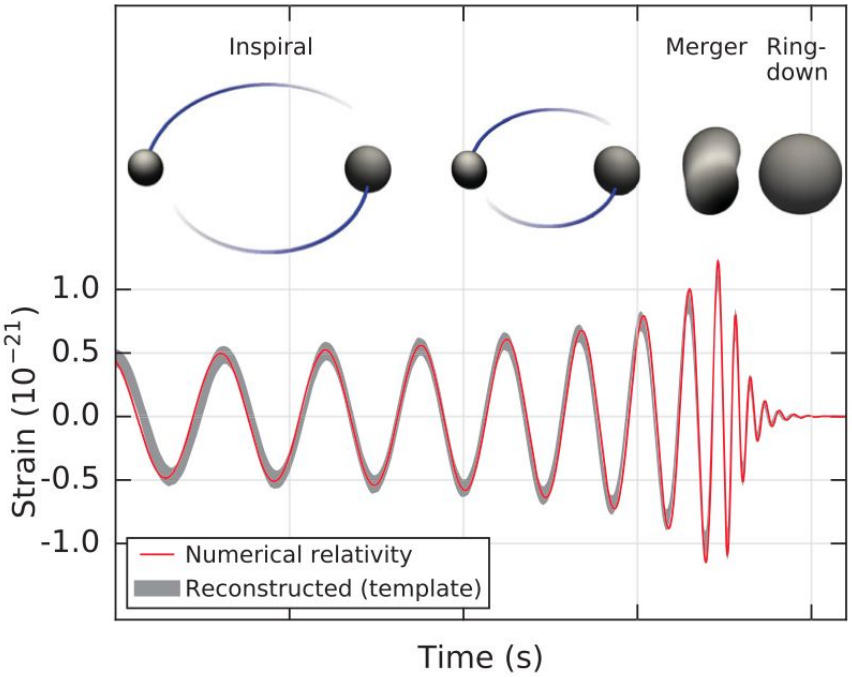
Implementation of a Differentiable Interferometer Simulator for Gravitational Wave Detector Discovery

Jonathan Klimesch

Master's Thesis in Physics

26.03.2025

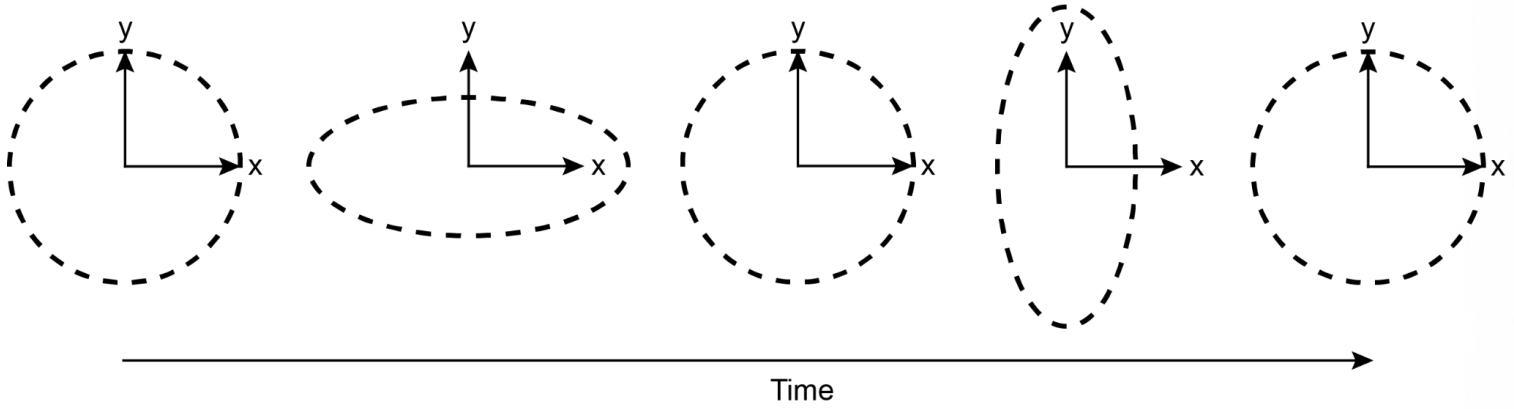
Gravitational Waves



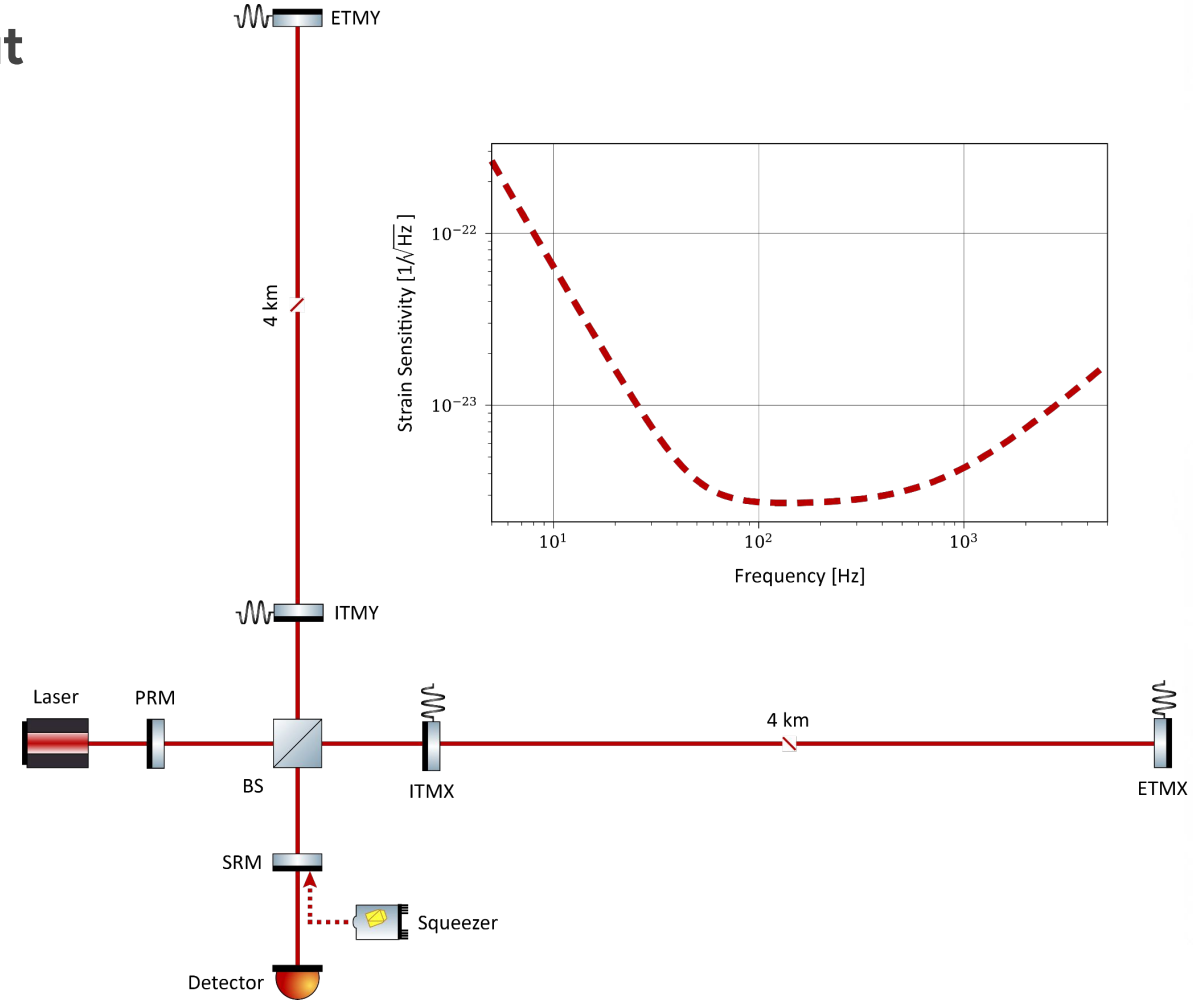
“Ripples in the curvature of spacetime that are emitted by violent astrophysical events”

- *Kip Thorne*

Gravitational Waves



LIGO Layout


ETMY
4 km
ITMY
Laser
PRM
BS
ITMX
4 km
ETMX
SRM
Squeezer
Detector
Strain Sensitivity [1/√Hz]
10^{-22}
10^{-23}
10^{1}
10^{2}
10^{3}
Frequency [Hz]

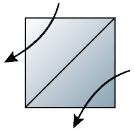
Can we use optimization techniques to find unorthodox, but useful designs?

Discrete-Continuous Search Space

Laser (Power)

Mirror (Reflectivity, Tuning)

Beamsplitter (Reflectivity)



Directional Beamsplitter

Detector

Squeezer (dB, Angle)

Space (Length)

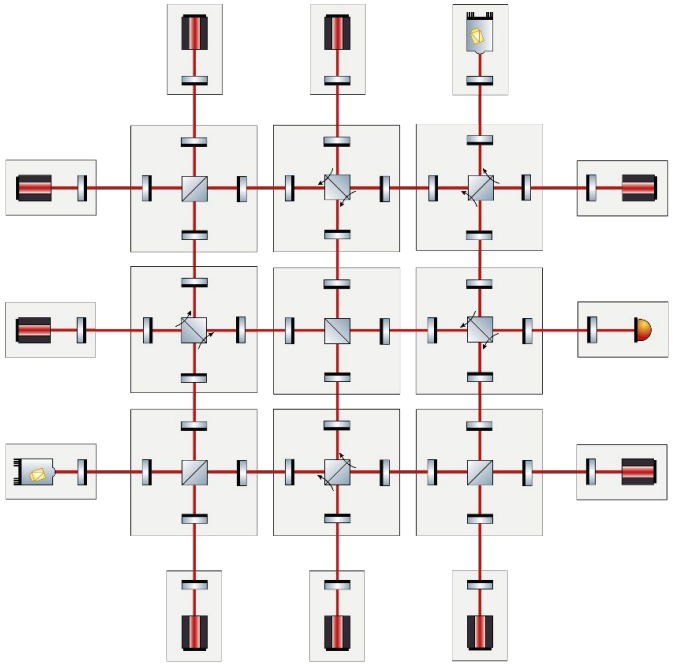
Suspension (Mass)

Where should we place which component?

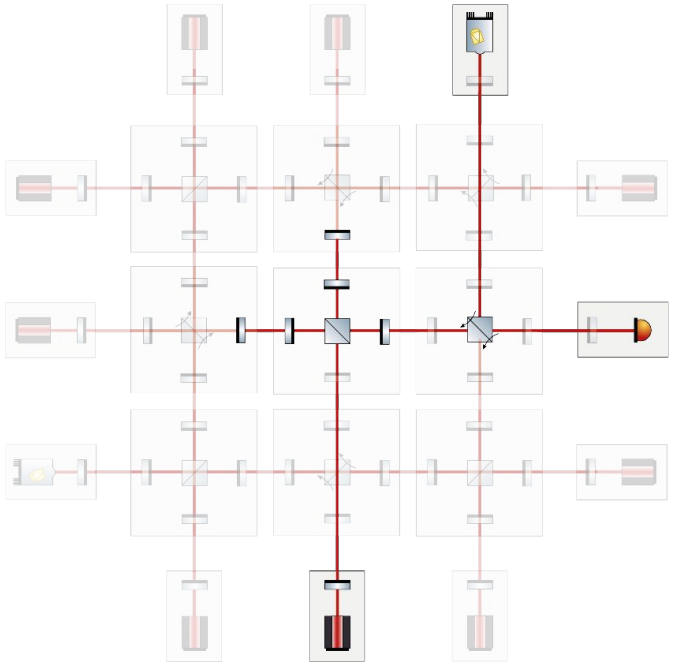
Which parameters should we choose?

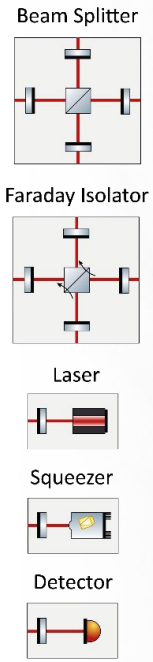
Continuous and Highly Expressive Search Space

A) Quasi-Universal Interferometer (UIFO)



B) Simplified aLIGO setup in UIFO





Krenn, M., Drori, Y., Adhikari, R. Digital discovery of interferometric gravitational wave detectors. *PRX* (2025).

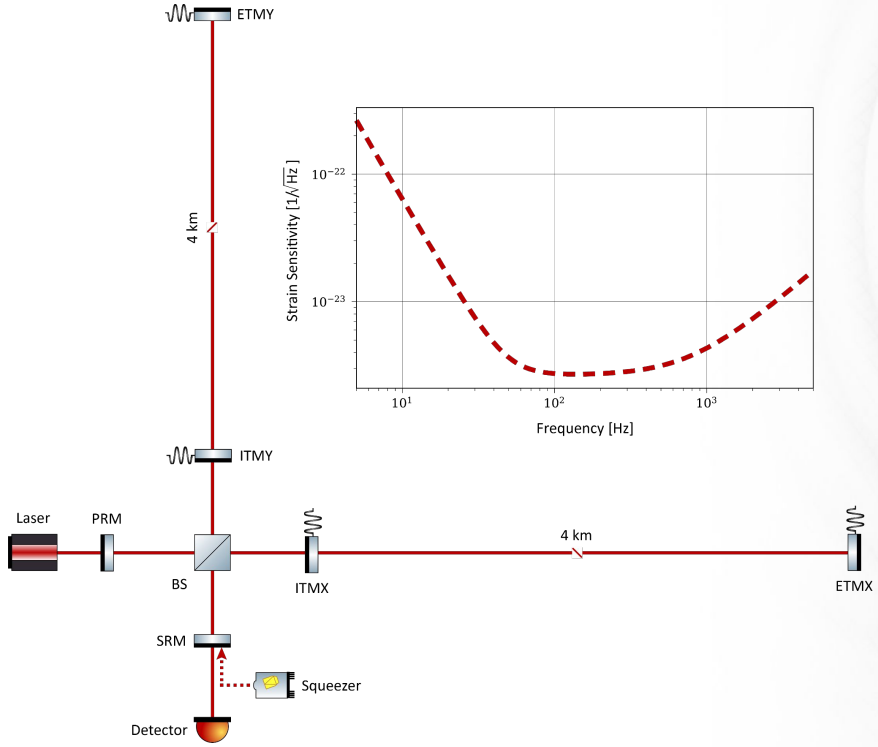
The Objective Function

$$\mathcal{L}_{\text{Strain}} = \int_{f_0}^{f_1} \log(S(f))df,$$

$$\text{Penalty}_{\text{hard}} = \sum_{RO} f\left(p(RO), co_h, c_1\right),$$

$$\text{Penalty}_{\text{soft}} = \sum_{TO} f\left(p(TO), co_s, c_2\right),$$

$$\text{Penalty}_{\text{bleach}} = \sum_{\text{Det}} f\left(p(Det), co_d, c_3\right).$$



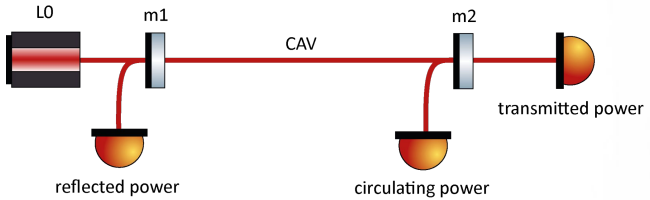
$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{Strain}} + \alpha \cdot \text{Penalty}_{\text{hard}} + \beta \cdot \text{Penalty}_{\text{soft}} + \gamma \cdot \text{Penalty}_{\text{bleach}}$$

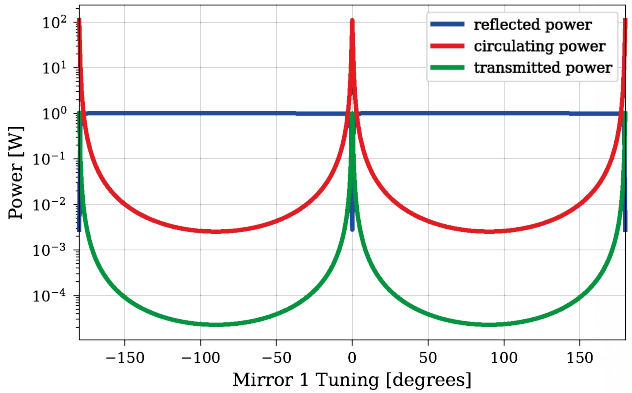
Finesse - **F**requency domain **in**terfe**r**ometer **s**imulation **s**oftwar**e**

```
    o_.-=.
   (\'".\|
  .>' (_--.
_=/d   ,^\
~~ \)-'   '
  / |
 '  '
```

Finesse 3

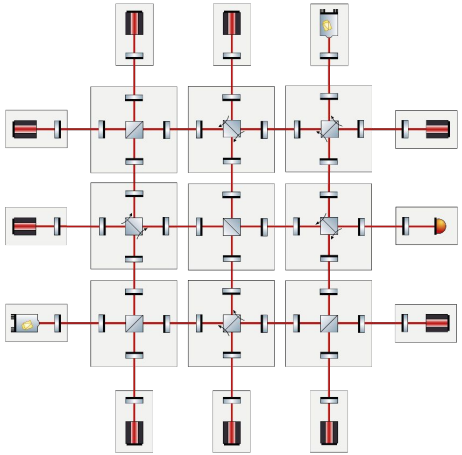
```python
import finesse
finesse.configure(plotting=True)

kat = finesse.Model()
kat.parse(
    """
    # Add a Laser named L0 with a power of 1 W.
    l L0 P=1

    # Space attaching L0 <-> m1 with length of 0 m (default).
    s s0 L0.p1 m1.p1

    # Highly reflective input mirror of cavity
    m m1 R=0.99 T=0.01

    # Intra-cavity space with length of 1 m.
    s CAV m1.p2 m2.p1 L=1

    # Highly reflective end mirror of cavity.
    m m2 R=0.991 T=0.009

    # Power detectors on reflection, circulation and transmission.
    pd reflected_power m1.p1.o
    pd circulating_power m2.p1.i
    pd transmitted_power m2.p2.o
    """
)

out = kat.run("xaxis(m1.phi, lin, -180, 180, 400)")
out.plot(logy=True)
```





Ingredients

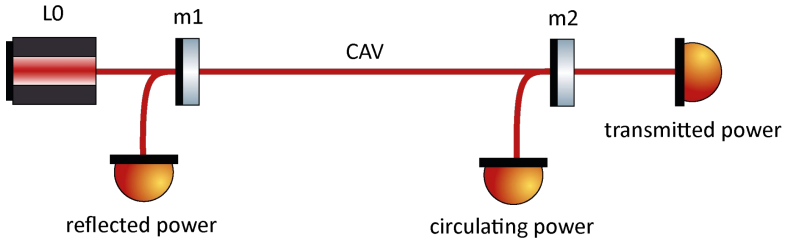
1. Search Space



2. Objective Function

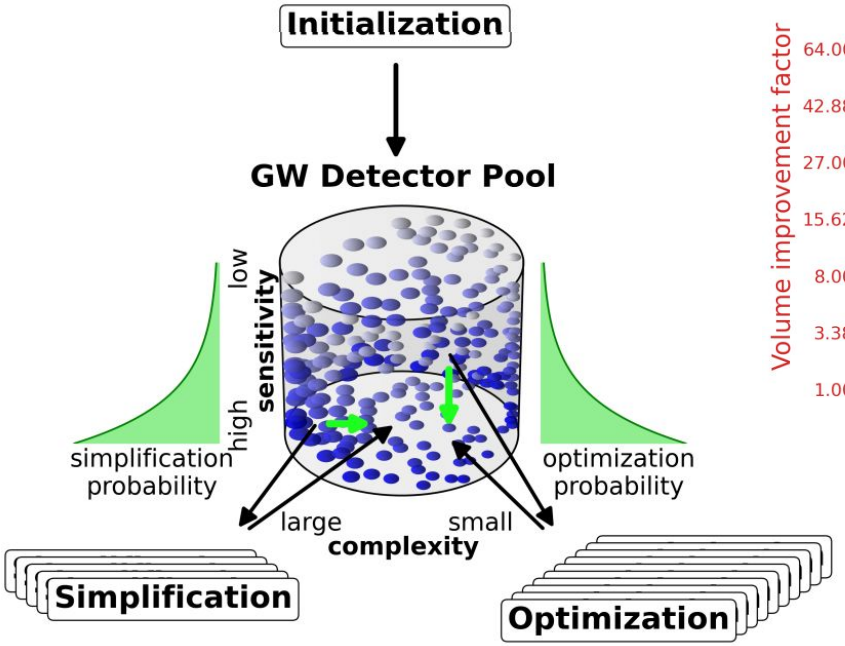
$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{Strain}} + \alpha \cdot \text{Penalties}$$

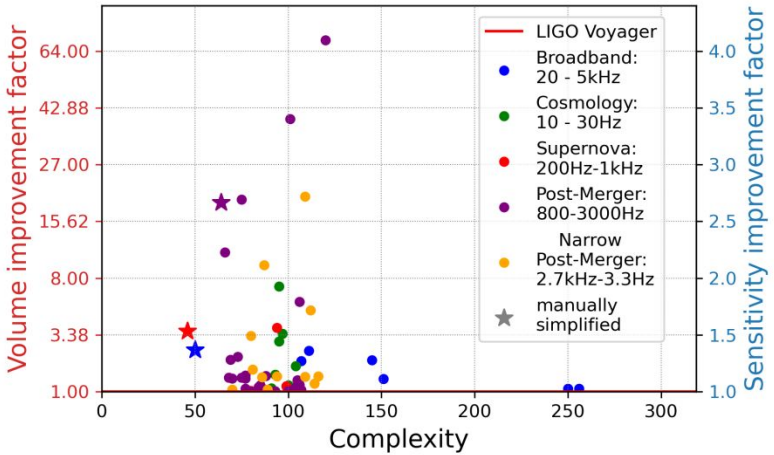
3. Simulator



How do we optimize this?

Urania - Local-Global Optimizer

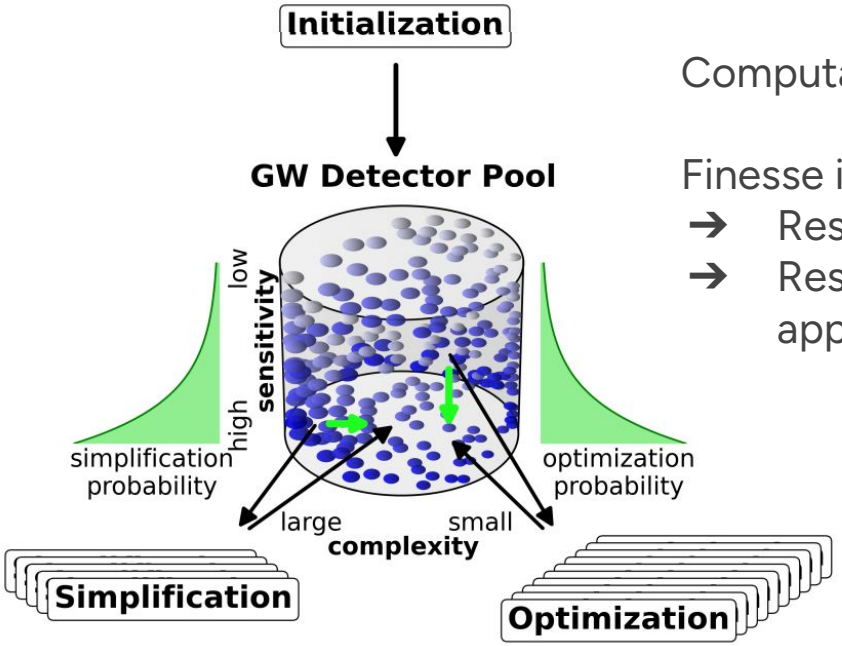




Optimization using BFGS

Krenn, M., Drori, Y., Adhikari, R. Digital discovery of interferometric gravitational wave detectors. *PRX* (2025).

The Optimization Bottleneck



Computational cost: **1.5 million CPU-hours**

Finesse is implemented in Numpy / Cython

➔ Restriction to CPU

➔ Restriction to numerical gradient approximation

Optimization using BFGS

Krenn, M., Drori, Y., Adhikari, R. Digital discovery of interferometric gravitational wave detectors. *PRX* (2025).

Can we implement a simulator designed for detector optimizations?

- Library for array-oriented numerical computation (à la Numpy)
- Automatic Differentiation
- Just-in-time compilation

diffferometor - A Differentiable Interferometer Simulator

Plane Wave Propagation

Signal Sidebands

Quantum Noise

Optomechanics

cloc - Lines of Code Comparison

Finesse 3

dⅈfferometor

	Finesse 3	dⅈfferometor
Files	269	8
Blank	14794	450
Comment	21736	646
Code	49152	2123

What differometor does not have

Hermite-Gaussian Beams

Surface Motion Simulation

Complex Suspension Systems

Thermal Effects

differometor - Subsystems

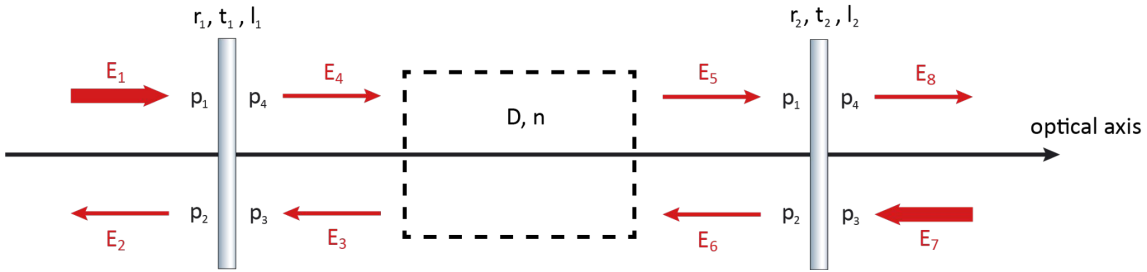
Plane Wave Propagation

Signal Sidebands

Quantum Noise

Optomechanics

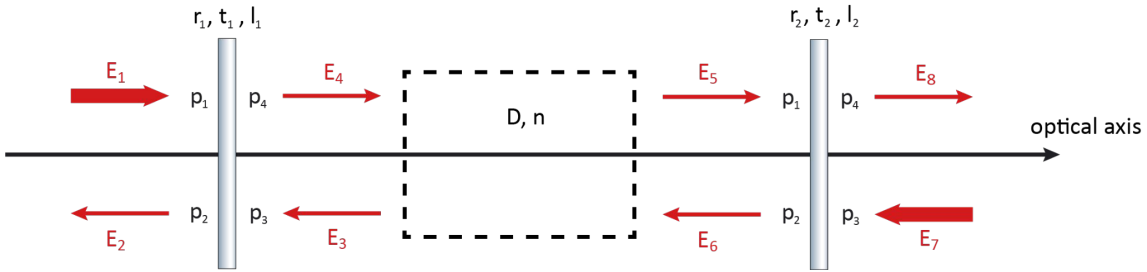
The Carrier System



$$\begin{pmatrix} E_2 \\ E_4 \end{pmatrix} = \begin{pmatrix} \mathrm{i}t & r \\ r & \mathrm{i}t \end{pmatrix} \begin{pmatrix} E_3 \\ E_1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ -r & 1 & -it & 0 \\ 0 & 0 & 1 & 0 \\ -it & 0 & -r & 1 \end{pmatrix} \begin{pmatrix} E_1 \\ E_2 \\ E_3 \\ E_4 \end{pmatrix} = \begin{pmatrix} E_{1i} \\ 0 \\ E_{3i} \\ 0 \end{pmatrix}$$

The Carrier System

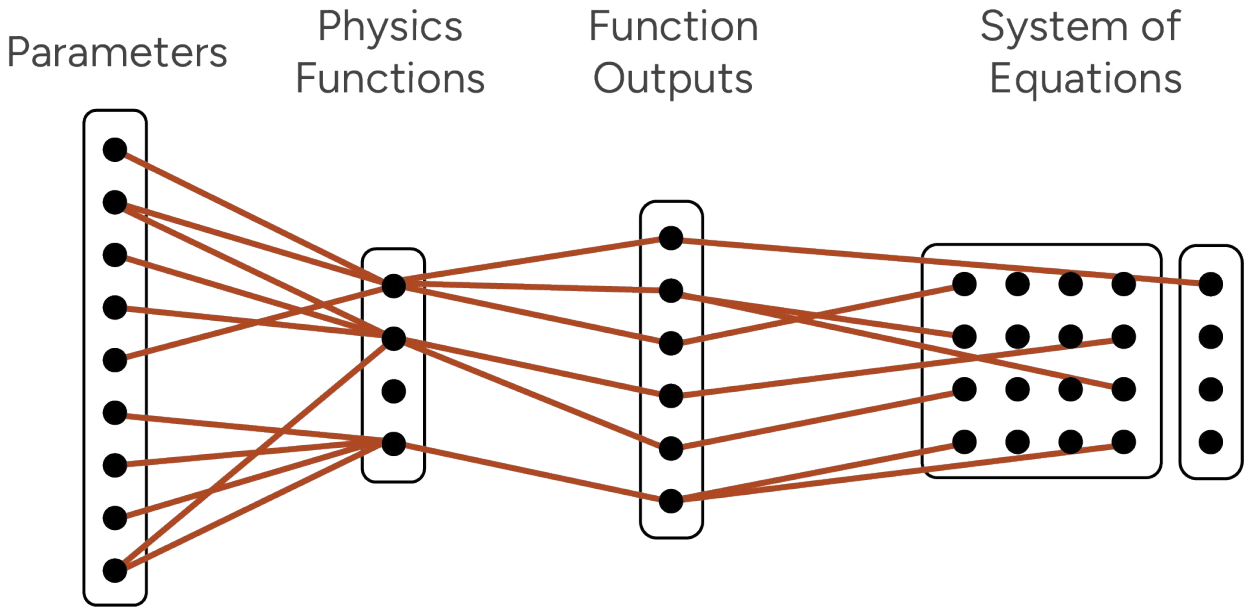


$$\begin{pmatrix} E_2 \\ E_4 \end{pmatrix} = \begin{pmatrix} \mathrm{i}t & r \\ r & \mathrm{i}t \end{pmatrix} \begin{pmatrix} E_3 \\ E_1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ -r & 1 & -it & 0 \\ 0 & 0 & 1 & 0 \\ -it & 0 & -r & 1 \end{pmatrix} \begin{pmatrix} E_1 \\ E_2 \\ E_3 \\ E_4 \end{pmatrix} = \begin{pmatrix} E_{1i} \\ 0 \\ E_{3i} \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -r_1 & 1 & -it_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -e^{-iknD} & 0 & 0 \\ -it_1 & 0 & -r_1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -e^{-iknD} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -r_2 & 1 & -it_2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & -it_2 & 0 & -r_2 & 1 \end{pmatrix} \begin{pmatrix} E_1 \\ E_2 \\ E_3 \\ E_4 \\ E_5 \\ E_6 \\ E_7 \\ E_8 \end{pmatrix} = \begin{pmatrix} E_{1i} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ E_{7i} \\ 0 \end{pmatrix}$$

The Carrier System



$$f_{\text{laser}}(P, \varphi) = \sqrt{\frac{2P}{\epsilon_0 c}} \cdot \exp(i\varphi)$$

$$f_{\text{space}}(\Delta f, L, n) = -\exp(-\mathrm{i}2\pi \frac{\Delta f}{c} nL)$$

differometor - Subsystems

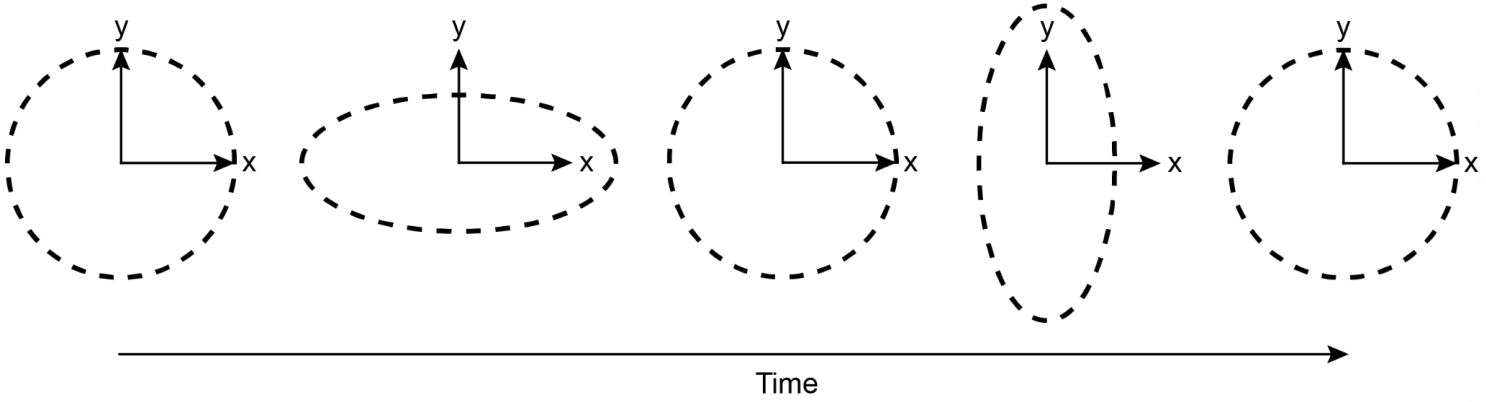
Plane Wave Propagation

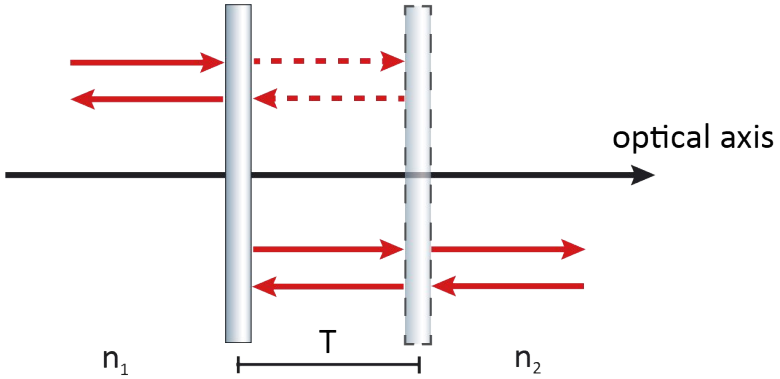
Signal Sidebands

Quantum Noise

Optomechanics

The Signal System



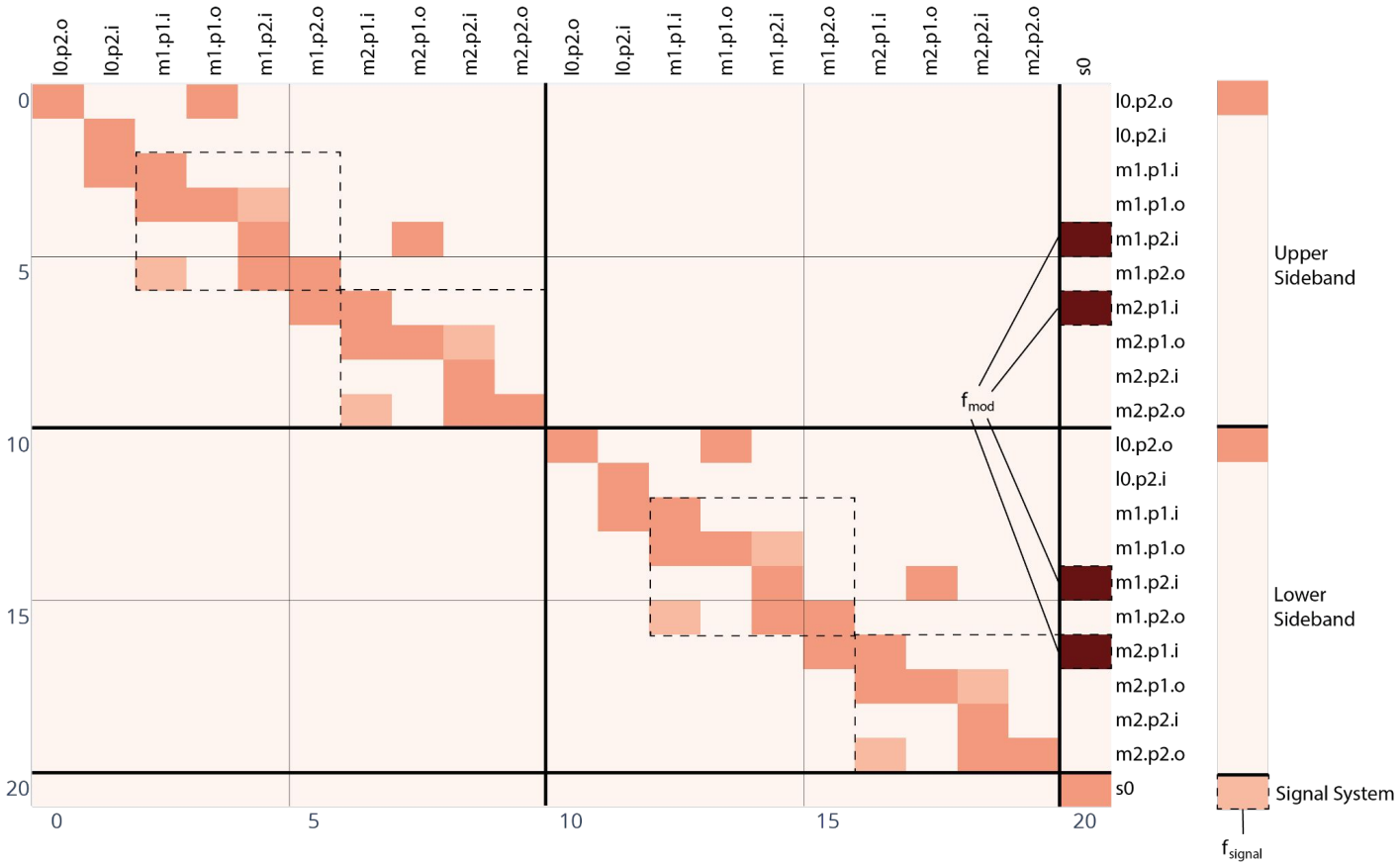


$$E = E_0 \exp(\mathrm{i}(\omega_0 t + \varphi_0 + \phi(t)))$$

$$\phi(t) = m \cos(\Omega t + \varphi_s)$$

$$E = E_0 \left(1 - \frac{m^2}{4}\right) \exp(i(w_0 t + \varphi_0)) + E_0 \frac{m}{2} \exp\left(i\left((w_0 - \Omega) t + \varphi_0 + \frac{\pi}{2} - \varphi_s\right)\right) + E_0 \frac{m}{2} \exp\left(i\left((w_0 + \Omega) t + \varphi_0 + \frac{\pi}{2} + \varphi_s\right)\right)$$

The Signal System



differometor - Subsystems

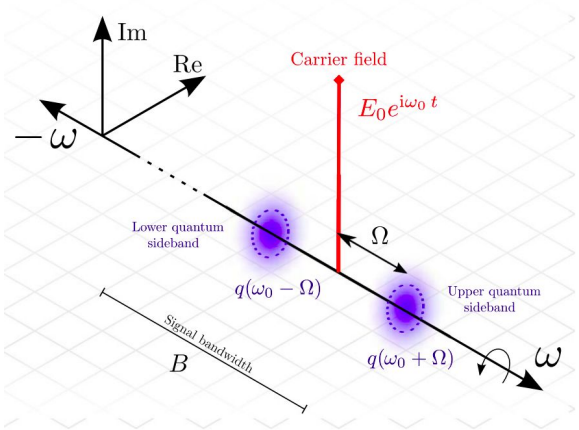
Plane Wave Propagation

Signal Sidebands

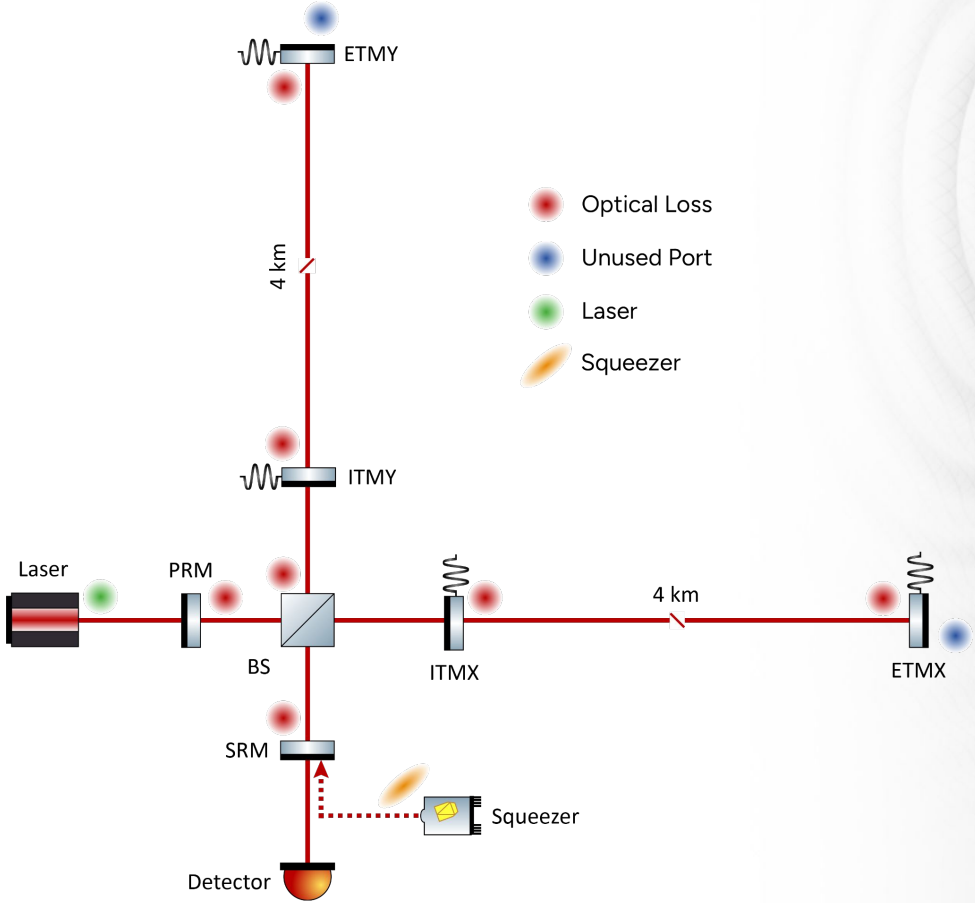
Quantum Noise

Optomechanics

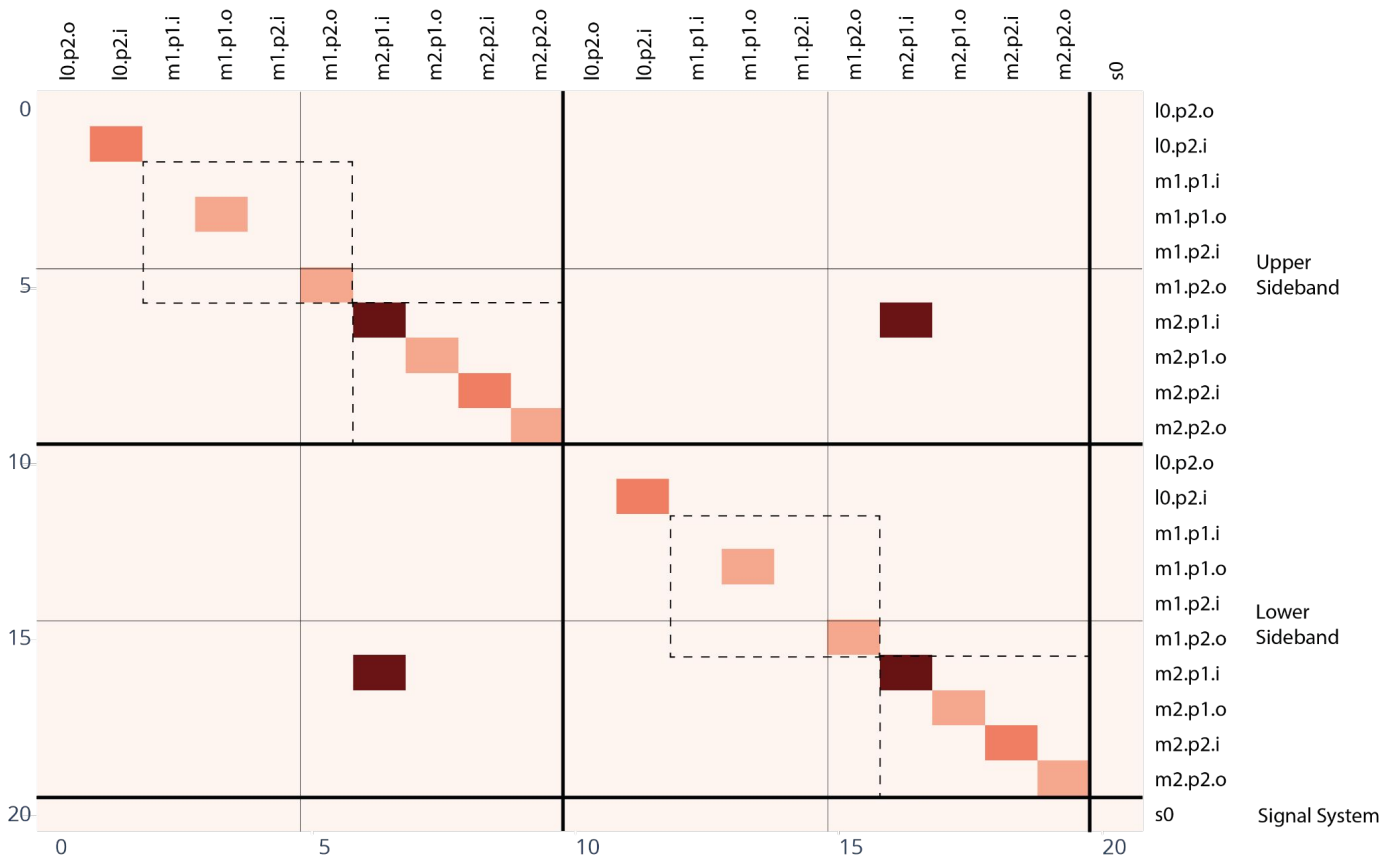
Quantum Noise



$$\mathbb{V}_o = \mathbb{M}^{-1}\mathbb{V}_i\mathbb{M}^{-1\dagger}$$



Quantum Noise



differometor - Subsystems

Plane Wave Propagation

Signal Sidebands

Quantum Noise

Optomechanics

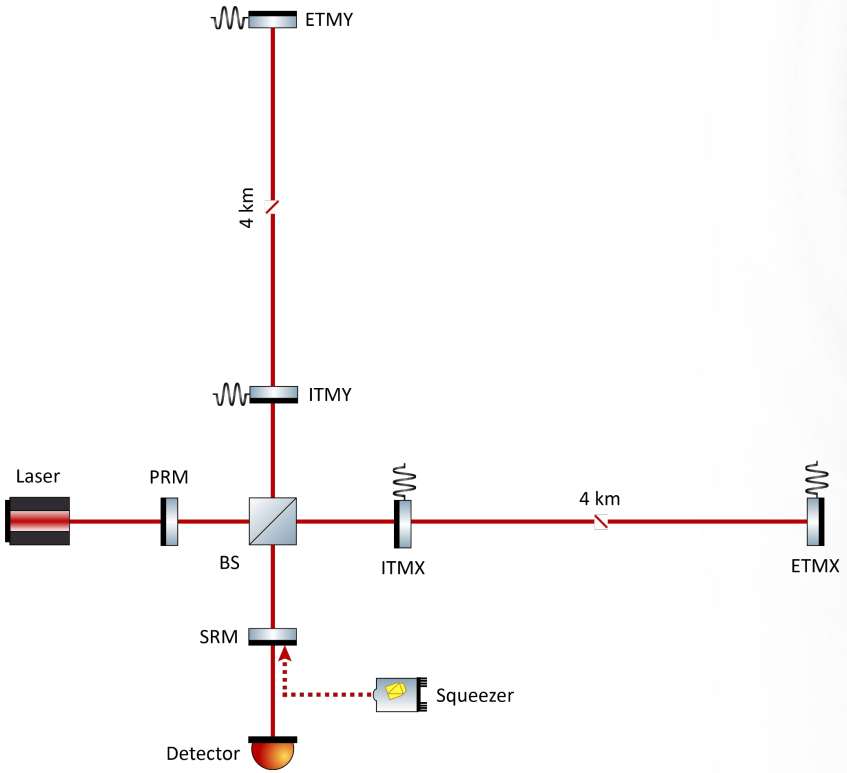
Optomechanics

$$F(\omega) = \frac{P(\omega)\cos\alpha}{c}$$

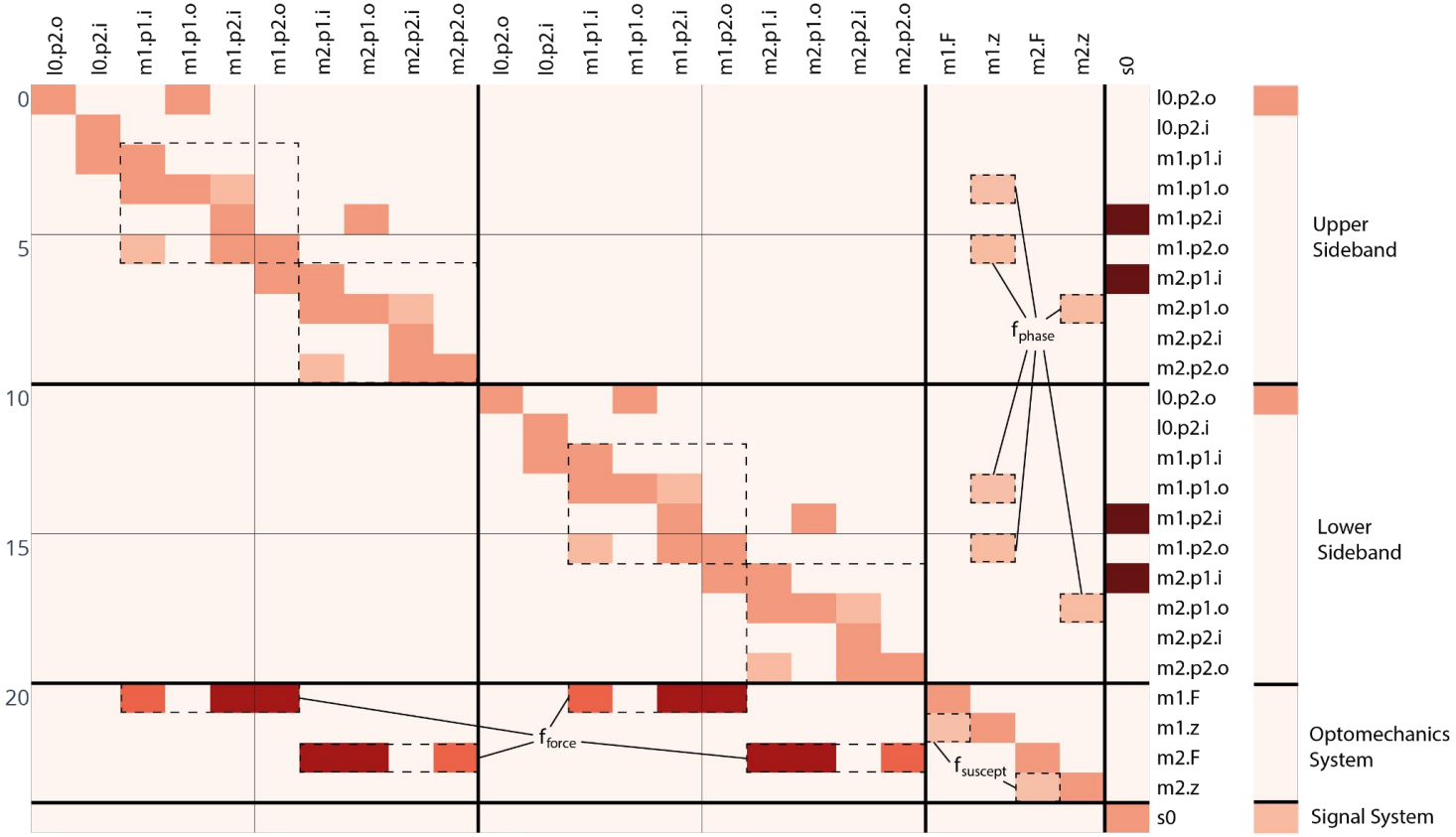
$$H(\omega) = -\frac{1}{m\omega^2}$$

$$\delta z(\omega) = H(\omega)\sum_{n=0}^{N_f} F_n(\omega)$$

$$\varphi = -k\delta z(\omega)$$



Optomechanics



A new simulator should ...

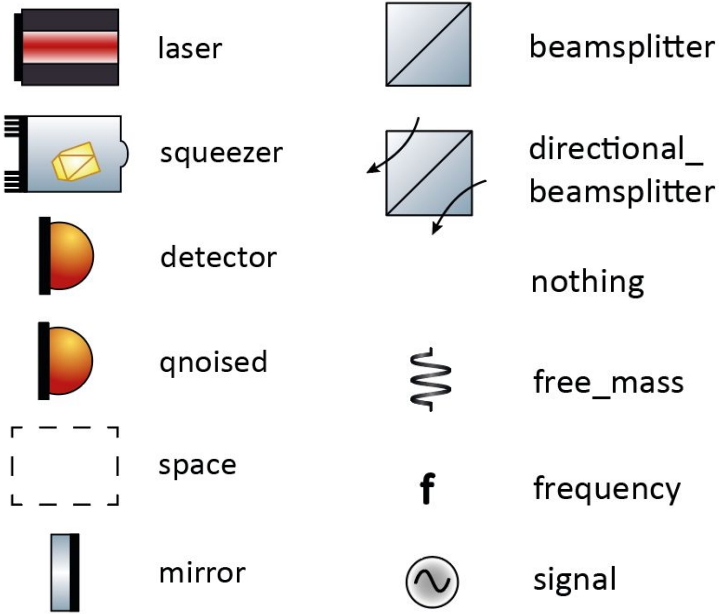
- be similar to Finesse
- implement basic components for detector optimization
- produce the same results as Finesse
- speed up simulations through GPU support and JIT compilation
- speed up optimizations through auto-differentiation
- support power constrained optimizations
- support large discovery optimizations

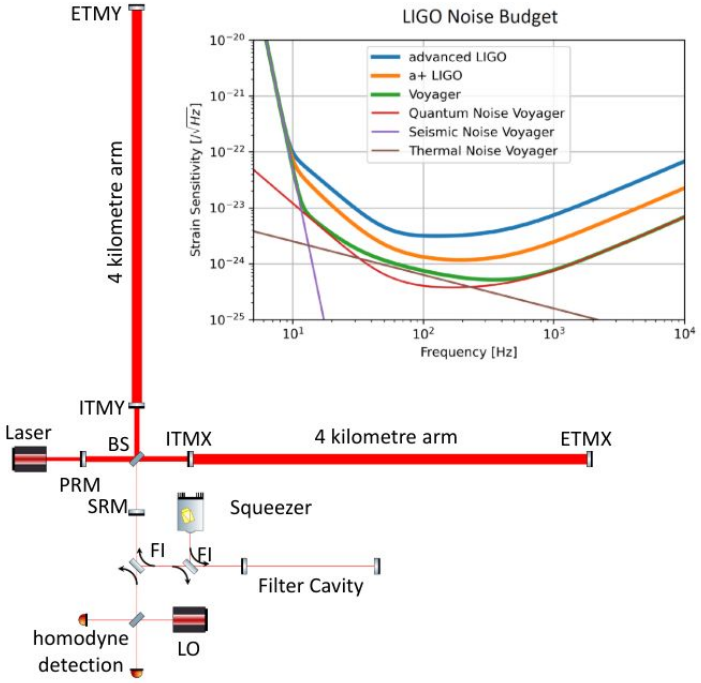
... be similar to Finesse

```
import finesse
finesse.configure(plotting=True)

kat = finesse.Model()
kat.parse(
    """
    # Add a Laser named L0 with a power of 1 W.
    l L0 P=1

    # Space attaching L0 <-> m1 with length of 0 m (default).
    s s0 L0.p1 m1.p1

    # Highly reflective input mirror of cavity
    m m1 R=0.99 T=0.01

    # Intra-cavity space with length of 1 m.
    s CAV m1.p2 m2.p1 L=1

    # Highly reflective end mirror of cavity.
    m m2 R=0.991 T=0.009

    # Power detectors on reflection, circulation and transmission.
    pd reflected_power m1.p1.o
    pd circulating_power m2.p1.i
    pd transmitted_power m2.p2.o
    """
)

out = kat.run("xaxis(m1.phi, lin, -180, 180, 400)")
out.plot(logy=True)
```

```
import networkx as nx

G = nx.DiGraph()
G.add_node("l0", component="laser")
G.add_node("m1", component="mirror", properties={"reflectivity": 0.99})
G.add_node("m2", component="mirror", properties={"reflectivity": 0.991})

G.add_edge("l0", "m1")
G.add_edge("m1", "m2", properties={"length": 1})

G.add_node("reflected_power", component="detector", target="m1",
↪  direction="out")
G.add_node("circulating_power", component="detector", target="m2")
G.add_node("transmitted_power", component="detector", target="m2",
↪  port="right", direction="out")
```

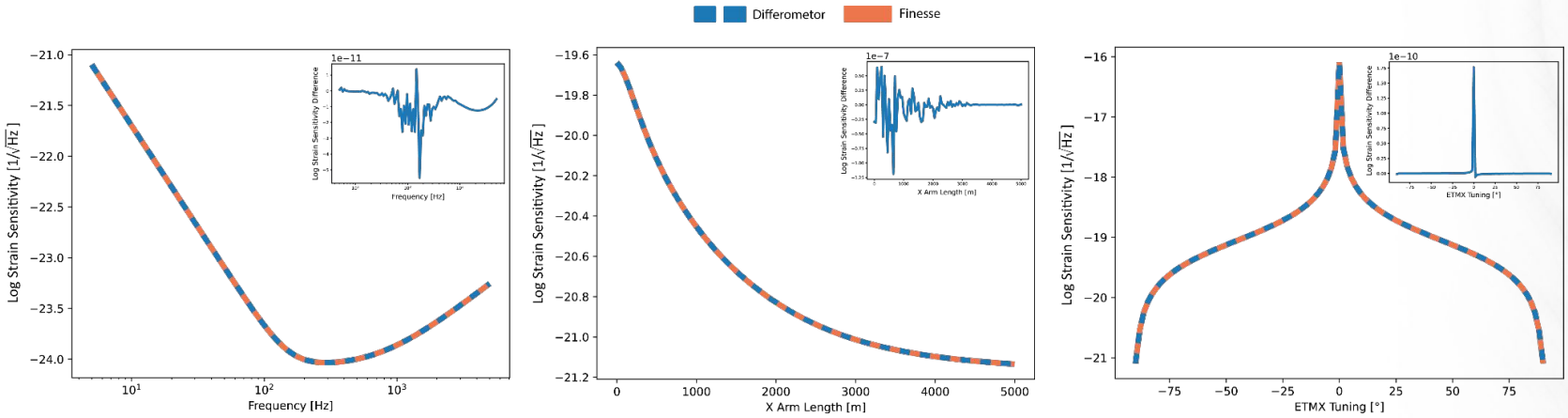
… implement basic components for detector optimization



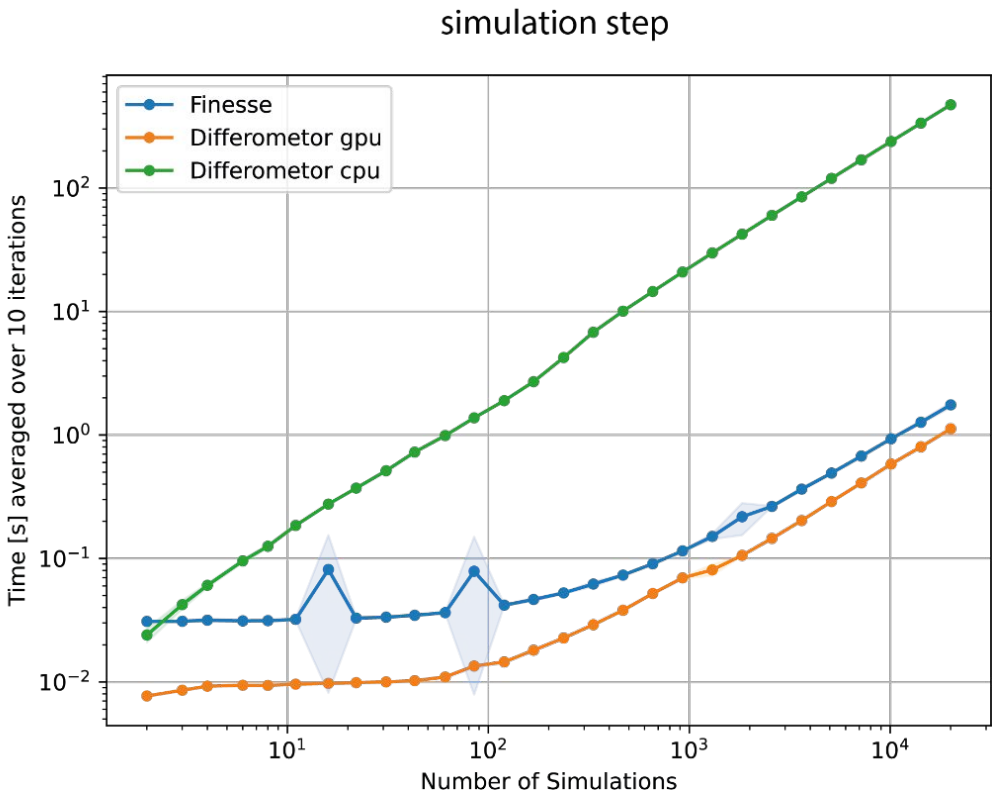


… produce the same results as Finesse

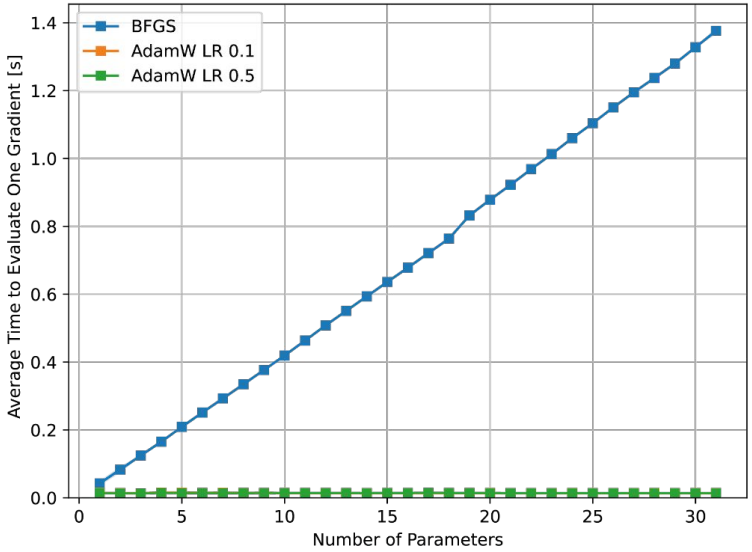
> > 99% agreement using normalized root-mean-square deviation

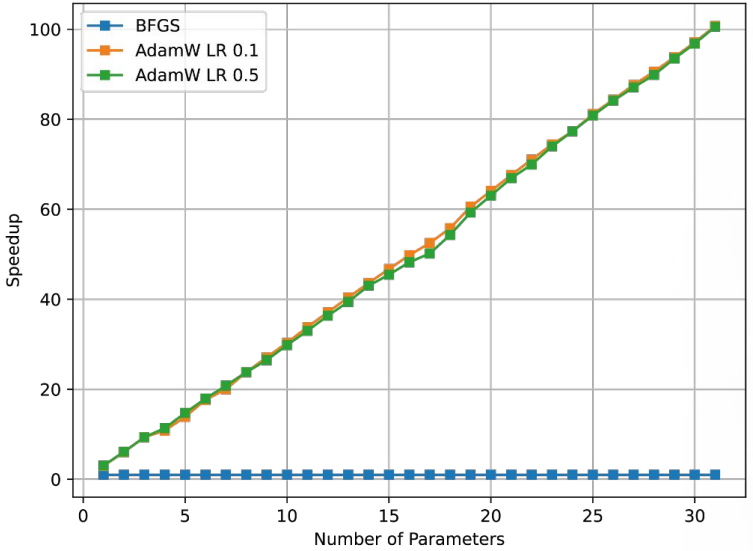


… speed up simulations through GPU support and JIT compilation



… speed up optimizations through auto-differentiation





… support power constrained optimizations

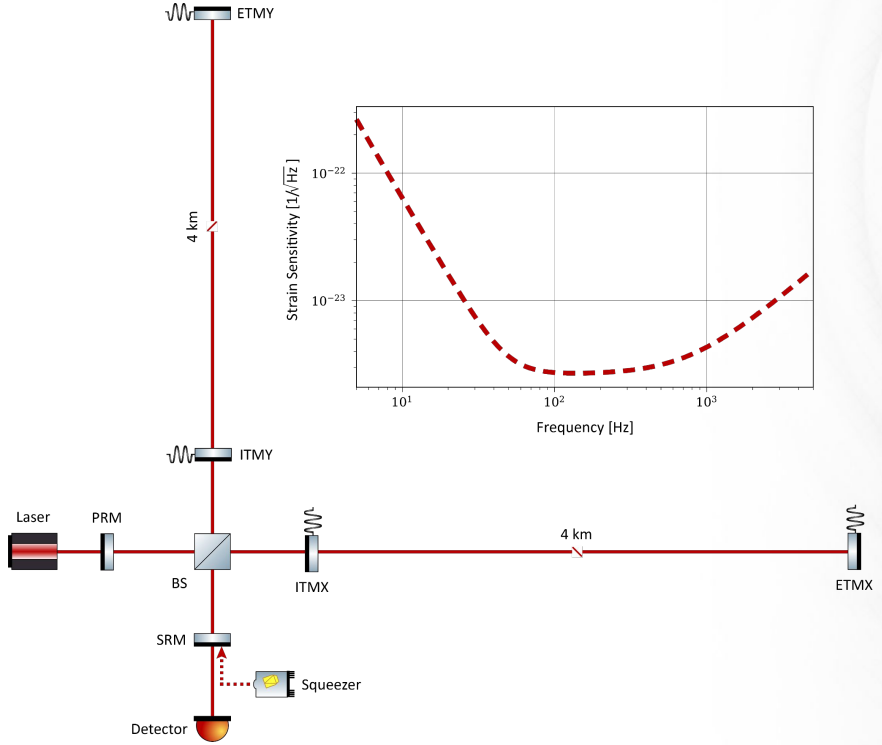
$$\mathcal{L}_{\text{Strain}} = \int_{f_0}^{f_1} \log(S(f))df,$$

$$\text{Penalty}_{\text{hard}} = \sum_{RO} f\left(p(RO), co_h, c_1\right),$$

$$\text{Penalty}_{\text{soft}} = \sum_{TO} f\left(p(TO), co_s, c_2\right),$$

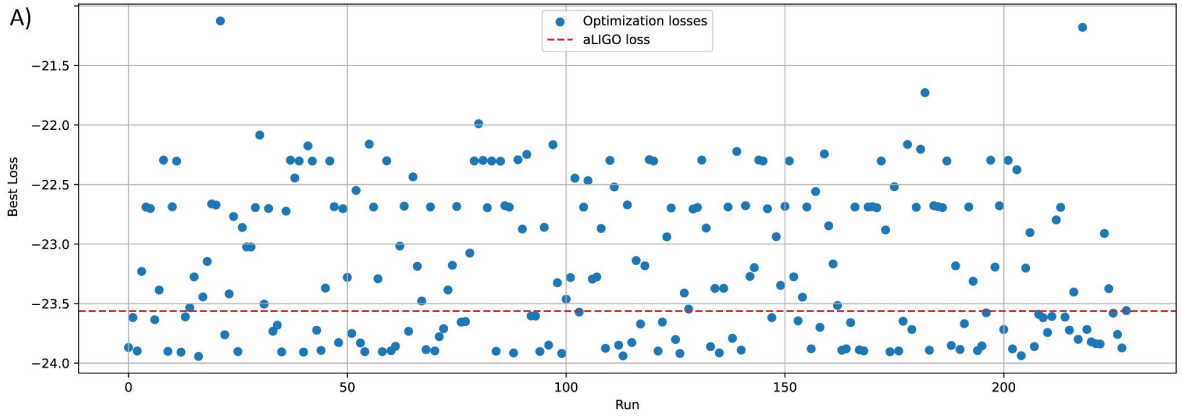
$$\text{Penalty}_{\text{bleach}} = \sum_{\text{Det}} f\left(p(Det), co_d, c_3\right).$$

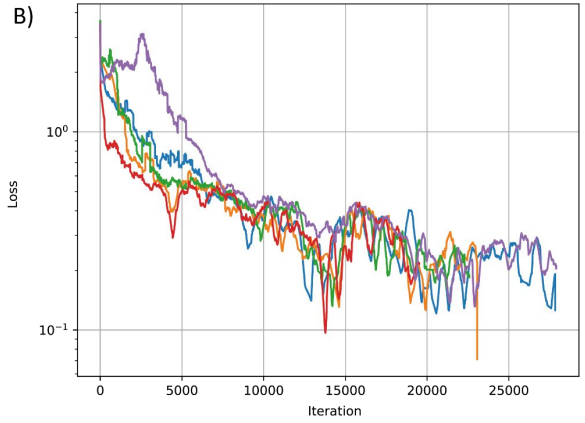
$$f(p, co, c) = \begin{cases} 0 & \text{if } p \leq co \\ (p - co) + c & \text{if } p > co \end{cases}$$

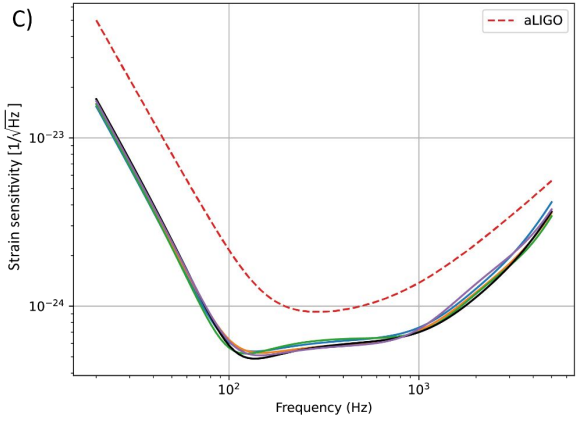


$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{Strain}} + \alpha \cdot \text{Penalty}_{\text{hard}} + \beta \cdot \text{Penalty}_{\text{soft}} + \gamma \cdot \text{Penalty}_{\text{bleach}}$$

… support power constrained optimizations







… support large discovery optimizations

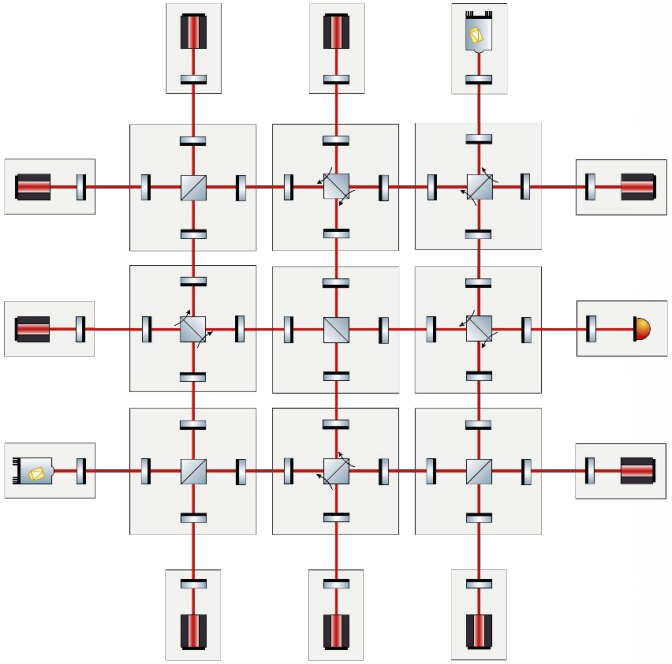
$$\mathcal{L}_{\text{Strain}} = \int_{f_0}^{f_1} \log(S(f))df,$$

$$\text{Penalty}_{\text{hard}} = \sum_{RO} f\left(p(RO), co_h, c_1\right),$$

$$\text{Penalty}_{\text{soft}} = \sum_{TO} f\left(p(TO), co_s, c_2\right),$$

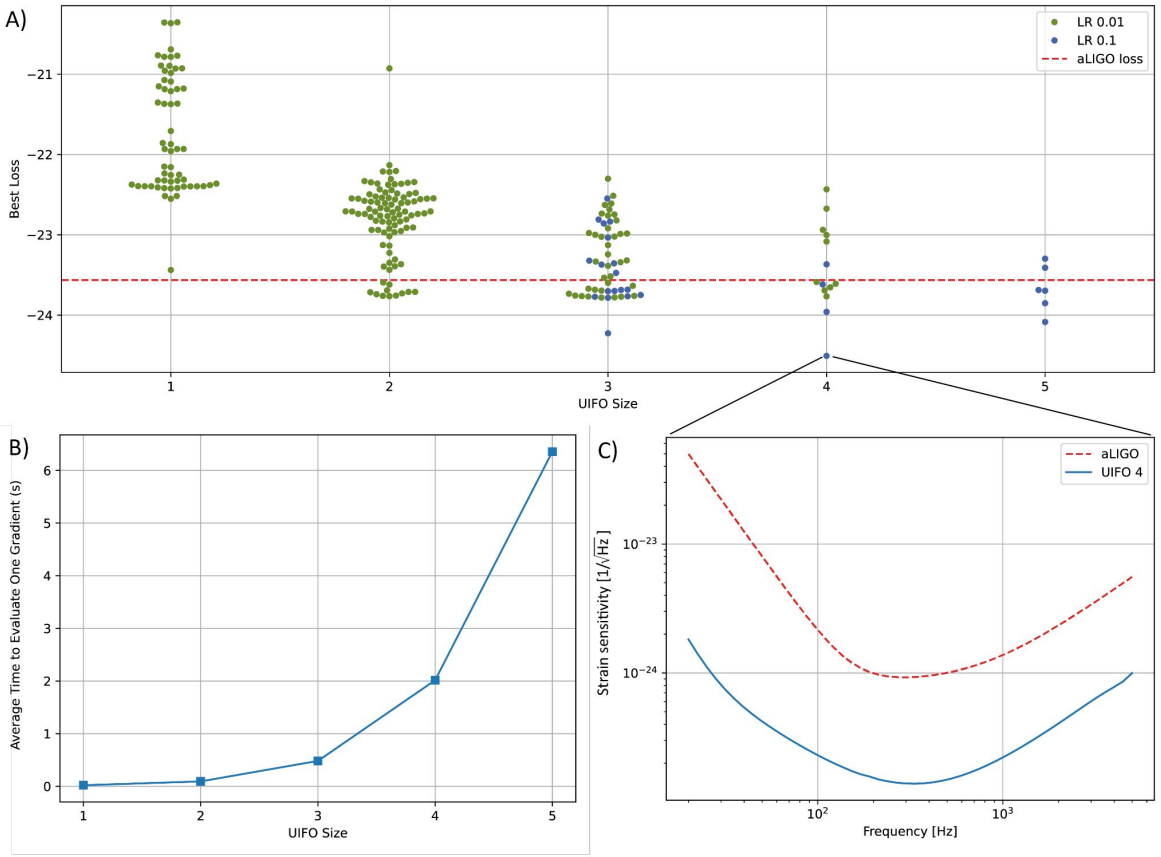
$$\text{Penalty}_{\text{bleach}} = \sum_{\text{Det}} f\left(p(Det), co_d, c_3\right).$$

$$f(p, co, c) = \begin{cases} 0 & \text{if } p \leq co \\ (p - co) + c & \text{if } p > co \end{cases}$$



$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{Strain}} + \alpha \cdot \text{Penalty}_{\text{hard}} + \beta \cdot \text{Penalty}_{\text{soft}} + \gamma \cdot \text{Penalty}_{\text{bleach}}$$

… support large discovery optimizations



Size	Parameters
1	48 - 51
2	148 - 160
3	296 - 323
4	492 - 537
5	742 - 805

dfferometor …

- is similar to Finesse
- implements basic components for detector optimization
- produces the same results as Finesse
- speeds up simulations through GPU support and JIT compilation
- speeds up optimizations through auto-differentiation
- supports power constrained optimizations
- supports large discovery optimizations

What’s next?

- Large-scale discovery optimizations
- Sparsification of interferometer matrices
- Implementation of more components

dfferometor …

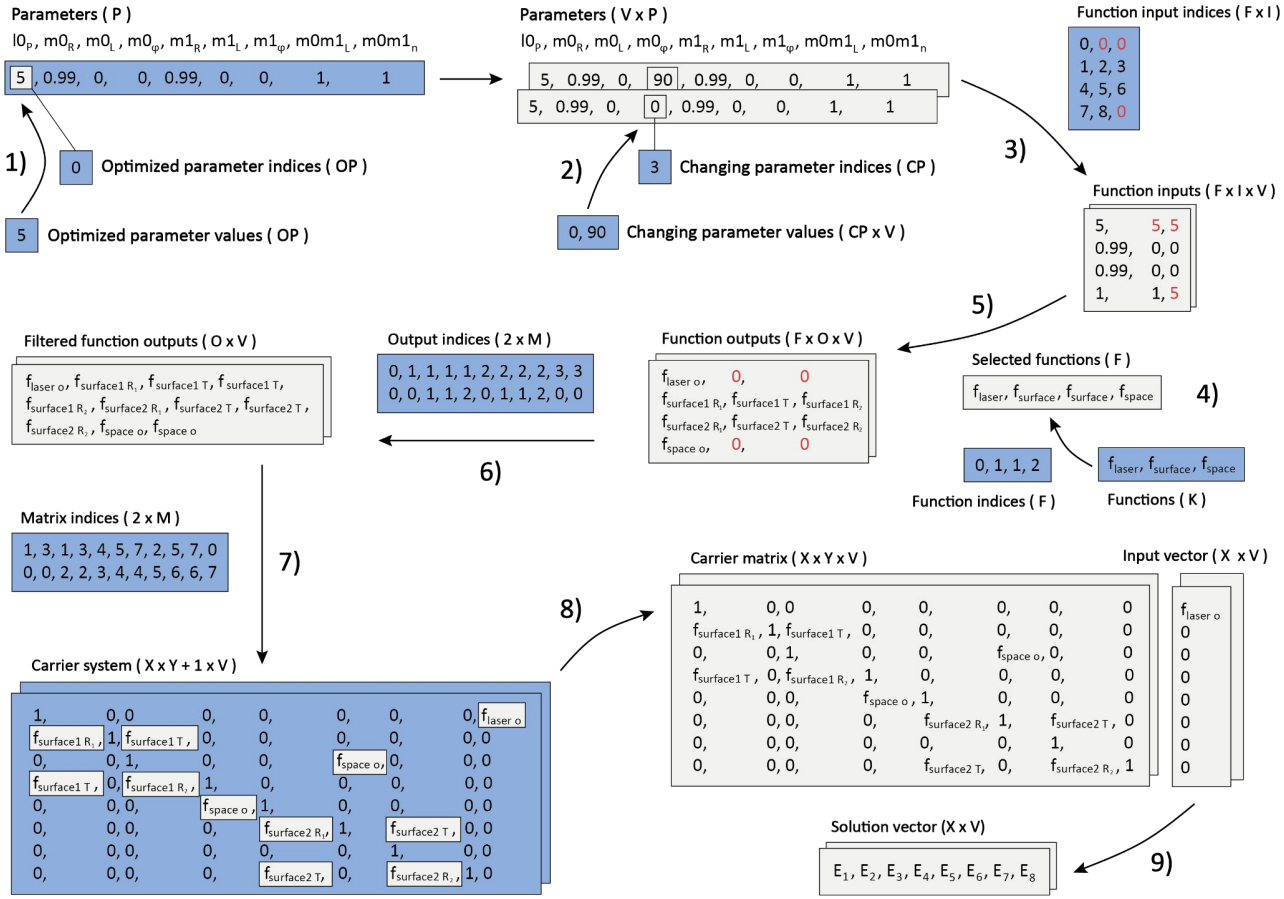
- is similar to Finesse
- implements basic components for detector optimization
- produces the same results as Finesse
- speeds up simulations through GPU support and JIT compilation
- speeds up optimizations through auto-differentiation
- supports power constrained optimizations
- supports large discovery optimizations

What's next?

- Large-scale discovery optimizations
- Sparsification of interferometer matrices
- Implementation of more components

Thank you for listening!
Any questions?

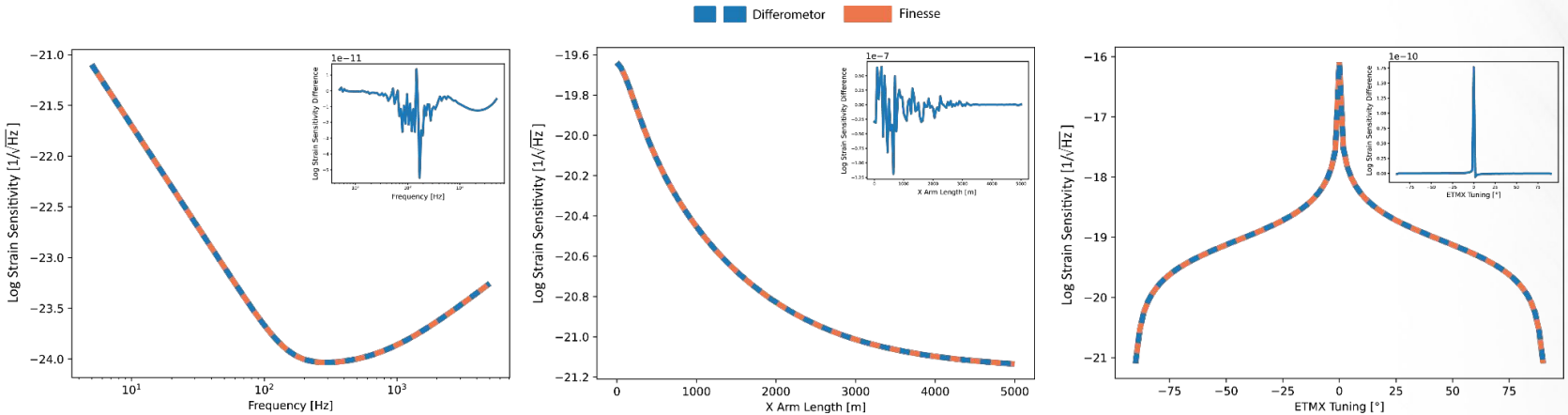
Carrier System



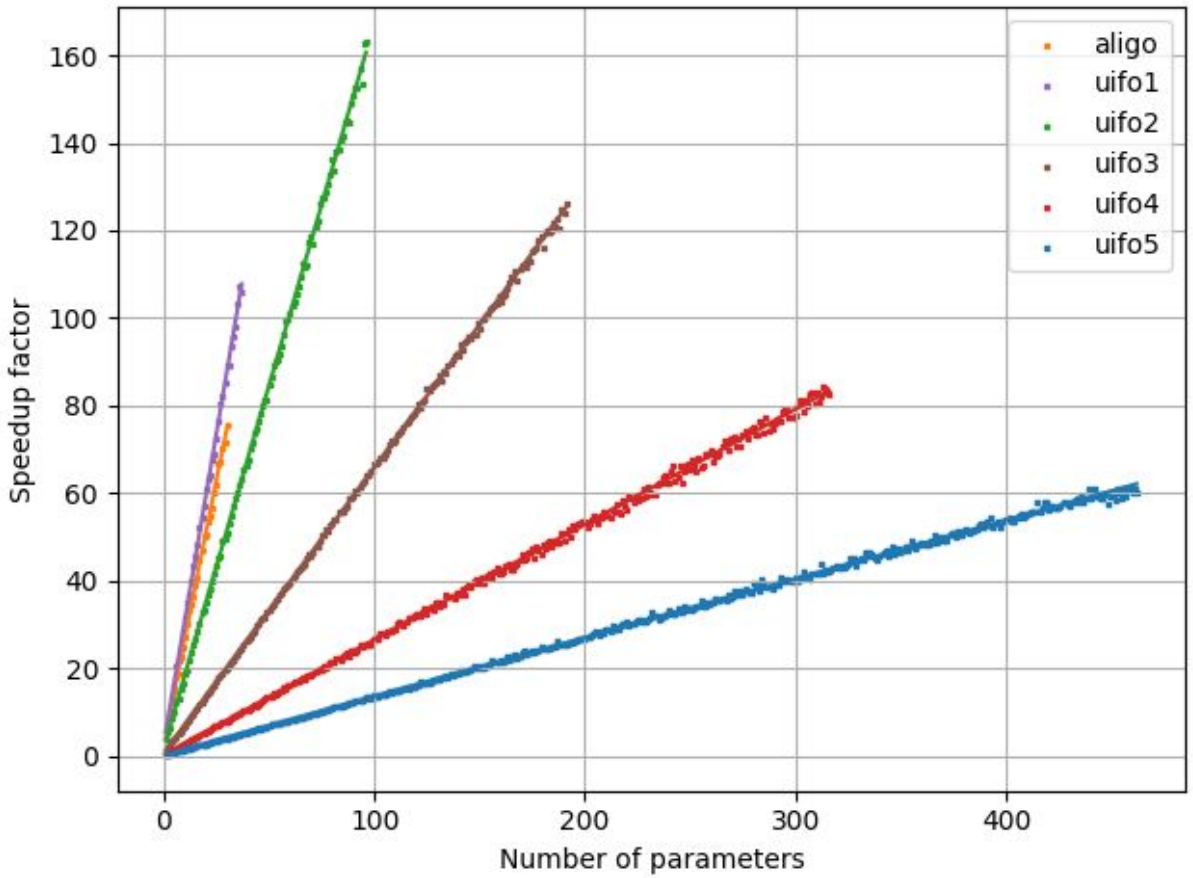
… produce the same results as Finesse

<< 1% agreement using normalized root-mean-square deviation:

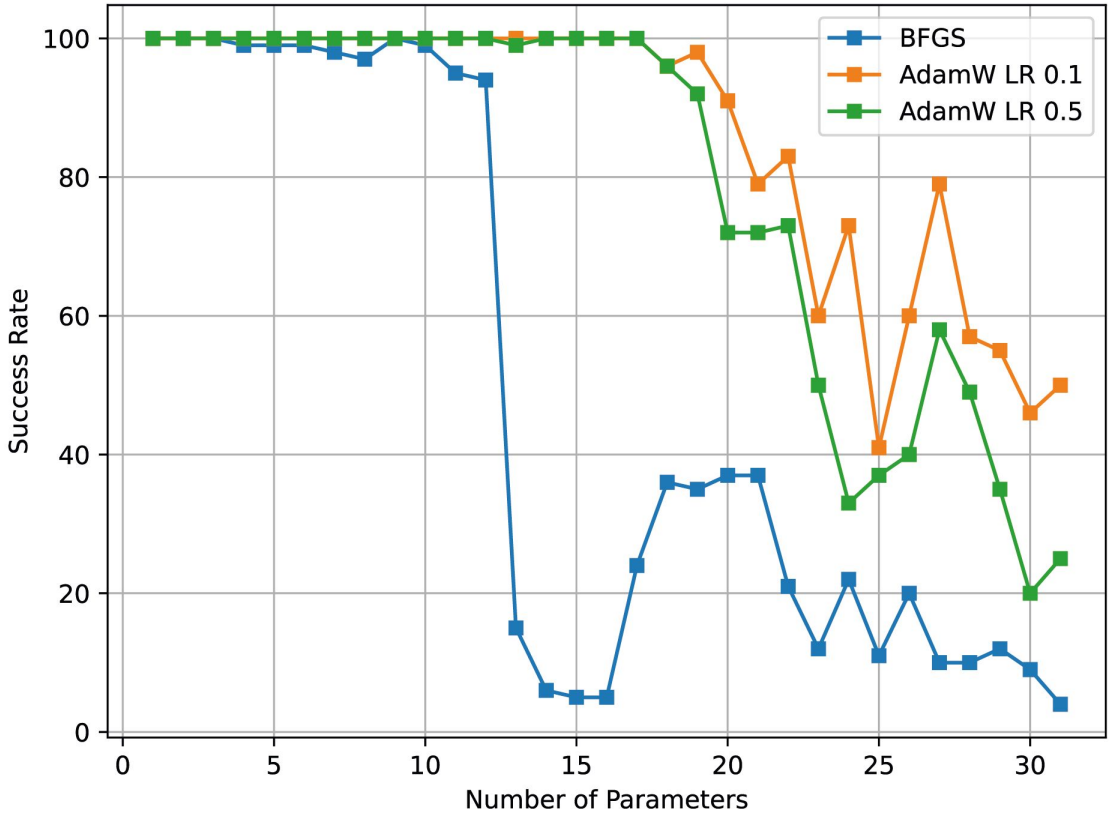
$$\mathrm{NRMSD} = \frac{\sqrt{\sum_{i=1}^{N} \left(log_{10}\left(y_{i,\mathrm{Differometor}}\right) - log_{10}\left(y_{i,\mathrm{FINESSE}}\right)\right)^2 / N}}{\sum_{i=1}^{N} \left(-log_{10}\left(y_{i,\mathrm{Differometor}}\right)\right) / N} \times 100\%$$



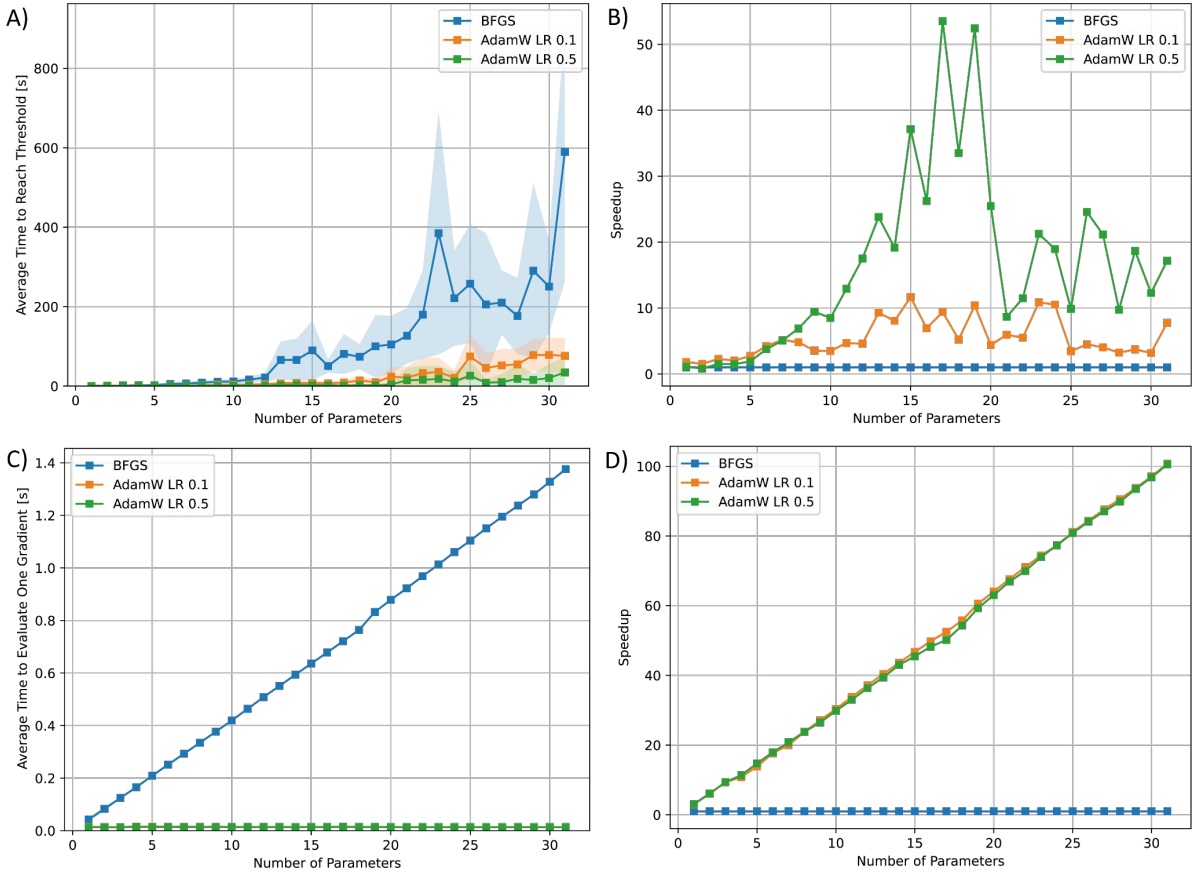
Major optimization speedups through autodiff



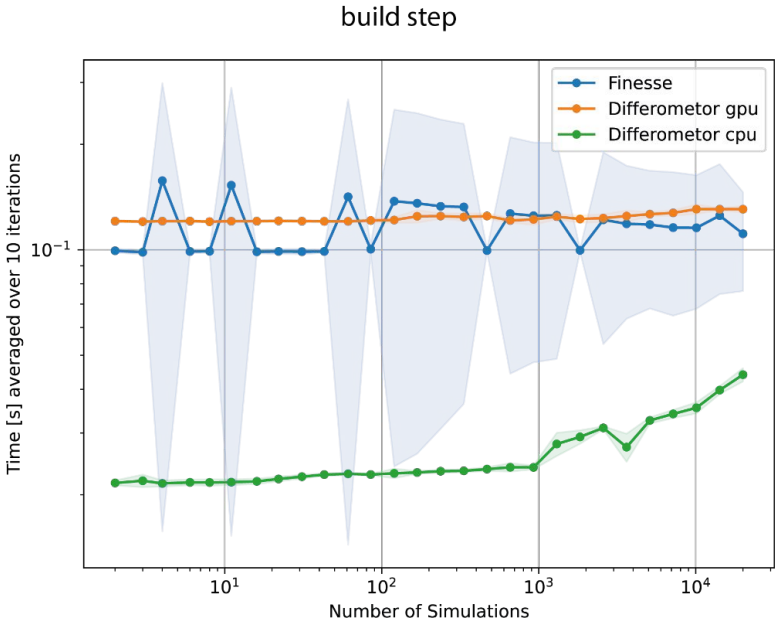
… speed up optimizations through auto-differentiation

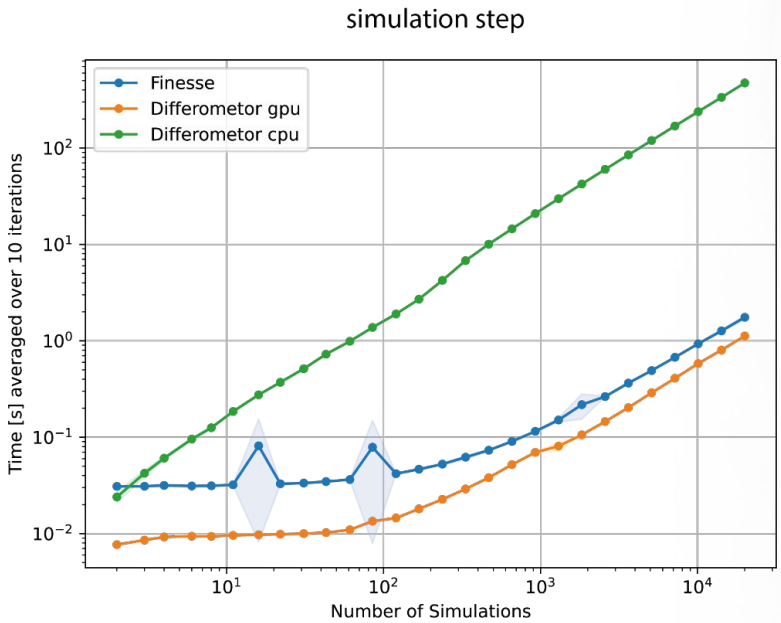


… speed up optimizations through auto-differentiation



… speed up simulations through GPU support and JIT compilation





Quantum Noise

To calculate $E(t)$, we first represent quantum fluctuations as noise in both amplitude and phase of a carrier field:

$$E(t) = [a_0 + n_a(t)]\, e^{\mathrm{i}\omega_0 t + n_\phi(t)/a_0} + \text{c.c} = [a_0 + n_a(t) + \mathrm{i}n_\phi(t)]\, e^{\mathrm{i}\omega_0 t} + \text{ c.c.} \tag{2.64}$$

where n_a and n_ϕ are real amplitudes of phase and amplitude fluctuations which in the frequency domain can form the complex noise

$$q(\omega) = n_a(\omega) + \mathrm{i}n_\phi(\omega). \tag{2.65}$$

$n_a(\omega)$ and $n_\phi(\omega)$ are characterized by Gaussian probability density functions with mean

Quantum Noise

$$
\begin{aligned}
\mathbb{M}\ \vec{q}_{\text{out}} &= \vec{q}_{\text{in}} \\
\mathbb{M}\ \vec{q}_{\text{out}}\ \vec{q}_{\text{out}}^{\,\dagger}\ \mathbb{M}^{\dagger} &= \vec{q}_{\text{in}}\ \vec{q}_{\text{in}}^{\,\dagger} \\
\vec{q}_{\text{out}}\ \vec{q}_{\text{out}}^{\,\dagger} &= \mathbb{M}^{-1}\vec{q}_{\text{in}}\ \vec{q}_{\text{in}}^{\,\dagger}\ \mathbb{M}^{-1\dagger} \\
\left\langle \vec{q}_{\text{out}}\ \vec{q}_{\text{out}}^{\,\dagger} \right\rangle &= \mathbb{M}^{-1} \left\langle \vec{q}_{\text{in}}\ \vec{q}_{\text{in}}^{\,\dagger} \right\rangle \mathbb{M}^{-1\dagger} \\
\mathbb{V}_o &= \mathbb{M}^{-1}\mathbb{V}_i\mathbb{M}^{-1\dagger}
\end{aligned}
$$