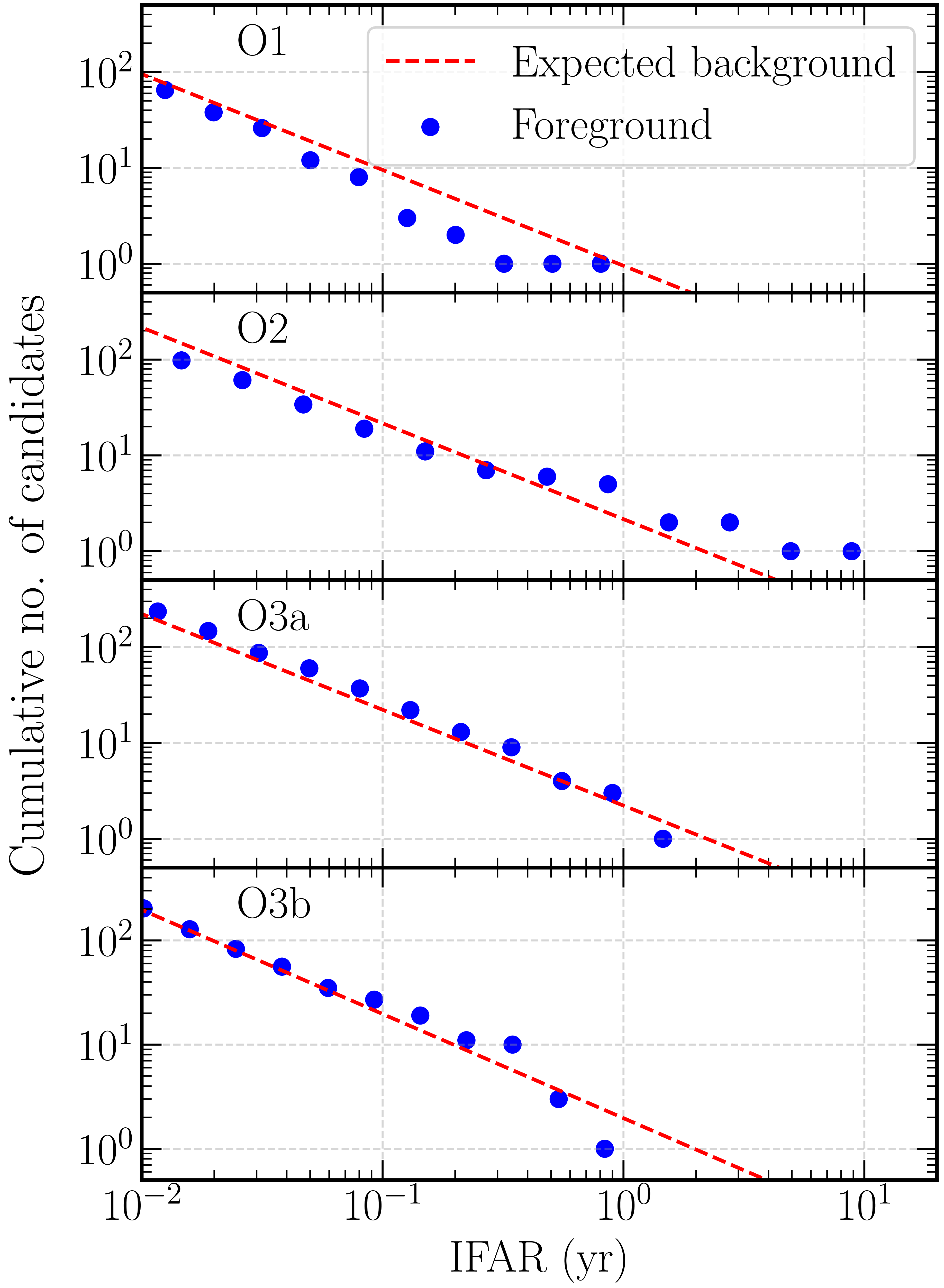
Anything outside this space we cannot guarantee coverage

[Roulet et al.: 1904.01683]

Null Results

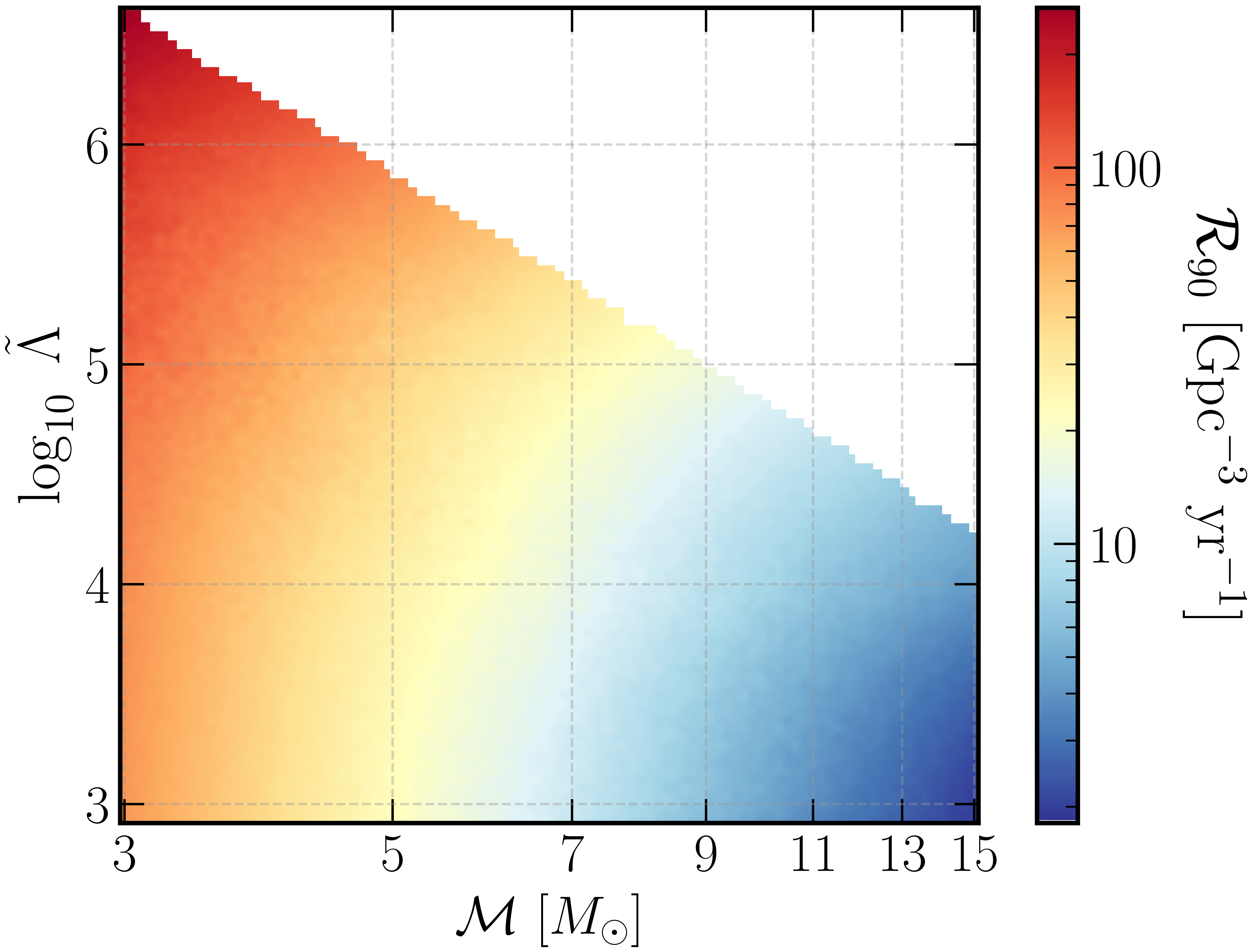
We performed the first inspiral-only search for signals that are **not** aligned-spin BBHs

No significant candidates found



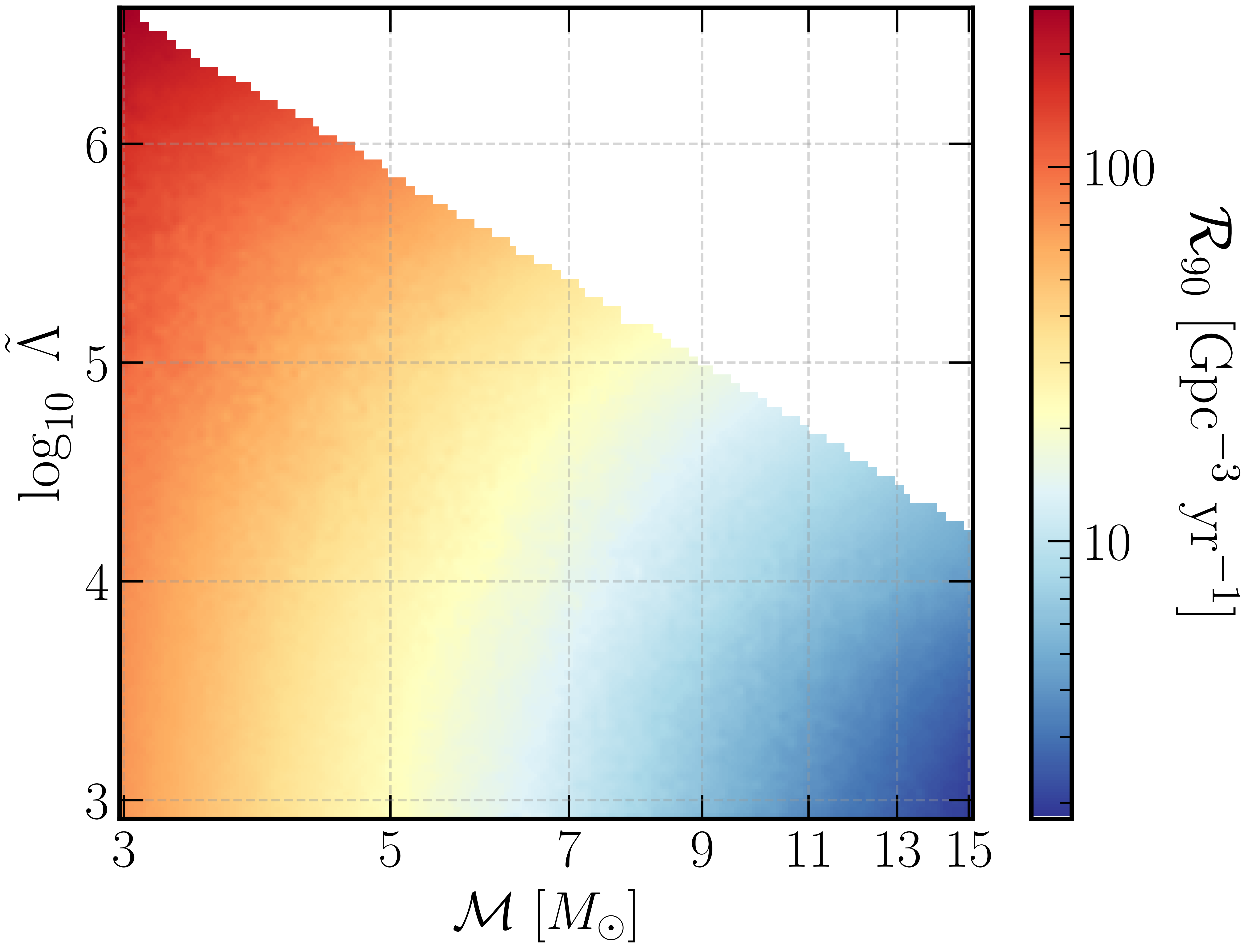
[Horng Sheng Chia, **TE**, Digvijay Wadekar, Aaron Zimmerman et al.: 2306.00050]

Rate Constraints



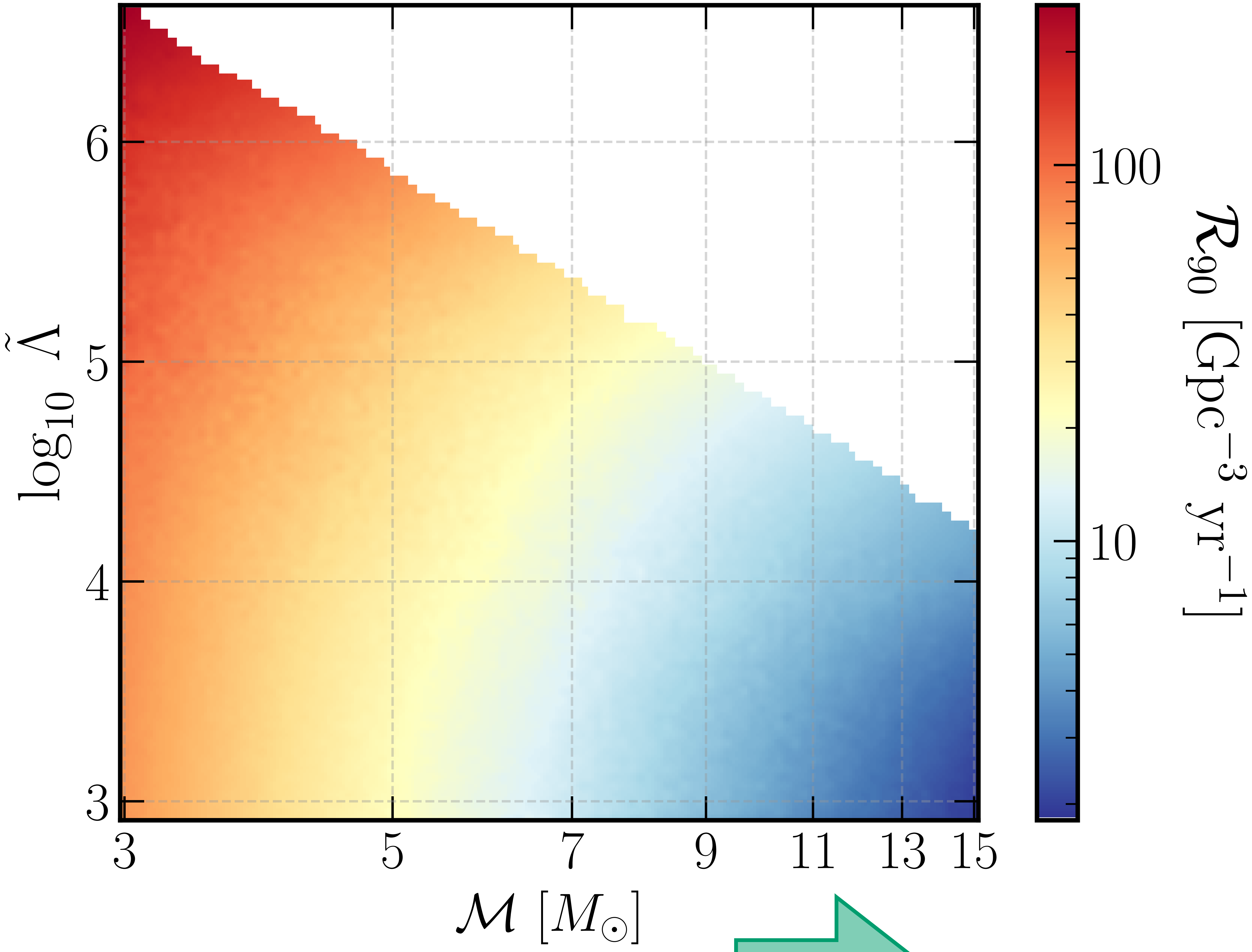
Rate Constraints

Constraints get weaker, due to shorter signal



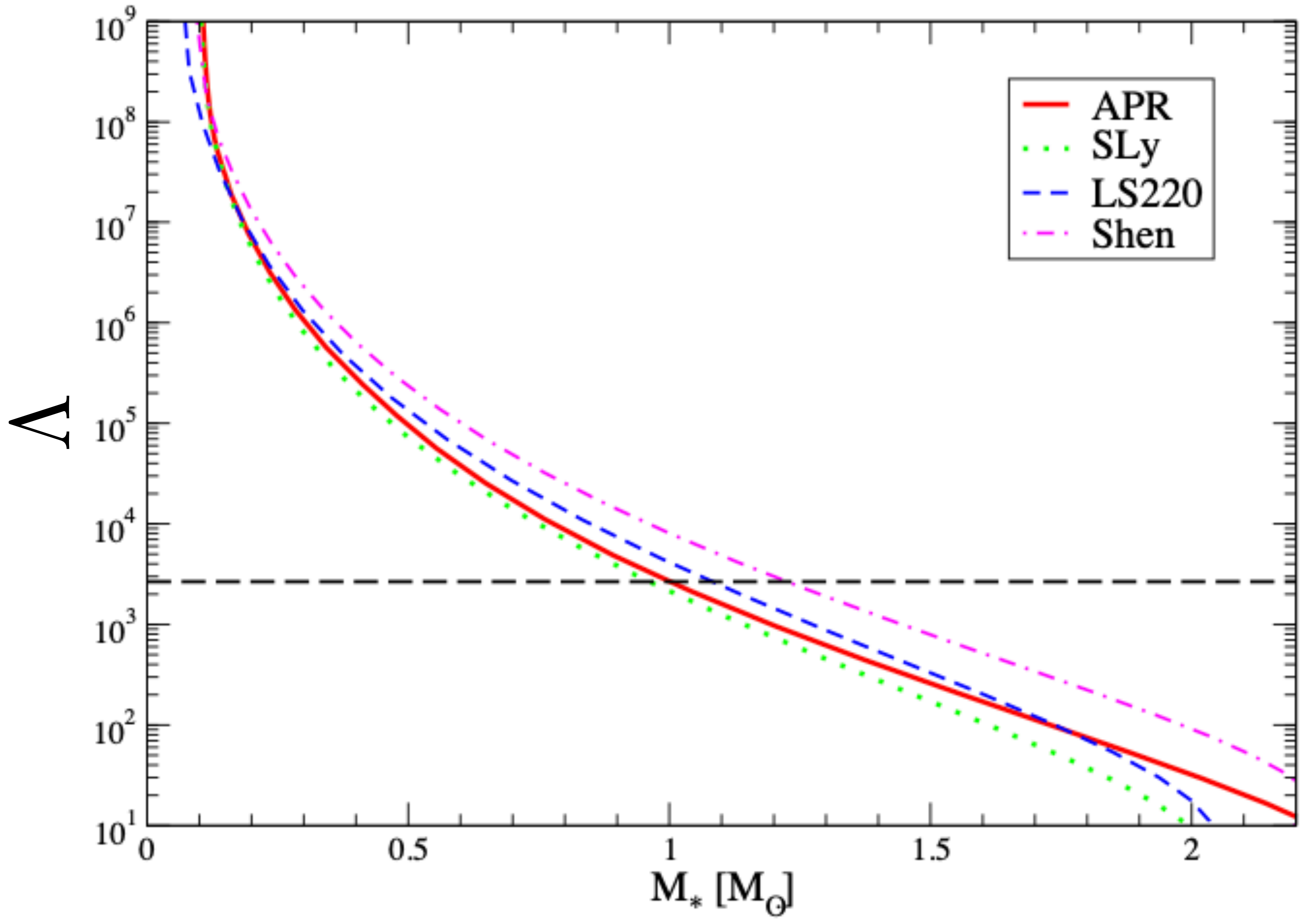
Rate Constraints

Constraints get weaker, due to shorter signal



Constraints get stronger as higher chirp masses produce higher SNR signals

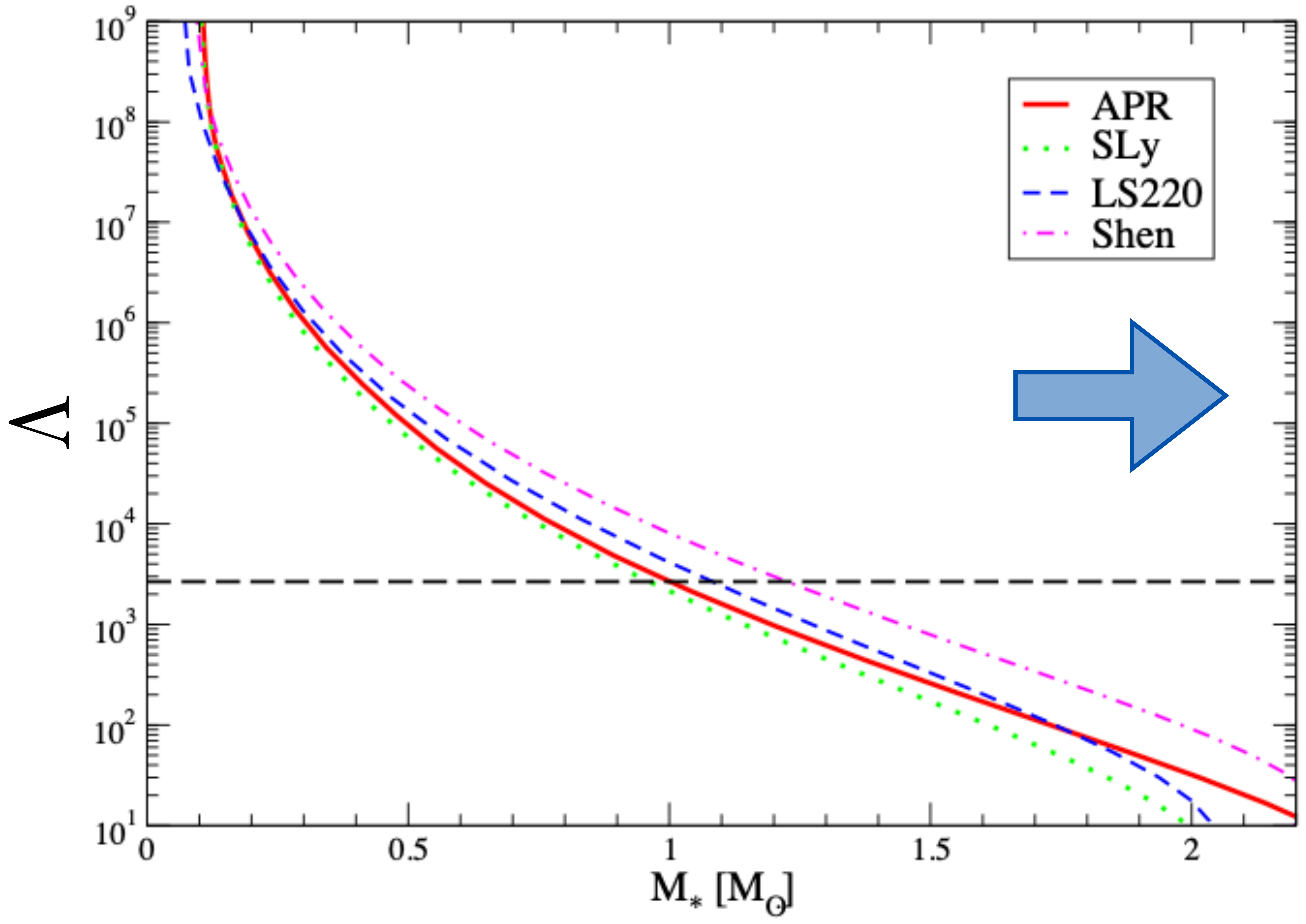
Future Searches



[Kent Yagi, Nicolas Yunes: 1303.1528]

Future Searches

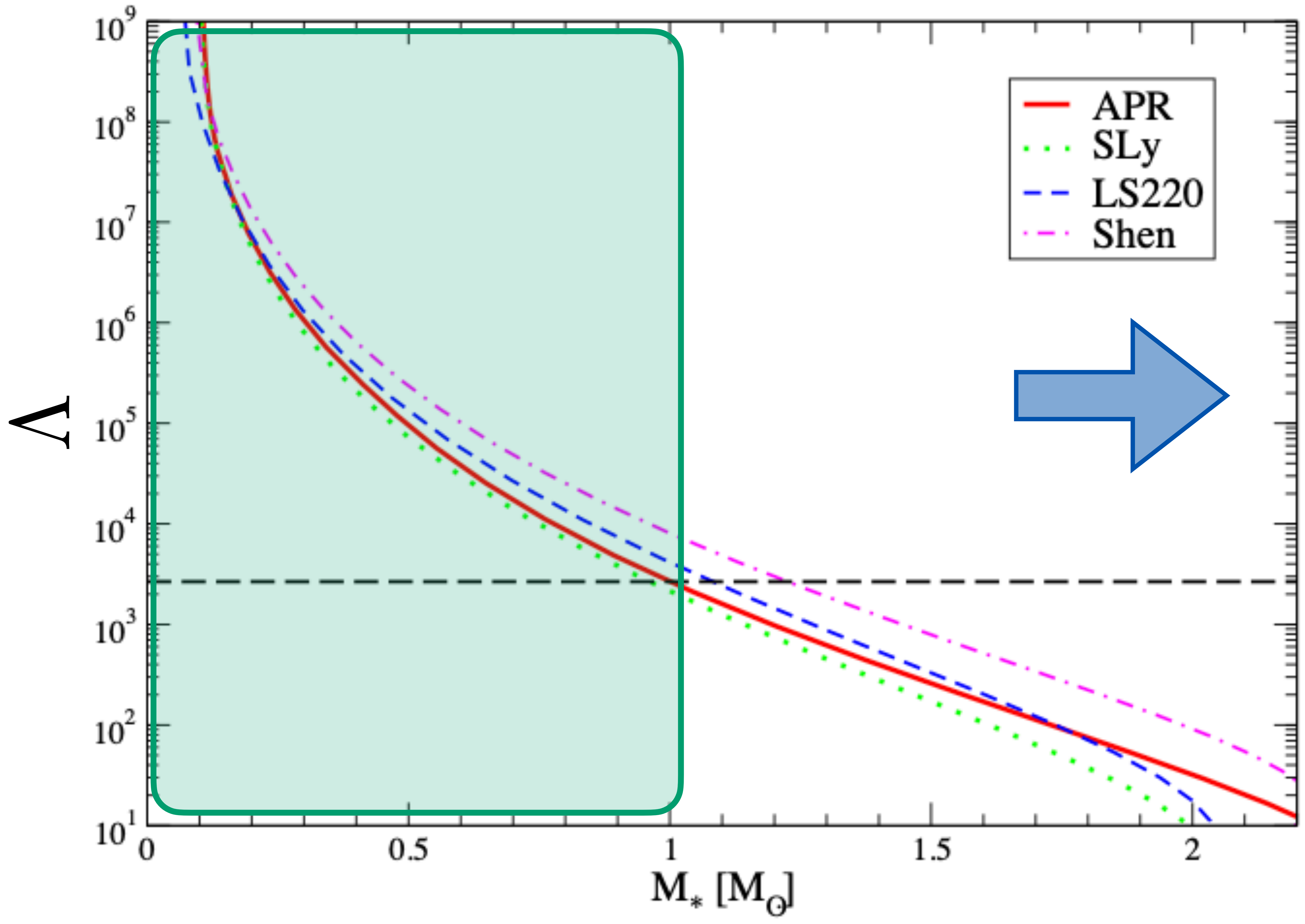
We have currently search in the high-mass region of parameter space due to computational limitations



Future Searches

We have currently search in the high-mass region of parameter space due to computational limitations

Future searchers should focus on the low-mass region, where neutron stars may be hiding



Conclusion and Next Steps

Conclusion and Next Steps

Differentiable models can contribute to many aspects of the GW data analysis pipeline

Conclusion and Next Steps

Differentiable models can contribute to many aspects of the GW data analysis pipeline

- Waveform developers should start to use pure python (or JAX) for public release
- Real time PE processing could potentially replace matched filtering search pipelines

Conclusion and Next Steps

Differentiable models can contribute to many aspects of the GW data analysis pipeline

- Waveform developers should start to use pure python (or JAX) for public release
- Real time PE processing could potentially replace matched filtering search pipelines

Searches, especially at low masses, are currently limited by their template bank coverage

Conclusion and Next Steps

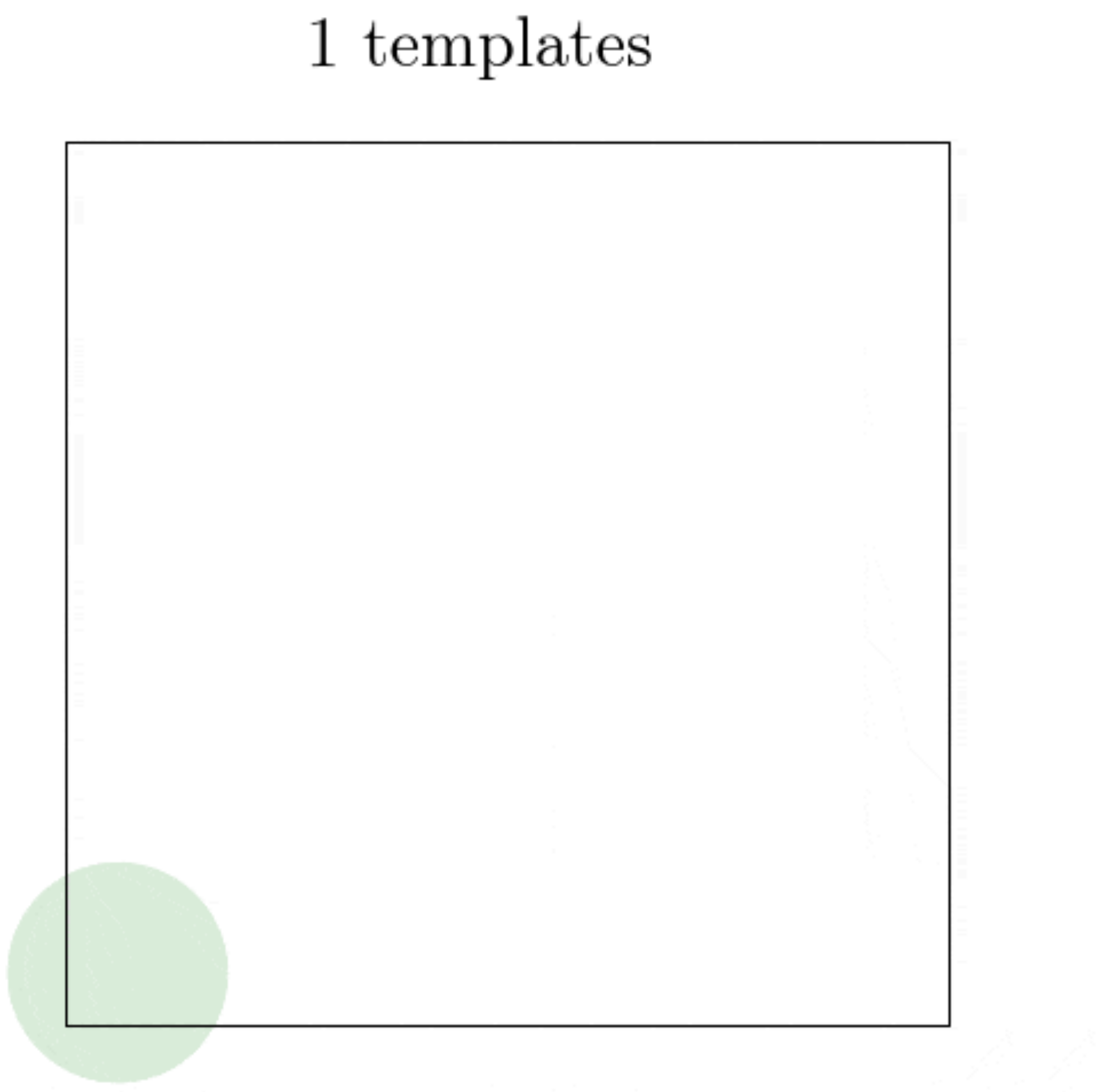
Differentiable models can contribute to many aspects of the GW data analysis pipeline

- Waveform developers should start to use pure python (or JAX) for public release
- Real time PE processing could potentially replace matched filtering search pipelines

Searches, especially at low masses, are currently limited by their template bank coverage

- Many more searches to be done in order to fully utilize the GW data we have
- Low-mass neutron stars are a great target for a future Love numbers search

Backup - Template Banks



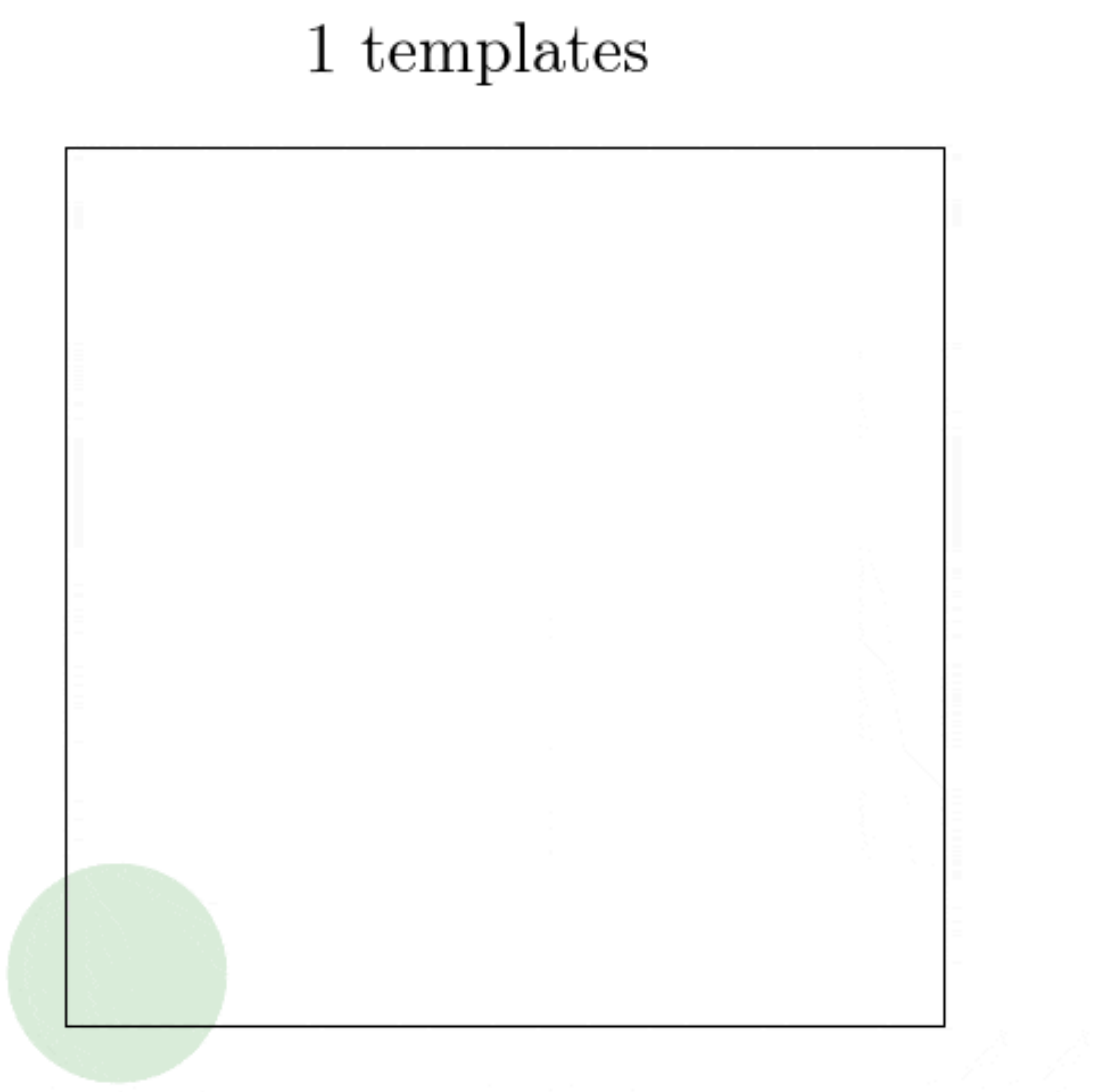
- Quick to generate
- Can achieve ~100% coverage
- Placement of points in highly curved spaces unknown or difficult
- Require a known *metric*

[Cokelaer: 0706.4437]

[Owen: 9511032]

[Owen and Sathyaprakash: 9808076]

Backup - Template Banks



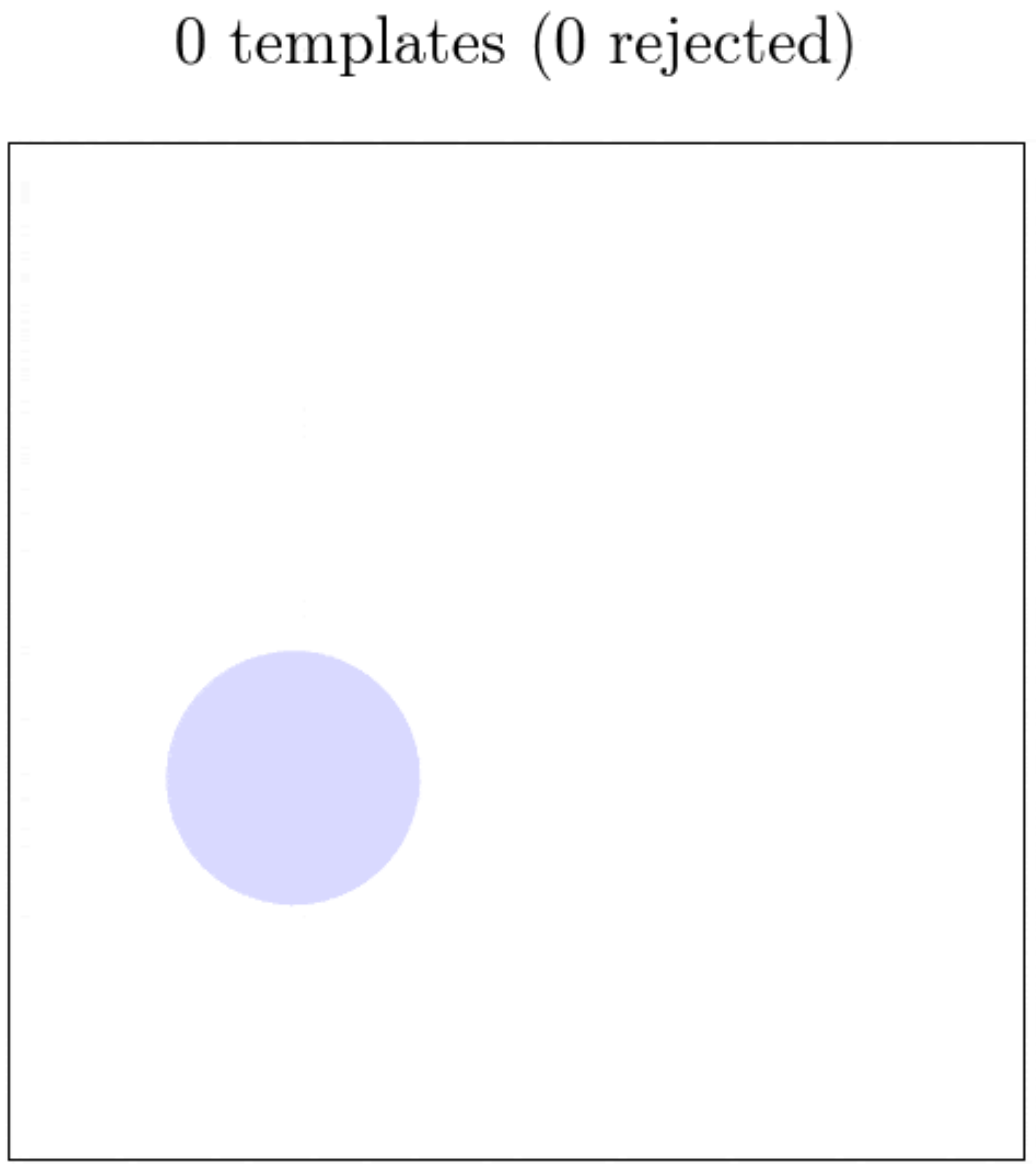
- Quick to generate
- Can achieve ~100% coverage
- Placement of points in highly curved spaces unknown or difficult
- Require a known *metric*

[Cokelaer: 0706.4437]

[Owen: 9511032]

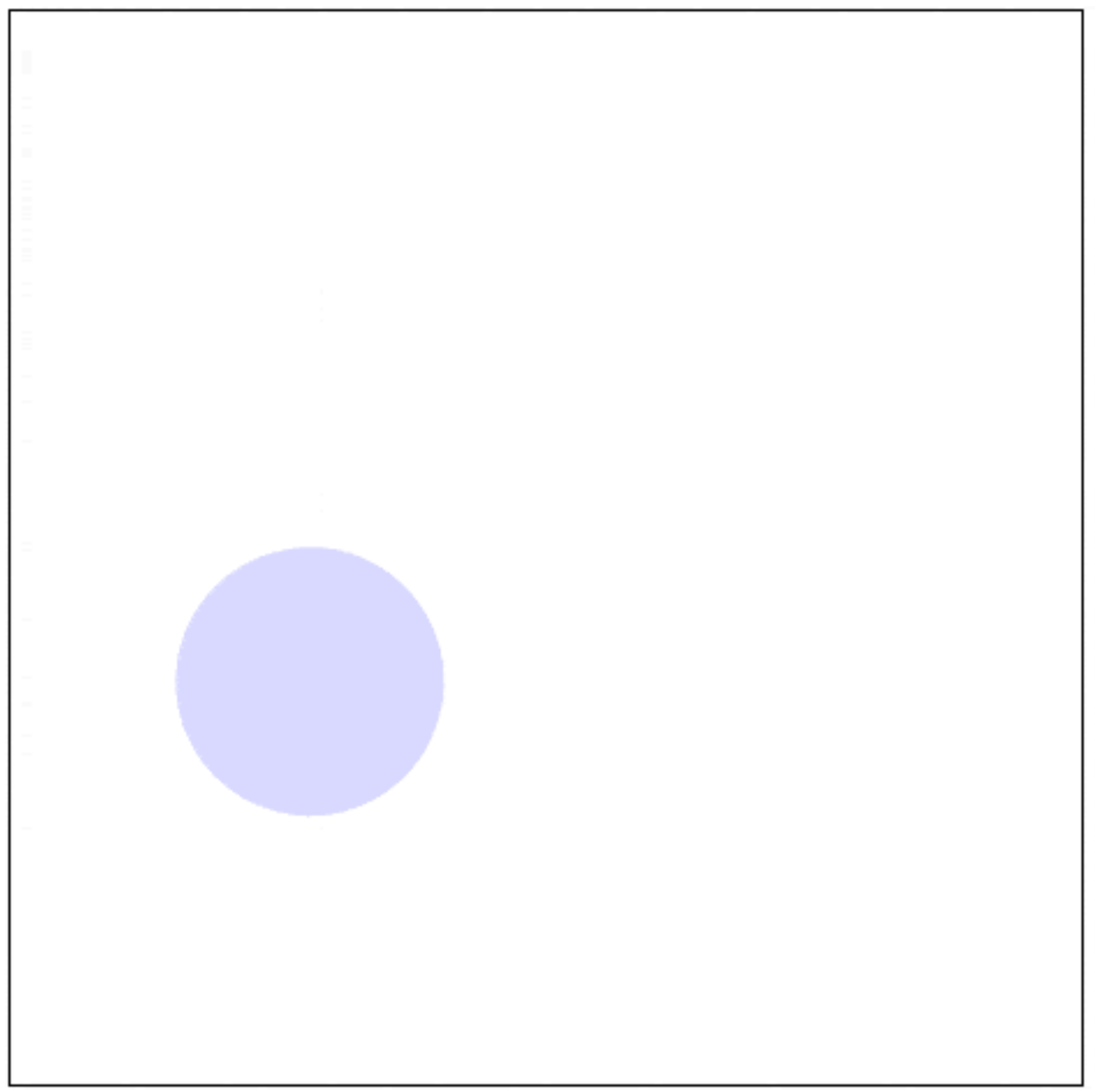
[Owen and Sathyaprakash: 9808076]

Backup - Template Banks



- Does **not** require a metric
- Can be very slow to converge

Backup - Template Banks

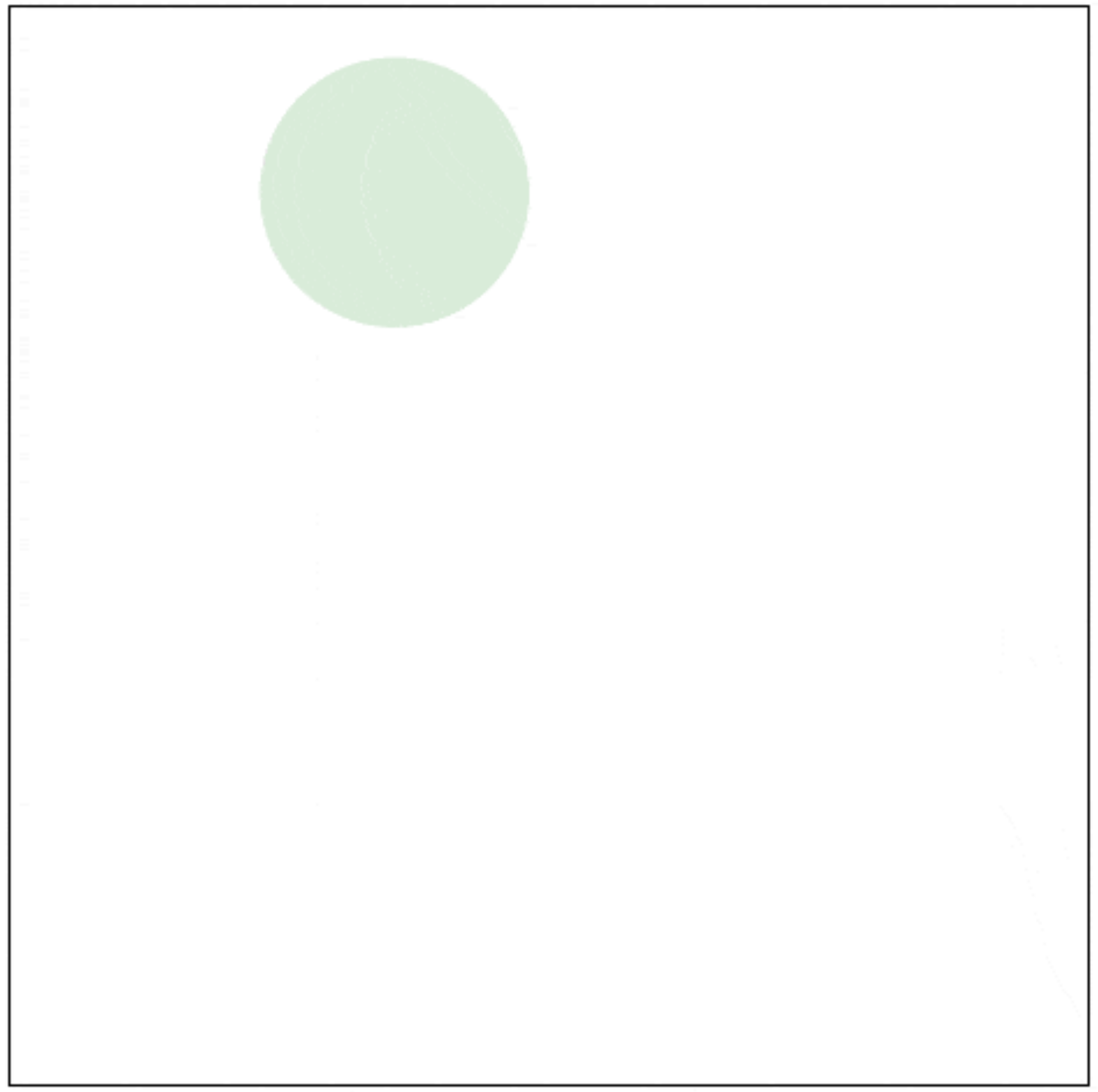


- Does **not** require a metric
- Can be very slow to converge

[Harry, Allen, and Sathyaprakash: 0908.2090]

Backup - Template Banks

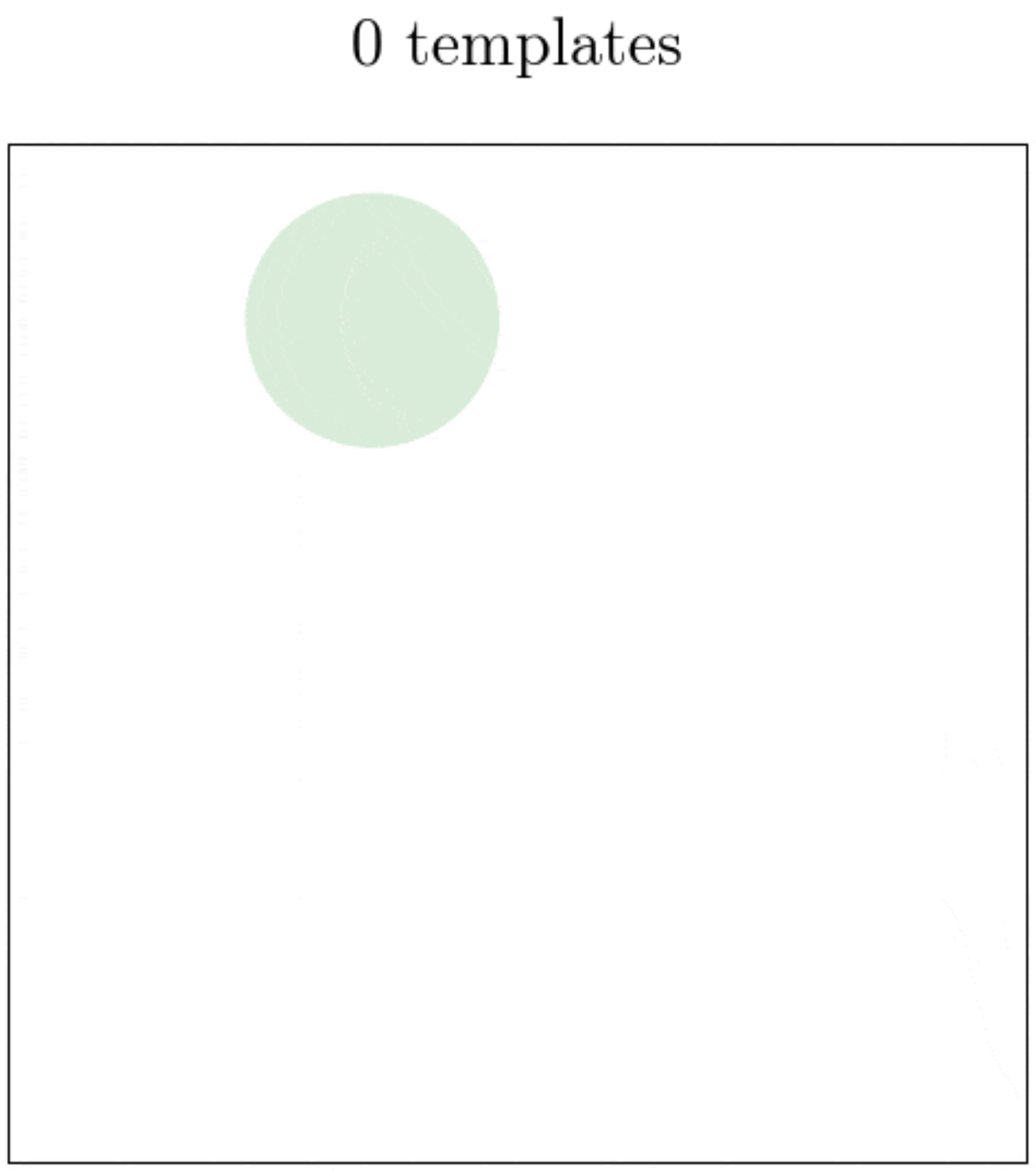
0 templates



Random Placement:

- Quick to generate
- Over covers the parameter space (in low dimensions)
- Requires a metric

Backup - Template Banks



Random Placement:

- Quick to generate
- Over covers the parameter space (in low dimensions)
- Requires a metric

[Messenger: 0809.5223]

Backup