

LASER INTERFEROMETER GRAVITATIONAL WAVE OBSERVATORY
- LIGO -
CALIFORNIA INSTITUTE OF TECHNOLOGY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Technical Note	LIGO-T2600158-vX	2026/07/03
Validating LIGO Interferometer Models		
Abhiram Sharma		

California Institute of Technology
LIGO Project, MS 18-34
Pasadena, CA 91125
Phone (626) 395-2129
Fax (626) 304-9834
E-mail: info@ligo.caltech.edu

Massachusetts Institute of Technology
LIGO Project, Room NW22-295
Cambridge, MA 02139
Phone (617) 253-4824
Fax (617) 253-7014
E-mail: info@ligo.mit.edu

LIGO Hanford Observatory
Route 10, Mile Marker 2
Richland, WA 99352
Phone (509) 372-8106
Fax (509) 372-8137
E-mail: info@ligo.caltech.edu

LIGO Livingston Observatory
19100 LIGO Lane
Livingston, LA 70754
Phone (225) 686-3100
Fax (225) 686-7189
E-mail: info@ligo.caltech.edu

<http://www.ligo.caltech.edu/>

1 Introduction

Gravitational waves (GWs) are distortions in the fabric of space-time, first predicted by Albert Einstein in 1915 as a result of general relativity. These waves propagate from energetic physical processes (colliding black holes, supernovae, colliding neutron stars) at the speed of light, carrying information about the events that created them. GWs are difficult to measure due to the minimal spatial distortion (strain) that they impose on Earth’s surface, on the order of 10^{-21} . The Laser Interferometer Gravitational-Wave Observatory (LIGO) was the first experiment to observe a GW in 2015 [1].

To reach a strain sensitivity capable of observing gravitational waves, LIGO must overcome many sources of noise including but not limited to: seismic noise, optical vacuum fluctuations, cosmic rays, and thermal noise [2]. This project will focus on simulating how mirror thermal fluctuations affect observed optical signals, in order to reproduce experimental data.

While previous studies have been done on the theoretical modeling of mirror thermal fluctuations [3, 4], there is still a mismatch between simulation predictions and experimental measurements. This is believed to originate from the nonstationary evolution of thermal distortions in the optics, due to incomplete knowledge of their initial thermal state and incomplete modeling of their thermal dynamics, which can introduce time-dependent systematic errors into the calibration and optical modes [5].

This project aims to construct accurate models of the interferometer that capture these thermal effects, and also utilize MCMC algorithms to analyze parameter sensitivity within the interferometer. The predictions of this project’s optical model can be validated against real LIGO data, specifically from stretches of data where the interferometer remains locked and the mirrors are allowed to reach a thermal steady-state. Data that satisfies these criteria is not trivial to obtain, as whenever significant transient noise is introduced into the system, the laser will not be locked. Ultimately, these optical simulations will improve our understanding of thermal effects in LIGO and may enable the development of more robust active thermal compensation techniques, mitigating the impact of thermal noise and allowing for even higher strain sensitivity needed for future gravitational wave detectors.

2 Background

As outlined in the proposal, the first objectives to work on were developing a realistic optical model of the interferometer optics, incorporating thermal effects into the model, and implementing MCMC algorithms into the simulation. A previously developed Python script that models the LIGO Livingston interferometer has been used by Professor Richardson’s group. This model accounts for thermal steady-state effects. This only leaves the MCMC implementation.

For the MCMC portion, I began with parameter recovery studies for FINESSE simulations of the LIGO Livingston Observatory (LLO) [6]. The workflow for this process is as follows:

1. Generate ground truth data from a base FINESSE model. This outputs a vector $\vec{d}$

where each index is a value measured by a power detector in the model.

2. Add random noise to each individual index in this vector to emulate experimental measurement noise and to prevent the algorithm from only returning the initial parameter values for each of its samples. Call this $\vec{d}^*$.
3. Choose n parameters of interest so that we can construct a parameter vector θ which lives in an n -dimensional parameter space.
4. Using an MCMC algorithm of choice, we can estimate the distribution of θ according to Bayes' theorem:

$$P(\theta|\vec{d}^*) = \frac{P(\vec{d}^*|\theta)P(\theta)}{P(\vec{d}^*)}$$

where $P(\vec{d}^*|\theta)$ is the likelihood of observing the noisy data given a set of proposed parameters, $P(\theta)$ is the prior probability distribution representing our physical constraints, and $P(\vec{d}^*)$ is the evidence (marginal likelihood) which serves as a normalization constant.

5. Because MCMC algorithms only require the relative probability between two states to accept or reject a step, the constant evidence $P(\vec{d}^*)$ can be ignored. To prevent numerical underflow during computation, we evaluate the non-normalized posterior in log-space:

$$\ln P(\theta|\vec{d}^*) \propto \ln P(\vec{d}^*|\theta) + \ln P(\theta)$$

6. Assuming the experimental measurement noise on our power detectors follows a Gaussian distribution by the Central Limit Theorem, we construct the log-likelihood using a χ^2 statistic:

$$\ln P(\vec{d}^*|\theta) = -\frac{1}{2} \sum_{i=1}^k \left(\frac{\vec{d}_i^* - m_i(\theta)}{\sigma_i} \right)^2$$

where $m_i(\theta)$ is the i -th simulated detector output evaluated by the FINESSE model at proposed coordinates θ , and σ_i is the standard deviation of the noise defined for that specific channel. This step, where $m(\theta)$ is created, is the forward model, where the actual FINESSE simulation is run and is the main computational bottleneck of this process.

7. We assign a uniform log-prior that returns 0 if the proposed parameters θ lie within physically realistic boundaries (e.g., valid limits for mirror radii of curvature and laser power) and $-\infty$ if they violate these constraints, immediately rejecting the step.
8. We are left with a distribution for $P(\theta|\vec{d}^*)$ and by analyzing the width and mean of this distribution alongside the magnitude of experimental noise added into the system, we can estimate the sensitivity of the system to fluctuations in the parameters in θ .

3 Approach/Method

3.1 Code

This work is hosted in the following Git repository:

<https://git.ligo.org/abhiram.sharma/surf2026>

For these studies, I chose to adopt the base `finesse-ligo` LLO model instead of the computationally expensive thermal steady-state model, since I would need to draw hundreds to thousands of samples for the forward model. I also added power detectors throughout the model which are comparable to physical detectors in LLO. These were identified by members of Professor Richardson's group working on LLO commissioning. I also chose to limit the max TEM modes to $n = 2$ to speed up the interferometer simulation. In FINESSE syntax, the base model is as follows:

```
factory = ALIGOFactory("llo_04.yaml")
factory.params.INPUT.LASER.power = 60
base_model = factory.make()

base_model.modes(maxtem=2)

#Addition of power detectors throughout interferometer,
#totaling 22 detectors

base_model.parse("""
    fd E_refl_u45 M5.p2.o f=+f2
    fd E_refl_l45 M5.p2.o f=-f2
    ...

    ...
    pd TRX ETMX.p2.o
    pd TRY ETMY.p2.o
""")
```

Then we lock and run the simulation to generate true detector values and add noise to these values.

```
lock = InitialLockLIGO(
    exception_on_lock_fail=False,
    lock_steps=100,
    gain_scale=0.4,
    pseudo_lock_arms=False,
    run_locks=True,
)
base_model.run(lock)
out_true = base_model.run()
```

```

true_itmx_roc = base_model.ITMX.Rcx.value
true_etmx_roc = base_model.ETMX.Rcx.value

target_keys = ["E_refl_u45", "E_refl_l45" ... ]

y_obs = {}
y_err = {}

np.random.seed(42)
for key in target_keys:
    val = out_true[key]

    if isinstance(val, np.ndarray):
        val = val[0] if val.size > 0 else 0.0

    if np.iscomplexobj(val):
        val = np.abs(val)

    #Addition of noise, d -> d*
    err = np.maximum(0.025 * np.abs(val), 1e-10)
    obs = val + err * np.random.randn()

    y_obs[key] = obs
    y_err[key] = err

```

I explored three options for MCMC algorithms: PyMC, dynesty, and emcee [7, 8, 9]. After running some exploratory Python code with these packages I settled on emcee, which is an Affine Invariant MCMC Ensemble Sampler [10]. It is suitable for skewed multi-dimensional distributions which is beneficial for this study as many parameters are highly correlated. For this package, one has to define the `log_prior`, `log_probability`, and `log_likelihood` functions according to physical constraints. For the study in Section 3.2 the code looks like the following, with certain exception catches and comments removed for brevity:

```

def log_prior(theta):
    rc_itm, rc_etm= theta
    if (1928.0 < rc_itm < 1948.0) and (2230.0 < rc_etm < 2250.0):
        return 0.0
    return -np.inf

def log_likelihood(theta, obs_dict, err_dict):
    rc_itm, rc_etm= theta
    global base_model, warm_model_cache, complex_keys, y_err_inv

```

```

is_warm_start = False
nearest_model_template = None

if len(warm_model_cache) > 0:
    cached_thetas = np.array([c['theta'] for c in warm_model_cache])
    distances = np.linalg.norm(cached_thetas - theta, axis=1)
    min_idx = np.argmin(distances)
    nearest_model_template = warm_model_cache[min_idx]['model']

if nearest_model_template is not None:
    model = nearest_model_template
    active_lock = fac.RunLocks(max_iterations=40, ...
    exception_on_fail=True)
    is_warm_start = True
else:
    model = base_model
    active_lock = InitialLockLIGO(
        exception_on_lock_fail=True,
        lock_steps=100,
        gain_scale=0.4,
        pseudo_lock_arms=False,
        run_locks=True,
    )

try:
    model.ITMX.Rcx = rc_itm
    model.ITMX.Rcy = rc_itm
    model.ETMX.Rcx = rc_etm
    model.ETMX.Rcy = rc_etm

    try:
        model.run(active_lock)

    except LostLock:
        if is_warm_start:
            model = base_model

            model.ITMX.Rcx = rc_itm
            model.ITMX.Rcy = rc_itm
            model.ETMX.Rcx = rc_etm
            model.ETMX.Rcy = rc_etm

            fallback_lock = InitialLockLIGO(
                exception_on_lock_fail=True,
                lock_steps=100,
                gain_scale=0.4,

```

```

        pseudo_lock_arms=False,
        run_locks=True
    )
    model.run(fallback_lock)
else:
    return -np.inf

out = model.run()

warm_model_cache.append({'theta': theta, 'model': model})

if len(warm_model_cache) > 16:
    warm_model_cache.pop(0)

chi2 = 0.0
for key in obs_dict.keys():
    try:
        val = out[key]

        if isinstance(val, np.ndarray):
            val = val[0] if val.size > 0 else 0.0

        if key in complex_keys:
            val = np.abs(val)

        chi2 += ((obs_dict[key] - val) * y_err_inv[key])**2
    except (KeyError, ValueError, TypeError, IndexError):
        return -np.inf

if np.isnan(chi2) or np.isinf(chi2):
    return -np.inf

return -0.5 * chi2

def log_probability(theta, obs_dict, err_dict):
    lp = log_prior(theta)
    if not np.isfinite(lp):
        return -np.inf
    return lp + log_likelihood(theta, obs_dict, err_dict)

```

As we can see, the majority of the complexity is in the `log_likelihood` function which consists of a few optimizations for running the FINESSE model. For example, whenever a new θ is proposed, a cache of previous models is parsed to begin locking of the interferometer from a model which is close to the proposed parameter values, saving on locking iteration steps. This is called warm locking. However, the code is currently optimized for single core optimization since I avoided using the FINESSE `.deepcopy()` method to save time.

Mutating an existing model rather than using `.deepcopy()` is about a 40% speedup in runtime. This issue is addressed in Section 3.3.

3.2 Results/Analysis

For the first simulation, I chose two parameters of interest, the radius of curvature (RoC) of the X arm's input and end test masses (ITMX and ETMX). I expect these to be highly correlated because, for proper mode matching of the laser, if one RoC increases, the other must decrease, creating a negative correlation which can be physically validated. Based on Chapter 19 of Lasers by Anthony E. Siegman [11]:

The spatial mode of a Fabry-Perot cavity is dictated by its g -parameters, defined as:

$$g_1 = 1 - \frac{L}{R_{ITM}} \quad (1)$$

$$g_2 = 1 - \frac{L}{R_{ETM}} \quad (2)$$

The MCMC walkers should be constrained to preserve the arm cavity's spatial mode, in order to match the power detector values in $\vec{d}^*$. This forces the walkers to trace a contour where the product of the g -parameters remains a constant C :

$$g_1 g_2 = C \quad (3)$$

To mathematically determine the slope of this ridge in the MCMC corner plot, we must find how a change in the ITM RoC (R_{ITM}) dictates a compensating change in the ETM RoC (R_{ETM}). We can do this by taking the implicit derivative of the g -parameter constraint with respect to R_{ITM} :

$$\frac{d}{dR_{ITM}} \left[\left(1 - \frac{L}{R_{ITM}} \right) \left(1 - \frac{L}{R_{ETM}} \right) \right] = \frac{d}{dR_{ITM}} [C] \quad (4)$$

$$\Rightarrow \frac{L}{R_{ITM}^2} \left(1 - \frac{L}{R_{ETM}} \right) + \left(1 - \frac{L}{R_{ITM}} \right) \left(\frac{L}{R_{ETM}^2} \frac{dR_{ETM}}{dR_{ITM}} \right) = 0 \quad (5)$$

We can plug in g_1 and g_2 for the expression in parentheses:

$$\frac{L}{R_{ITM}^2} g_2 + g_1 \frac{L}{R_{ETM}^2} \frac{dR_{ETM}}{dR_{ITM}} = 0 \quad (6)$$

Isolating $\frac{dR_{ETM}}{dR_{ITM}}$ gives an analytical formula for the slope of the parameter degeneracy:

$$\frac{dR_{ETM}}{dR_{ITM}} = - \left(\frac{R_{ETM}}{R_{ITM}} \right)^2 \frac{g_2}{g_1} \quad (7)$$

Applying the LLO physical parameters ($L = 3994.5$ m, $R_{ITM} = 1937.9$ m, and $R_{ETM} = 2240.0$ m), the baseline cavity properties evaluate to $g_1 \approx -1.061$ and $g_2 \approx -0.783$. Evaluating the derivative at these coordinates yields the predicted MCMC slope:

$$\frac{dR_{ETM}}{dR_{ITM}} = - \left(\frac{2240.0}{1937.9} \right)^2 \left(\frac{-0.783}{-1.061} \right) \approx -0.986 \quad (8)$$

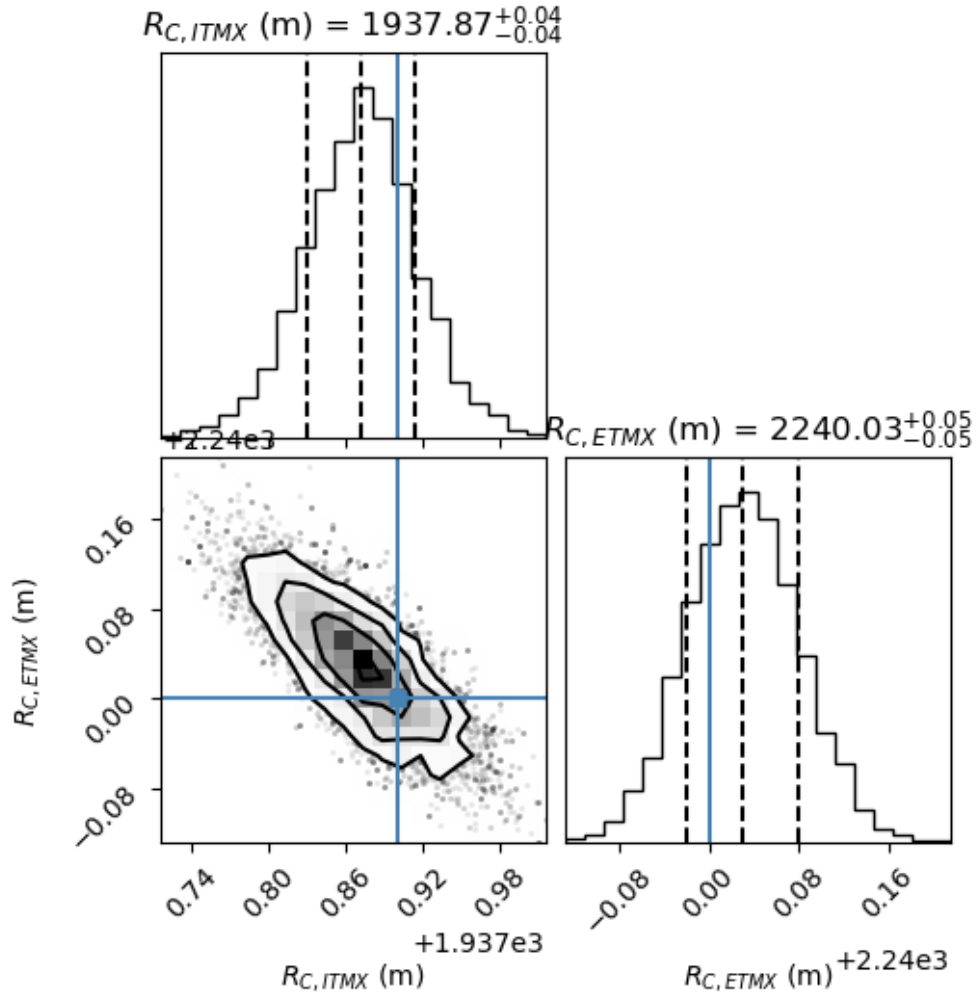


Figure 1: Corner plot of emcee sampler which ran for 500 steps and drew 13,500 samples. I considered the first 100 steps burn-in. The blue lines represent the true LLO ETM and ITM RoCs. The dashed lines represent cutoffs for the 16th, 50th, and 84th percentiles of the samples.

Fitting this data with a line yields the following slope:

```
burn_in = 100
samples = backend.get_chain(discard=burn_in, flat=True)

itm_x_samples = samples[:, 0]
etm_x_samples = samples[:, 1]

slope, intercept = np.polyfit(itm_x_samples, etm_x_samples, 1)
print(slope)
```

```
>> -0.876
```

This is an interesting side result in terms of physical validation of the algorithm. I thought this discrepancy between the measured value and predicted value could mostly be attributed to a lack of samples. However, before trying to run the sampler for longer I decided to work on speeding up the computation time as outlined in the next section.

3.3 Future Work

First, I have to parallelize the code before I start increasing the dimension of the parameter space significantly. This means using `.deepcopy()` within the forward model and evaluating how the warm model locking cache would work for multiple cores.

Additionally, after parallelization, the computation time reduction may allow me to use the full thermal steady state model. This would permit the addition of the Thermal Compensation System's ring heater power as a parameter of interest.

References

- [1] B.P. Abbott et al., *GW150914: The Advanced LIGO Detectors in the Era of First Discoveries*. Phys. Rev. Lett. 116, 131103 (2016).
- [2] Rana X. Adhikari, *Gravitational radiation detection with laser interferometry*. Reviews of Modern Physics. 86, 121 (2014).
- [3] V. B. Braginsky and S. P. Vyatchanin, *Thermodynamical fluctuations in optical mirror coatings*. Physics Letters A. 312, 244-255 (2003).
- [4] Gregory M. Harry et al., *Thermal noise from optical coatings in gravitational wave detectors*. Appl. Opt. 45, 1569-1574 (2006)
- [5] Ling Sun et al., *Characterization of systematic error in Advanced LIGO calibration* Class. Quantum Grav. 37 225008 (2020).
- [6] E. Thrane and C. Talbot, *An introduction to Bayesian inference in gravitational-wave astronomy: parameter estimation, model selection, and hierarchical models*. Publ. Astron. Soc. Aust. 36, e010 (2019).
- [7] O. Abril-Pla et al., *PyMC: a modern, and comprehensive probabilistic programming framework in Python*. PeerJ Comput. Sci. 9, e1516 (2023).
- [8] J. S. Speagle, *dynesty: a dynamic nested sampling package for estimating Bayesian posteriors and evidences*. Mon. Not. R. Astron. Soc. 493, 3132-3158 (2020).
- [9] D. Foreman-Mackey, D. W. Hogg, D. Lang, and J. Goodman, *emcee: The MCMC Hammer*. Publ. Astron. Soc. Pac. 125, 306 (2013).
- [10] J. Goodman and J. Weare, *Ensemble samplers with affine invariance*. Commun. Appl. Math. Comput. Sci. 5, 65-80 (2010).
- [11] A. E. Siegman, *Lasers*. University Science Books, Chapter 19 (1986).