# A Differentiable Interferometer Simulator for the Computational Design of Gravitational Wave Detectors

Jonathan Klimesch, Yehonathan Drori, Rana X. Adhikari, Mario Krenn

**Nikhef Institute on September 3, 2025**

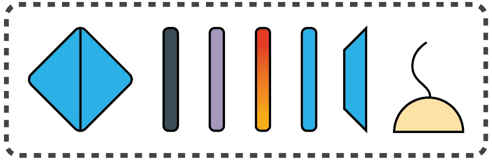
# Artificial Scientist Lab at the Tübingen Cybervalley

## Prof. Mario Krenn

## The Team

# Some Context ...

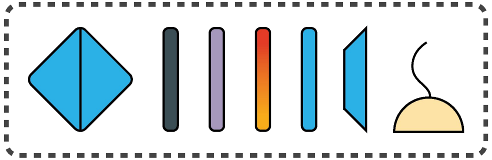
# Automated Search for New Quantum Experiments



Optical Element Toolbox

Krenn et al., Phys. Rev. Lett. 116 (2016)

# Automated Search for New Quantum Experiments



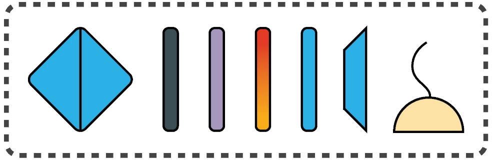
Optical Element Toolbox

$$\psi\rangle = \frac{1}{\sqrt{3}}\left(|0,0,0\rangle + |1,1,1\rangle + |2,2,2\rangle\right)$$

3D GHZ State

[Krenn et al., Phys. Rev. Lett. 116 (2016)]

# Automated Search for New Quantum Experiments



Optical Element Toolbox

Generate Random Setup

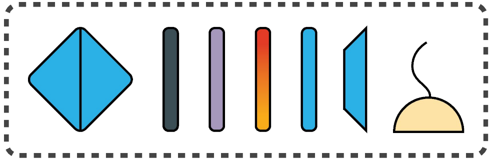
Simulate Setup &
Compute Objective

$$|\psi\rangle = \frac{1}{\sqrt{3}}\left(|0,0,0\rangle + |1,1,1\rangle + |2,2,2\rangle\right)$$

3D GHZ State

Krenn et al., Phys. Rev. Lett. 116 (2016)

# Automated Search for New Quantum Experiments



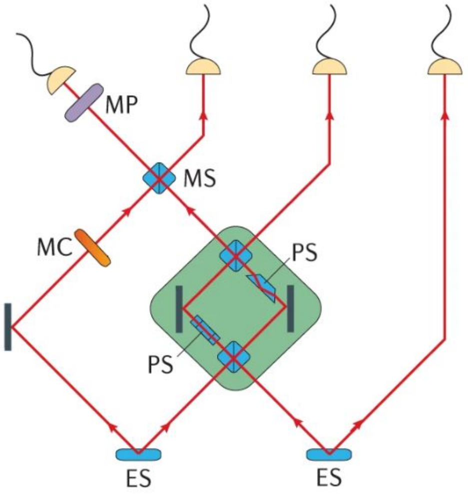
Optical Element Toolbox

Generate Random Setup

Simulate Setup & Compute Objective

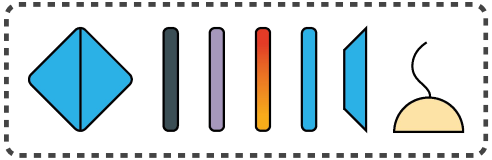
$$|\psi\rangle = \frac{1}{\sqrt{3}}\left(|0,0,0\rangle + |1,1,1\rangle + |2,2,2\rangle\right)$$

3D GHZ State




Krenn et al., Phys. Rev. Lett. 116 (2016)

# Automated Search for New Quantum Experiments



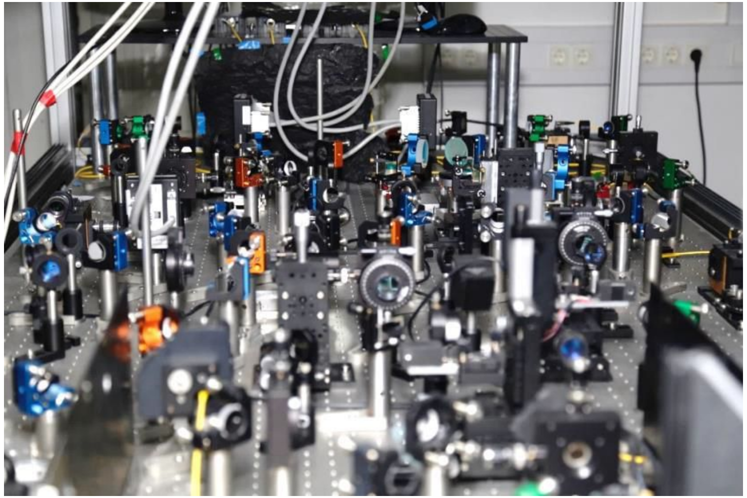
Optical Element Toolbox

Generate Random Setup

Simulate Setup &
Compute Objective

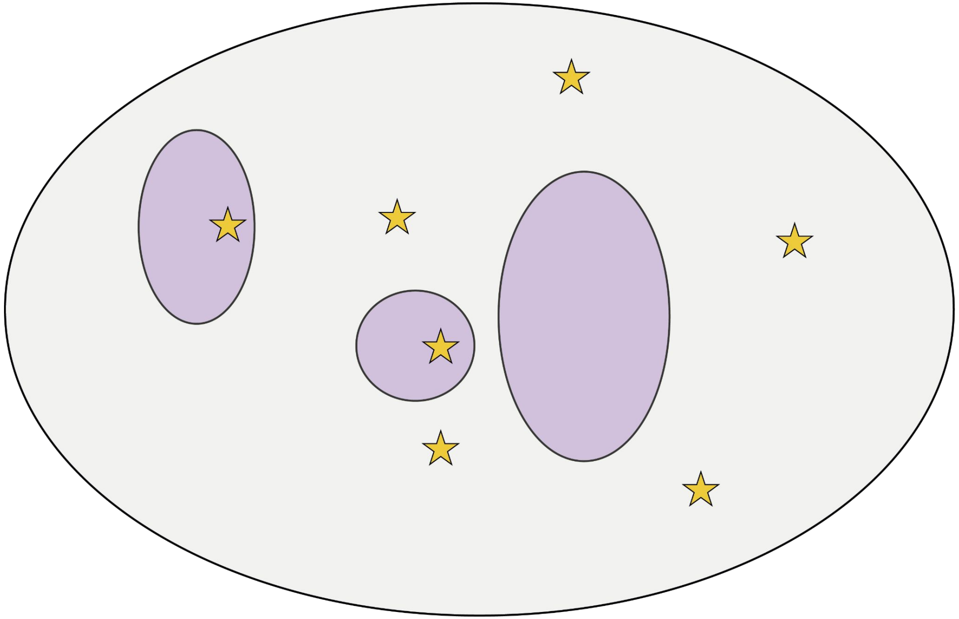
$$|\psi\rangle = \frac{1}{\sqrt{3}}\left(|0,0,0\rangle + |1,1,1\rangle + |2,2,2\rangle\right)$$

3D GHZ State



Erhard et al., Nature Photonics 12, 759 (2018)

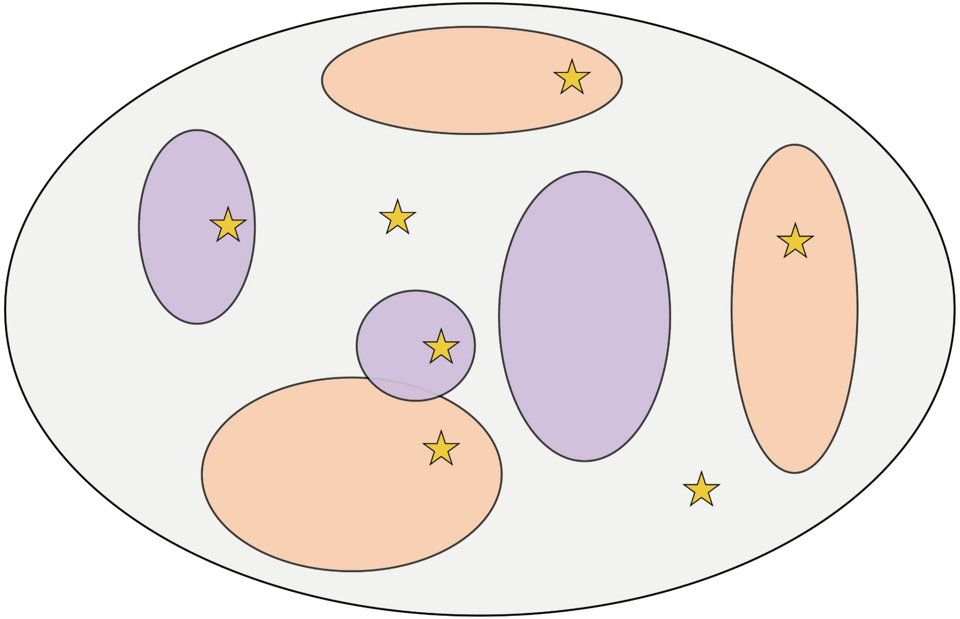
# Human Exploration



Ideas explored by human researchers

★ Exceptional Blueprints

# Computational Exploration



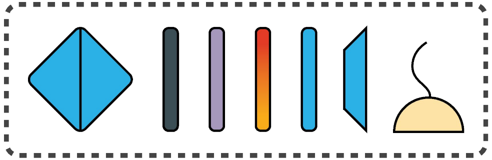
Ideas explored by human researchers

Ideas explored by computers

★ Exceptional Blueprints

# Can AI find unorthodox, but useful physics experiment designs?

# Automated Search for New Quantum Experiments



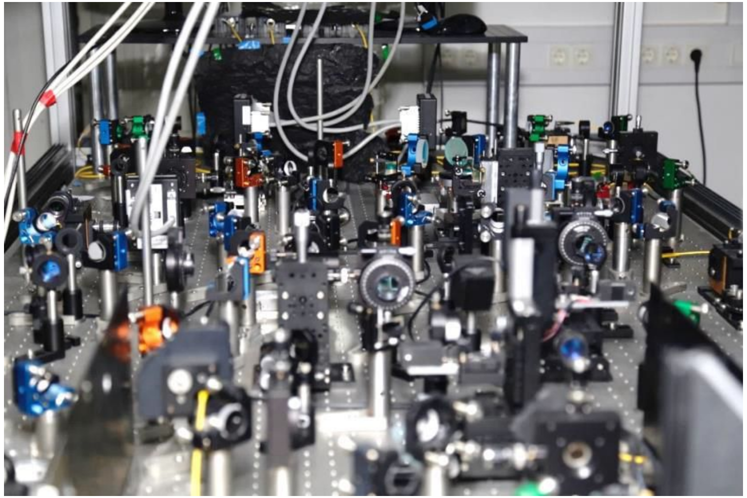
Optical Element Toolbox

Generate Random Setup

Simulate Setup & Compute Objective

$$\psi\rangle = \frac{1}{\sqrt{3}}\left(|0,0,0\rangle + |1,1,1\rangle + |2,2,2\rangle\right)$$

3D GHZ State



Erhard et al., Nature Photonics 12, 759 (2018)

# Automated Search for New Quantum Experiments



Optical Element Toolbox

Generate Random Setup

Simulate Setup & Compute Objective

$$|\psi\rangle = \frac{1}{\sqrt{3}}\left(|0,0,0\rangle + |1,1,1\rangle + |2,2,2\rangle\right)$$

JULY 2, 2021 | 8 MIN READ

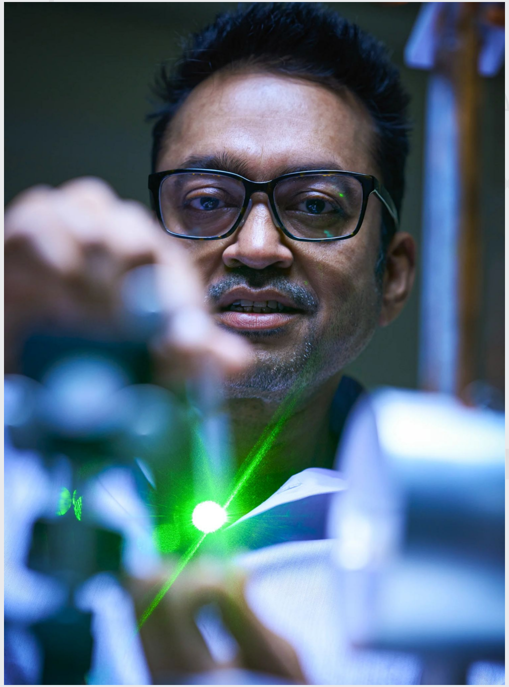
## AI Designs Quantum Physics Experiments beyond What Any Human Has Conceived

Originally built to speed up calculations, a machine-learning system is now making shocking progress at the frontiers of experimental quantum physics

BY ANIL ANANTHASWAMY EDITED BY LEE BILLINGS

Erhard et al., Nature Photonics 12, 759 (2018)

# Automated Search for New Quantum Experiments



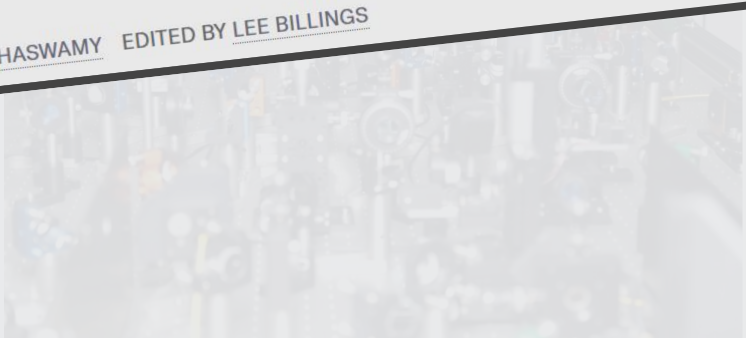
$$|\psi\rangle = \frac{1}{\sqrt{3}}\left(|0,0,0\rangle + |1,1,1\rangle + |2,2,2\rangle\right)$$

JULY 2, 2021 | 8 MIN READ

## AI Designs Quantum Physics Experiments beyond What Any Human Has Conceived

Originally built to speed up calculations, a machine-learning system is now making shocking progress at the frontiers of experimental quantum physics

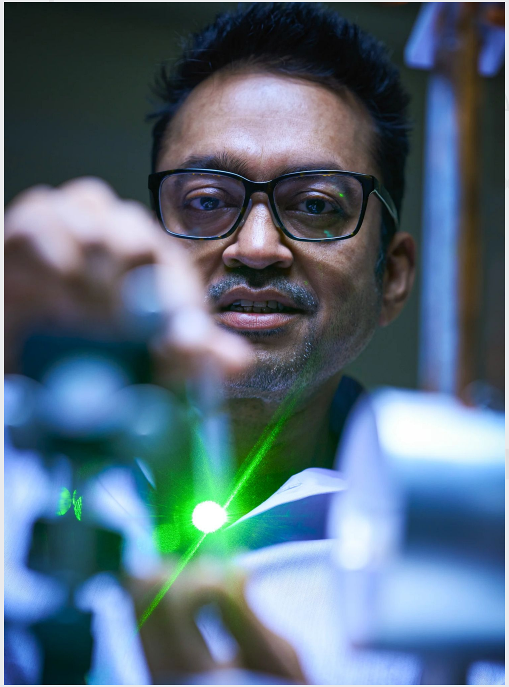
BY ANIL ANANTHASWAMY EDITED BY LEE BILLINGS



Simulate Setup & Compute Objective

Erhard et al., Nature Photonics 12, 759 (2018)

# Automated Search for New Quantum Experiments



OPEN ACCESS

PDF

## Digital Discovery of Interferometric Gravitational Wave Detectors

Share

Mario Krenn[1,*,†], Yehonathan Drori[2,*,‡], and Rana X Adhikari[2,§]

Show more

Phys. Rev. X **15**, 021012 – **Published 11 April, 2025**

DOI: https://doi.org/10.1103/PhysRevX.15.021012

Export Citation



### Abstract

Gravitational waves, detected a century after they were first theorized, are space-time distortions caused by some of the most cataclysmic events in the Universe, including black hole mergers and supernovae. The successful detection of these waves has been made possible by ingenious detectors designed by human experts. Beyond these successful designs, the vast space of experimental configurations remains largely unexplored, offering an exciting territory potentially rich in innovative and unconventional detection strategies. Here, we demonstrate an intelligent computational strategy to explore this enormous space, discovering unorthodox topologies for gravitational wave detectors that significantly outperform the currently best-known designs under realistic experimental constraints. This increases the potentially observable volume of the Universe by up to 50-fold. Moreover, by analyzing the best solutions from our superhuman algorithm, we uncover entirely new physics ideas at their core. At a bigger picture, our methodology can readily be extended to AI-driven design of experiments across wide domains of fundamental physics, opening fascinating new windows into the Universe.

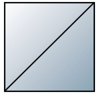
Simulate Setup & Compute Objective

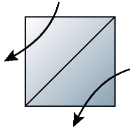
Erhard et al., Nature Photonics 12, 759 (2018)

# Discrete-Continuous Search Space

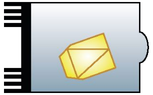

Laser (Power)

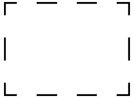

Mirror (Reflectivity, Tuning)


Beamsplitter (Reflectivity)


Directional Beamsplitter


Detector


Squeezer (dB, Angle)

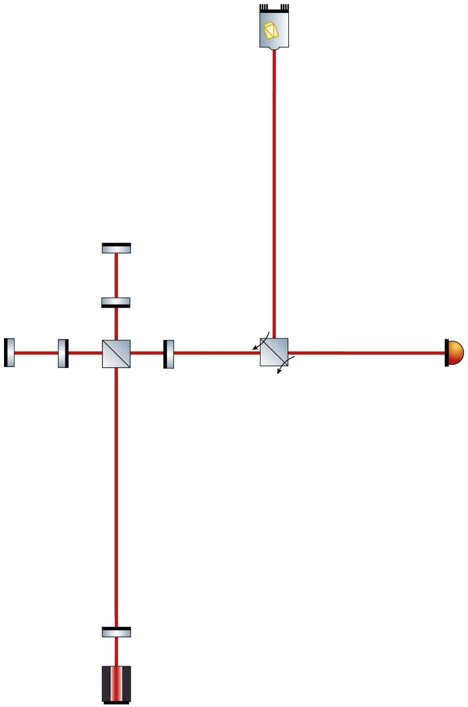

Space (Length)


Suspension (Mass)

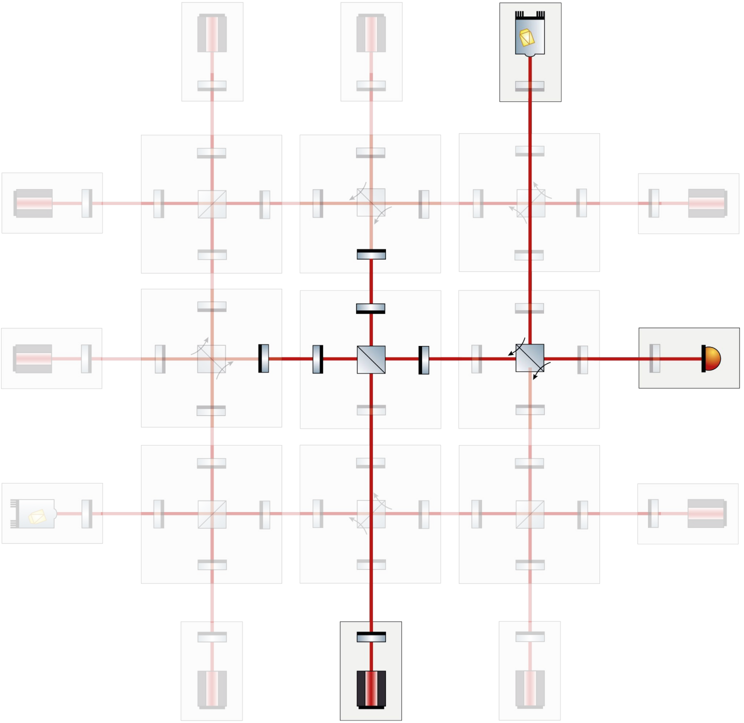
**Where should we place which component?**

**Which parameters should we choose?**

# Simplified aLIGO Setup

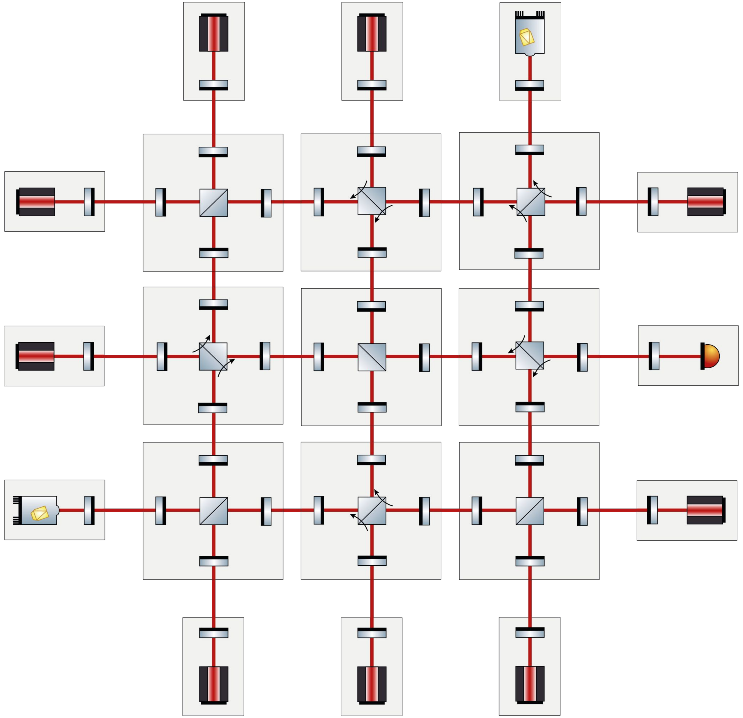


# Continuous and Highly Expressive Search Space



Krenn et al., Phys. Rev. X 15 (2025)

# Continuous and Highly Expressive Search Space

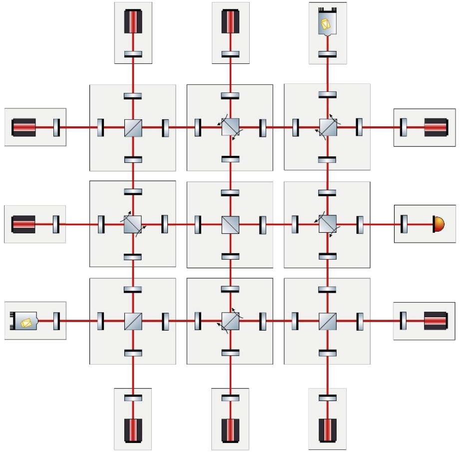


Quasi-Universal Interferometer
(UIFO)

Krenn et al., Phys. Rev. X 15 (2025)

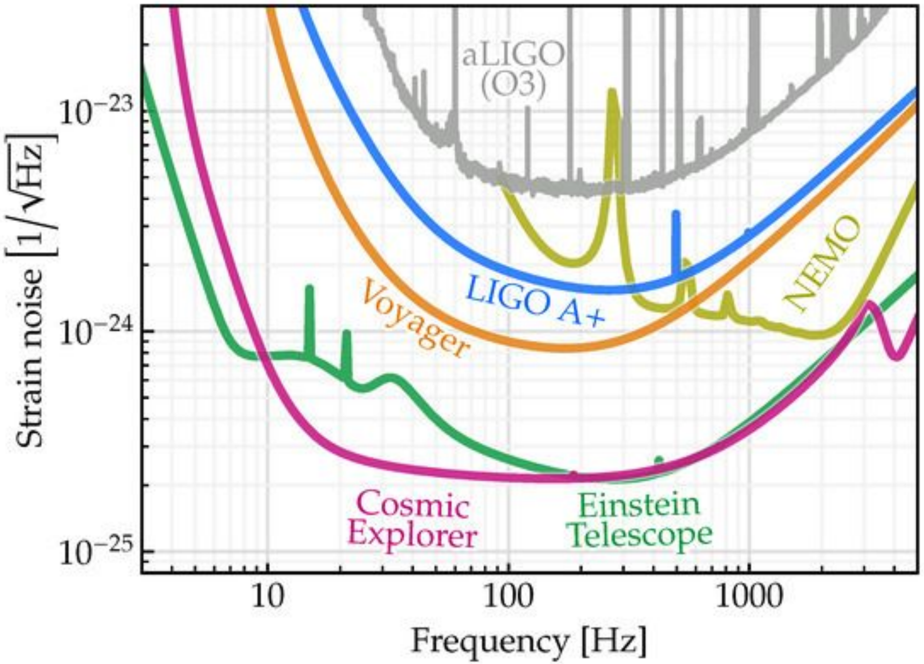
# Ingredients

1. Continuous Search Space



**What is our optimization goal?**

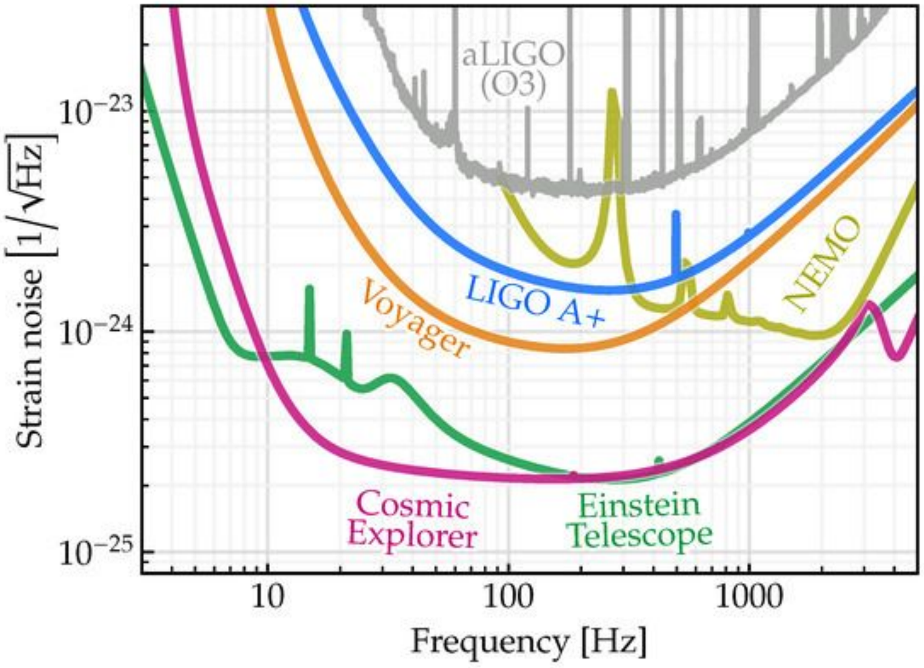
# The Objective Function




Hall E. et al., Galaxies 2022, 10(4), 90 (2022)

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

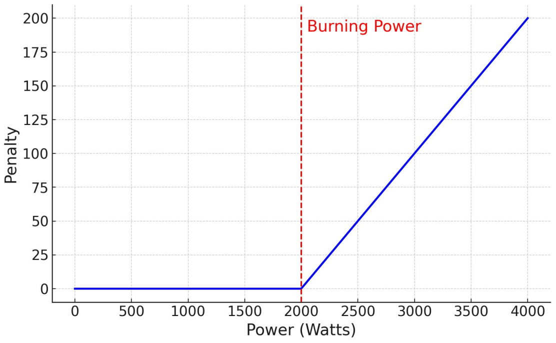
# The Objective Function




[Hall E. et al., Galaxies 2022, 10(4), 90 (2022)]

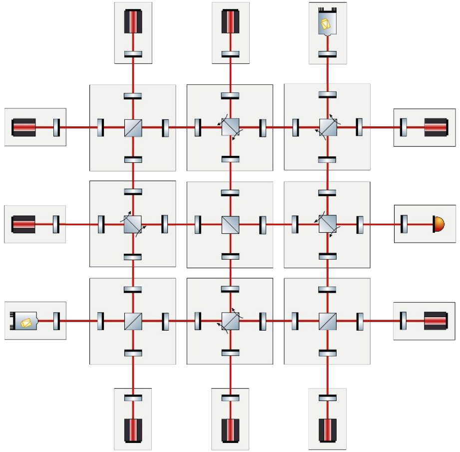
$$\mathcal{L}_{\text{Strain}} = \int_{f_0}^{f_1} \log(S(f))df$$

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



# Ingredients

1. Continuous Search Space



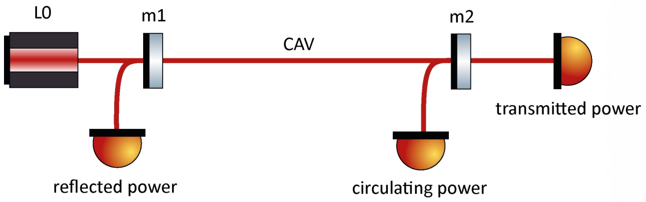
2. Objective Function

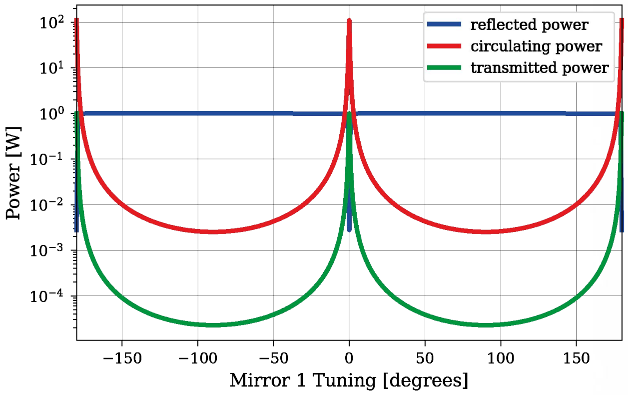
$$\mathcal{L}_{\text{Strain}} = \int_{f_0}^{f_1} \log(S(f))df$$

## How do we evaluate the objective function?

# Finesse - Frequency domain interferometer simulation software

```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)
```

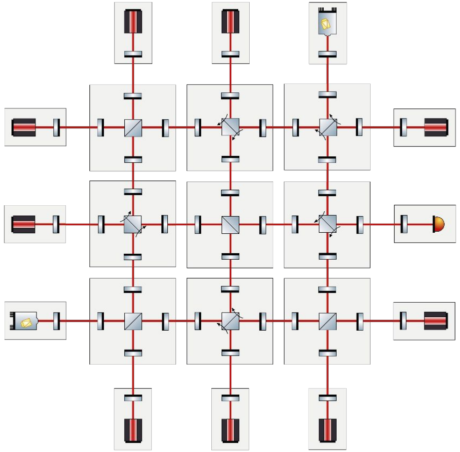



Brown D. et al., (2020)

# Ingredients

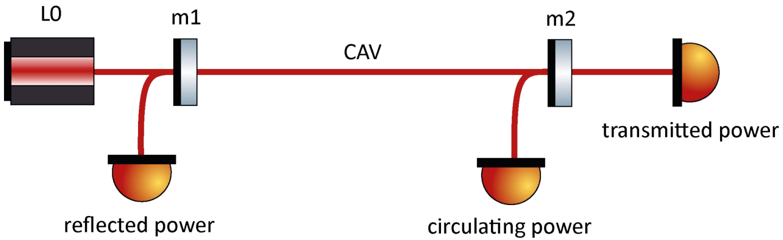
## 1. Continuous Search Space



## 2. Objective Function

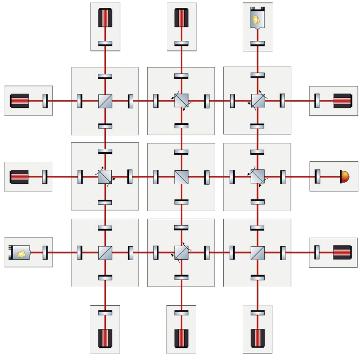
$$\mathcal{L}_{\text{Strain}} = \int_{f_0}^{f_1} \log(S(f))df$$

## 3. Simulator




## How do we optimize this?

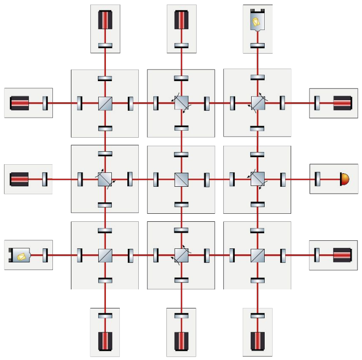
# Automated Search for New Gravitational Wave Detectors



Generate Random
UIFO

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

# Automated Search for New Gravitational Wave Detectors



Generate Random UIFO

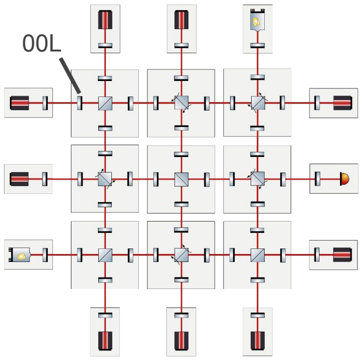
$$\mathcal{L}_{\text{Strain}} = \int_{f_0}^{f_1} \log(S(f))df$$

Simulate & Compute Objective

Calculate Gradients of Loss w.r.t. Parameters

Update Parameters

# Automated Search for New Gravitational Wave Detectors




Generate Random UIFO

Simulate & Compute Objective

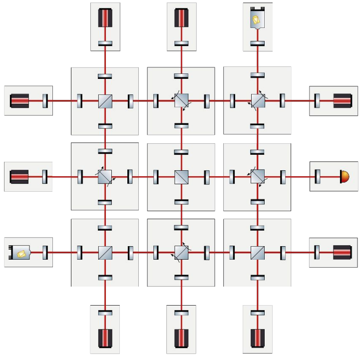
**Calculate Gradients of Loss w.r.t. Parameters**

Update Parameters

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

$$\begin{bmatrix} \frac{\partial \mathcal{L}_{\text{Strain}}}{\partial R_{00L}} \\ \frac{\partial \mathcal{L}_{\text{Strain}}}{\partial M_{00L}} \\ \vdots \end{bmatrix}$$

# Automated Search for New Gravitational Wave Detectors



Generate Random UIFO

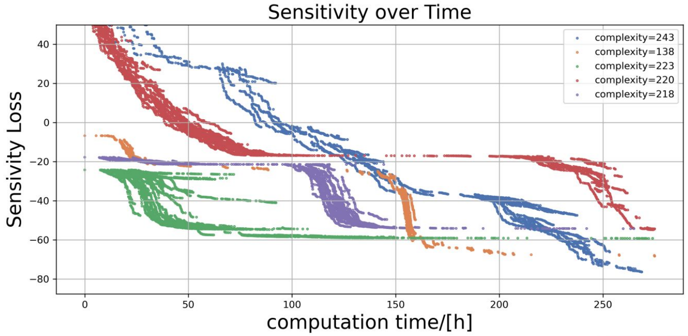
Simulate & Compute Objective

Calculate Gradients of Loss w.r.t. Parameters

Update Parameters

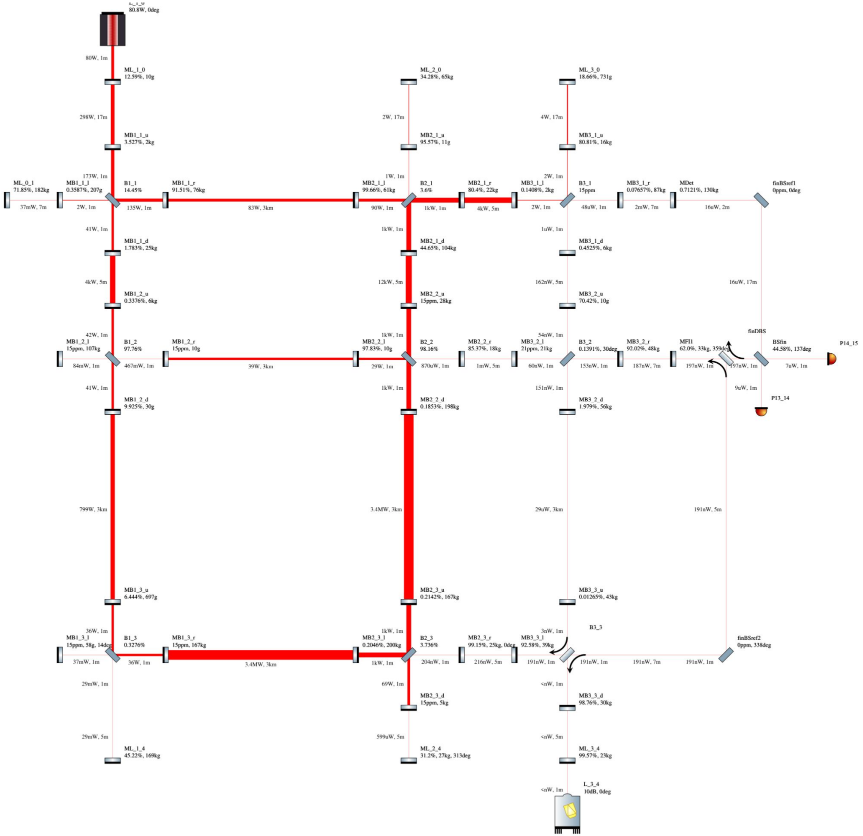
$$\mathcal{L}_{\text{Strain}} = \int_{f_0}^{f_1} \log(S(f))df$$

BFGS optimizations



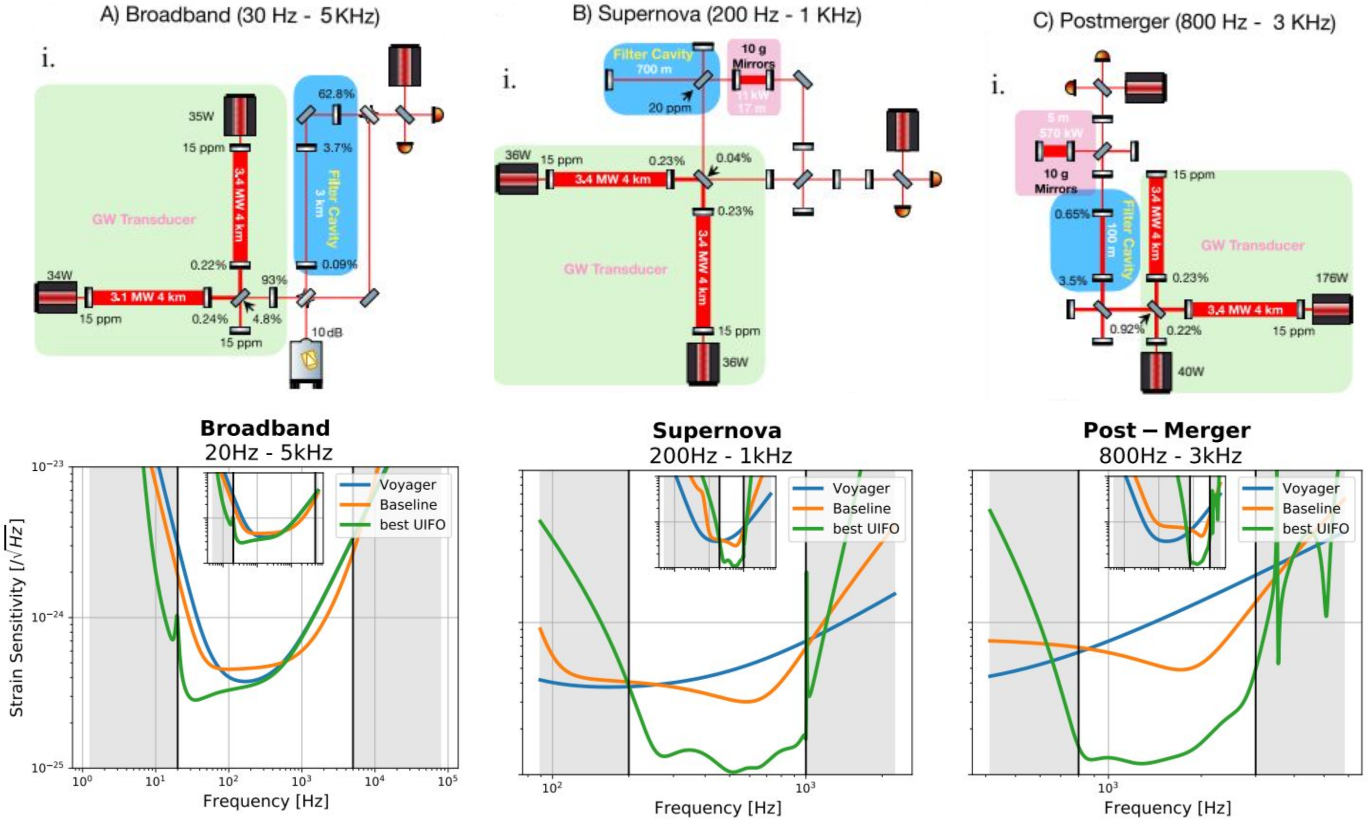

Krenn et al., Phys. Rev. X 15 (2025)

# Discovered Topologies - Raw Solutions



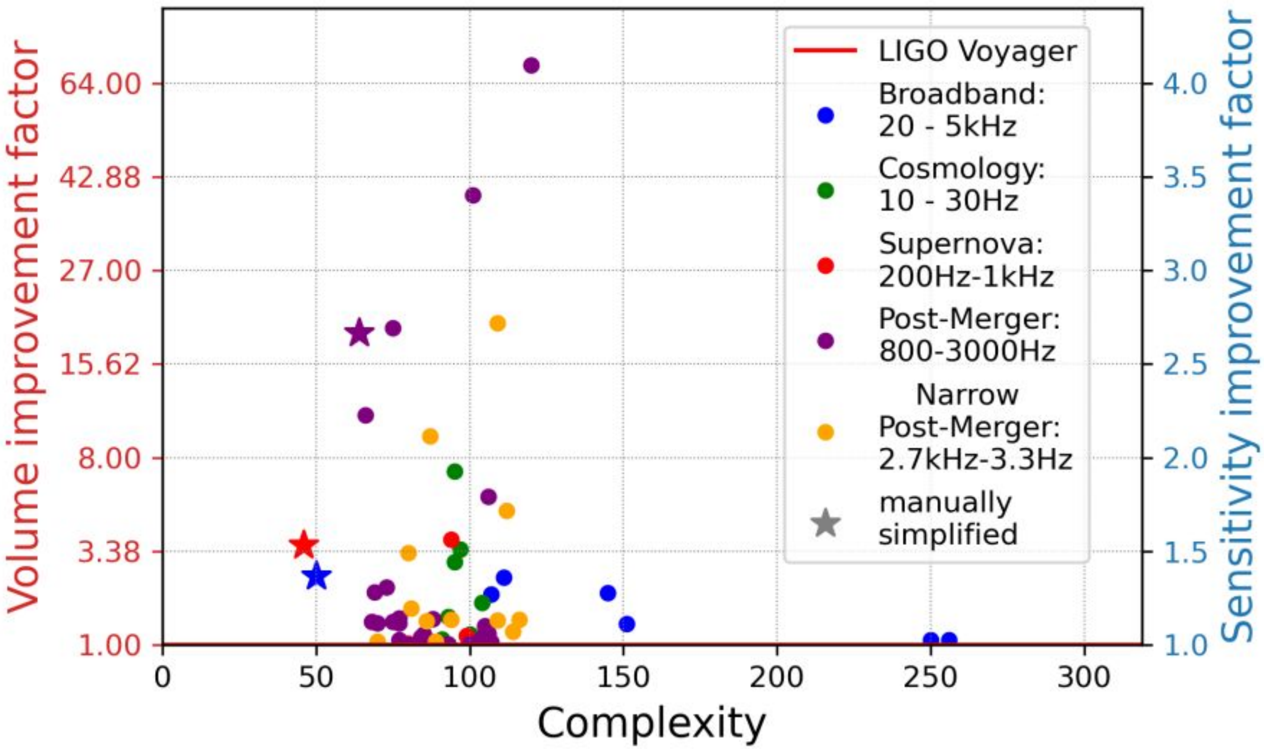
Krenn et al., Phys. Rev. X 15 (2025)

# Conceptualized UIFOs



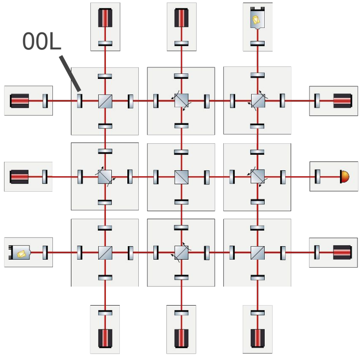

Krenn et al., Phys. Rev. X 15 (2025)

# Gravitational Wave Detector Zoo




Krenn et al., Phys. Rev. X 15 (2025)

# Automated Search for New Gravitational Wave Detectors




Generate Random UIFO

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

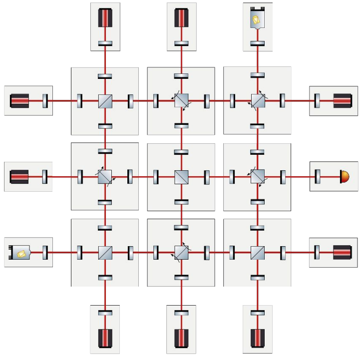
Update Parameters

Simulate & Compute Objective

$$\begin{bmatrix} \frac{\partial \mathcal{L}_{\text{Strain}}}{\partial R_{00L}} \\ \frac{\partial \mathcal{L}_{\text{Strain}}}{\partial M_{00L}} \\ \vdots \end{bmatrix}$$

**Calculate Gradients of Loss w.r.t. Parameters**

# Automated Search for New Gravitational Wave Detectors



Generate Random UIFO

Simulate & Compute Objective

**Calculate Gradients of Loss w.r.t. Parameters**

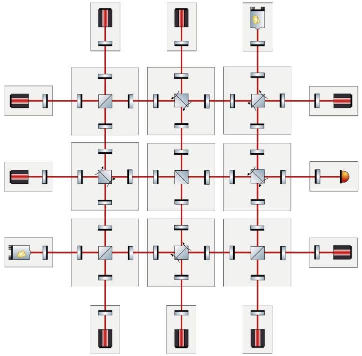
Update Parameters

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

$$f(x, y, z) = x + xy + z^2 y$$

$$\frac{\partial f}{\partial x} \approx \frac{f(x+h, y, z) - f(x, y, z)}{h}$$

# Automated Search for New Gravitational Wave Detectors



Generate Random
UIFO

Simulate &
Compute Objective

**Calculate Gradients of Loss w.r.t. Parameters**

Update Parameters

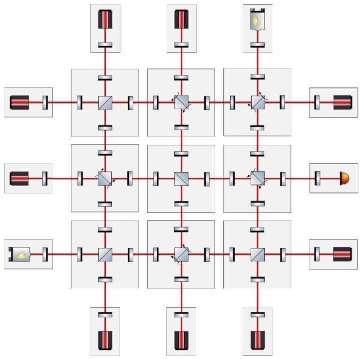
$$\mathcal{L}_{\text{Strain}} = \int_{f_0}^{f_1} \log(S(f))df$$

$$f(x, y, z) = x + xy + z^2 y$$

$$\frac{\partial f}{\partial x} \approx \frac{f(x+h, y, z) - f(x, y, z)}{h}$$

$$\frac{\partial f}{\partial y} \approx \frac{f(x, y+h, z) - f(x, y, z)}{h}$$

# Automated Search for New Gravitational Wave Detectors



Generate Random UIFO

Simulate & Compute Objective

**Calculate Gradients of Loss w.r.t. Parameters**

Update Parameters

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

$$f(x, y, z) = x + xy + z^2 y$$

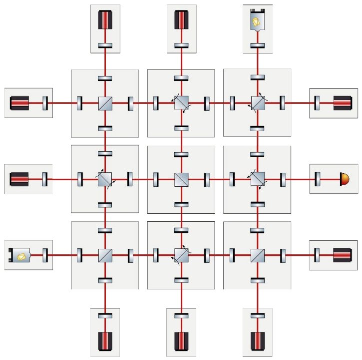
$$\frac{\partial f}{\partial x} \approx \frac{f(x+h, y, z) - f(x, y, z)}{h}$$

$$\frac{\partial f}{\partial y} \approx \frac{f(x, y+h, z) - f(x, y, z)}{h}$$

$$\frac{\partial f}{\partial z} \approx \frac{f(x, y, z+h) - f(x, y, z)}{h}$$

$$\nabla f \approx \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}\right)$$

# Automated Search for New Gravitational Wave Detectors



Generate Random UIFO

Simulate & Compute Objective

**Calculate Gradients of Loss w.r.t. Parameters**

Update Parameters

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

$$f(x, y, z) = x + xy + z^2 y$$

$$\frac{\partial f}{\partial x} \approx \frac{f(x+h, y, z) - f(x, y, z)}{h}$$

$$\frac{\partial f}{\partial y} \approx \frac{f(x, y+h, z) - f(x, y, z)}{h}$$

$$\frac{\partial f}{\partial z} \approx \frac{f(x, y, z+h) - f(x, y, z)}{h}$$

$$\nabla f \approx \left( \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z} \right)$$

# Can we make Finesse differentiable?

# Making something differentiable $f(x, y, z) = x + xy + z^2 y$

# Making something differentiable $f(x, y, z) = x + xy + z^2 y$

Replace with
learned model

$$\approx \hat{f}(x, y, z; \theta)$$

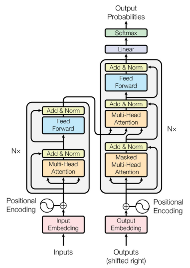
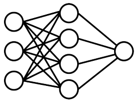
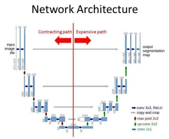


Figure 1: The Transformer - model architecture.



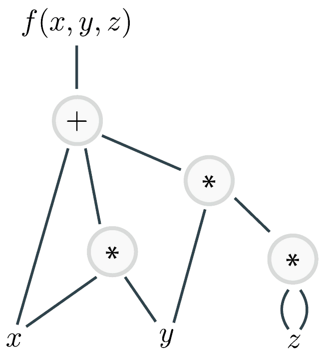


# Making something differentiable $f(x, y, z) = x + xy + z^2 y$

Replace with learned model

Rewrite in autodiff framework

$\approx \hat{f}(x, y, z; \theta)$




$+$ $x' + y'$

$*$ $x'y + xy'$

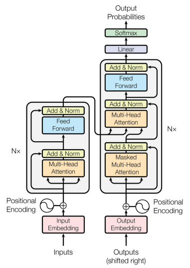
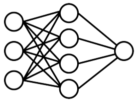
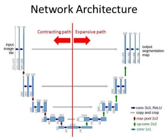



Figure 1: The Transformer - model architecture.



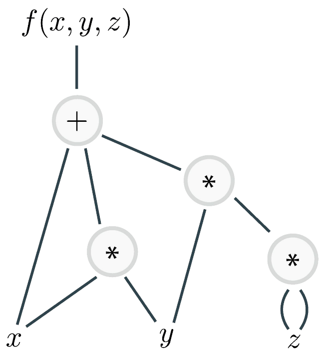



JAX PyTorch TensorFlow julia

# Making something differentiable $f(x, y, z) = x + xy + z^2 y$

| Replace with learned model | Rewrite in autodiff framework | Augment source code for autodiff |
|---|---|---|

$$\approx \hat{f}(x, y, z; \theta)$$




+ $x' + y'$

* $x'y + xy'$

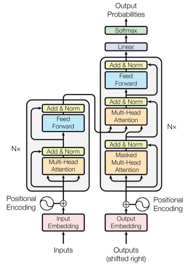
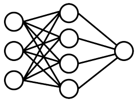
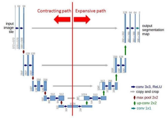



Figure 1: The Transformer - model architecture.



Network Architecture



JAX

PyTorch

TensorFlow

julia

Enzyme AD

Clad

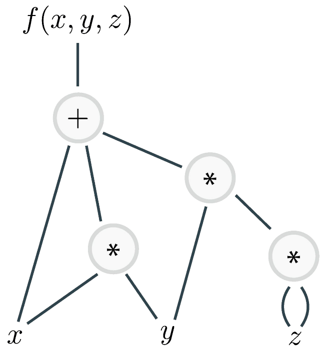
Derivgrind

TAPENADE

# Making something differentiable

$$f(x, y, z) = x + xy + z^2 y$$

Rewrite in autodiff framework



# - High performance array computing

**Features:**

- GPU support & vectorization
- Just-in-Time compilation
- Automatic Differentiation

J. Bradbury et al. (2018)

# JAX - High performance array computing

**Features:**

- GPU support & vectorization
- Just-in-Time compilation
- Automatic Differentiation

**Constraints:**

- Only pure functions
- No conditionals
- Only static shapes

J. Bradbury et al. (2018)

# JAX - High performance array computing

**Features:**

- GPU support & vectorization
- Just-in-Time compilation
- Automatic Differentiation

**Constraints:**

- Only pure functions
- No conditionals
- Only static shapes

```python
import jax

class Counter:
  def __init__(self):
    self.n = 0

  def count(self) -> int:
    self.n += 1
    return self.n

counter = Counter()
fast_count = jax.jit(counter.count)

# This results in 1, 1, 1!
for _ in range(3):
  print(fast_count())
```

J. Bradbury et al. (2018)

# JAX - High performance array computing

**Features:**

- GPU support & vectorization
- Just-in-Time compilation
- Automatic Differentiation

**Constraints:**

- Only pure functions
- No conditionals
- Only static shapes

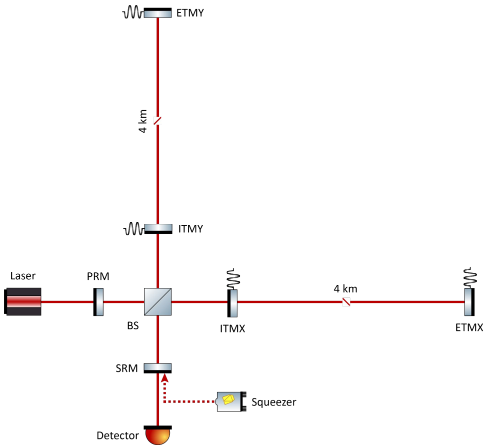
```python
@jit
def f(x):
  if x < 3:
    return 3. * x ** 2
  else:
    return -4 * x

# This will fail!
f(2)
```

J. Bradbury et al. (2018)

# Rewriting Finesse from Scratch in JAX

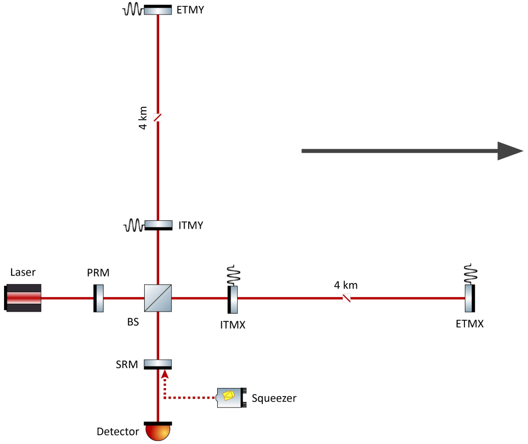
# Rewriting Finesse from Scratch in JAX

## Build Step

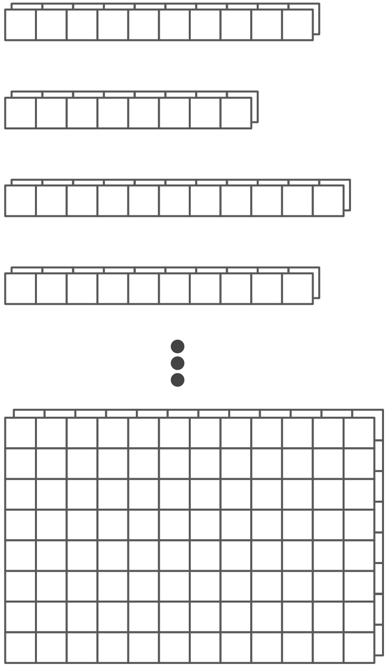


# Rewriting Finesse from Scratch in JAX

## Build Step




## Index & Value Arrays



Parameters

Indices of optimized parameters

Frequency values

Function Indices

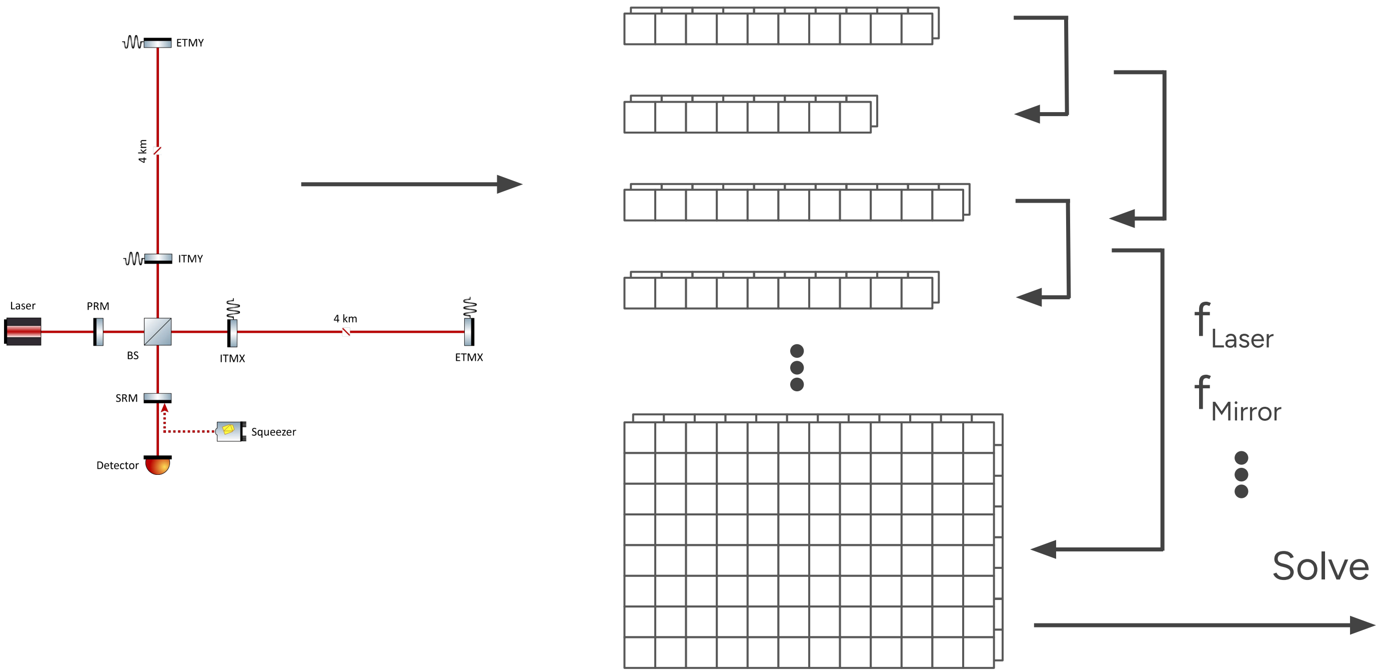
System Matrix

# Rewriting Finesse from Scratch in

## Build Step

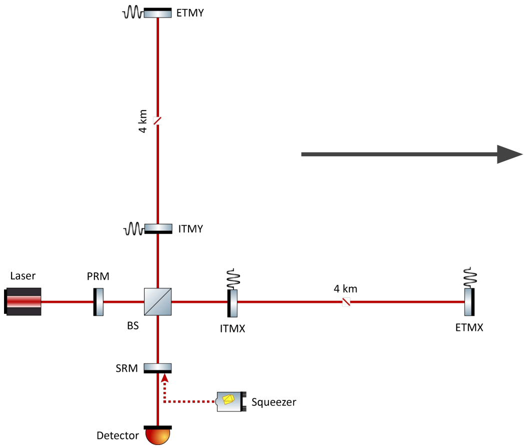
## Index & Value Arrays

## Simulation

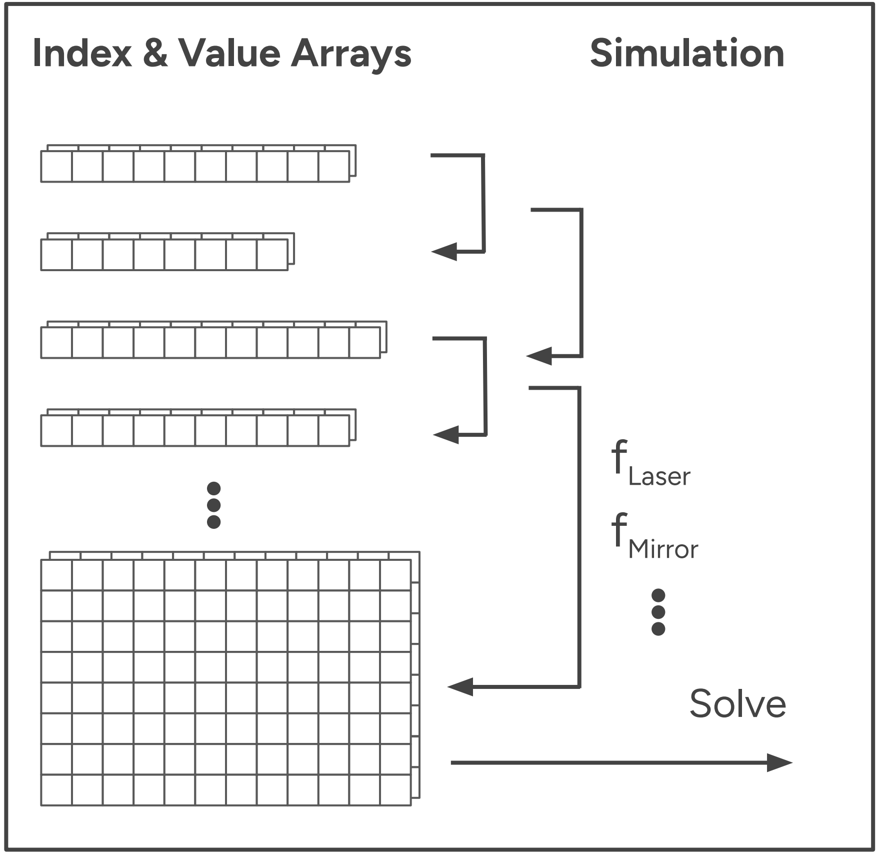


# Rewriting Finesse from Scratch in JAX

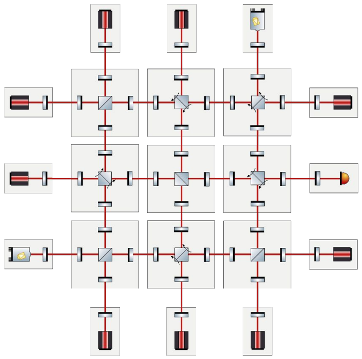
## Build Step




## JIT Compilation



# JIT Compiled Gradient Calculation



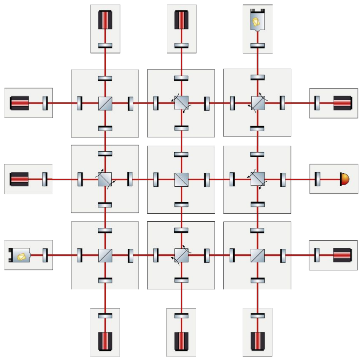
Generate Random UIFO

Simulate & Compute Objective

Calculate Gradients of Loss w.r.t. Parameters

Update Parameters

# JIT Compiled Gradient Calculation



Generate Random UIFO

Update Parameters

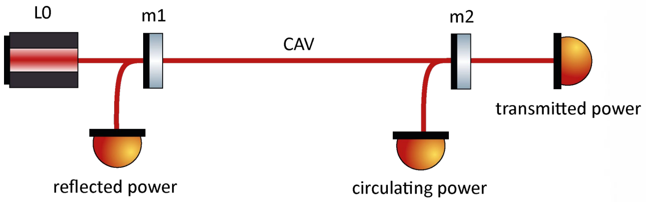
Simulate & Compute Objective

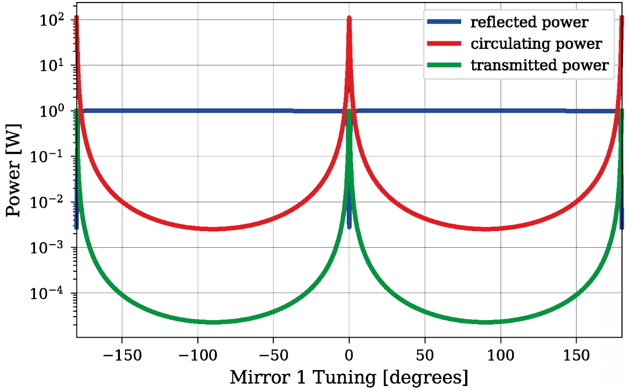
Calculate Gradients of Loss w.r.t. Parameters

**JIT Compilation**

# Differometor - Differentiable Interferometer Simulator

```python
import differometor as df
import jax.numpy as jnp
from differometor.components import power_detector
import matplotlib.pyplot as plt

# define a simple cavity setup with three detectors
S = df.Setup()
S.add("laser", "l0", power=1)
S.add("mirror", "m0", reflectivity=0.99, loss=0)
S.add("mirror", "m1", reflectivity=0.991, loss=0)
S.space("l0", "m0", length=1)
S.space("m0", "m1", length=1)
S.add("detector", "refl", target="m0", port="left", direction="out")
S.add("detector", "circ", target="m1", port="left", direction="in")
S.add("detector", "trns", target="m1", port="right", direction="out")

# set the tuning range
tunings = jnp.linspace(-180, 180, 400)

# run the simulation with the tuning as the changing parameter
carrier, signal, noise, detector_ports, *_ = df.run(S, [("m0", "tuning")],
↪  tunings)

# calculate the power
powers = power_detector(carrier)

# plot the power at the detector ports
plt.figure()
plt.plot(tunings, powers[detector_ports[0]], label="refl")
plt.plot(tunings, powers[detector_ports[1]], label="circ")
plt.plot(tunings, powers[detector_ports[2]], label="trns")
plt.yscale("log")
plt.xlabel("Tuning (degrees)")
plt.ylabel("Power (W)")
plt.grid()
plt.legend()
plt.tight_layout()
plt.savefig("output_cavity.png")
```



# Differometor Components

**Optics:**

- Mirror
- Beamsplitter
- Directional Beamsplitter

# Differometor Components

**Optics:**

- Mirror
- Beamsplitter
- Directional Beamsplitter

**Sources:**

- Laser (only 0 Hz offset)
- Squeezer
- Signal Generator
  - Space Length
  - Laser Amplitude
  - Laser Frequency

# Differometor Components

**Optics:**

- Mirror
- Beamsplitter
- Directional Beamsplitter

**Sources:**

- Laser (only 0 Hz offset)
- Squeezer
- Signal Generator
  - Space Length
  - Laser Amplitude
  - Laser Frequency

**Connectors:**

- Space
- Nothing

# Differometor Components

**Optics:**

- Mirror
- Beamsplitter
- Directional Beamsplitter

**Sources:**

- Laser (only 0 Hz offset)
- Squeezer
- Signal Generator
    - Space Length
    - Laser Amplitude
    - Laser Frequency

**Connectors:**

- Space
- Nothing

**Detectors:**

- Power Detector
- Demodulating Power Detector
- Quantum Noise Detector
- Balanced Homodyne Detector

# Differometor Components

**Optics:**

- Mirror
- Beamsplitter
- Directional Beamsplitter

**Sources:**

- Laser (only 0 Hz offset)
- Squeezer
- Signal Generator
    - Space Length
    - Laser Amplitude
    - Laser Frequency

**Connectors:**

- Space
- Nothing

**Detectors:**

- Power Detector
- Demodulating Power Detector
- Quantum Noise Detector
- Balanced Homodyne Detector

**Mechanical Elements:**

- Free Mass (only z node)

# Differometor Components

**Optics:**

- Mirror
- Beamsplitter
- Directional Beamsplitter

**Sources:**

- Laser (only 0 Hz offset)
- Squeezer
- Signal Generator
  - Space Length
  - Laser Amplitude
  - Laser Frequency

**Connectors:**

- Space
- Nothing

**Detectors:**

- Power Detector
- Demodulating Power Detector
- Quantum Noise Detector
- Balanced Homodyne Detector

**Mechanical Elements:**

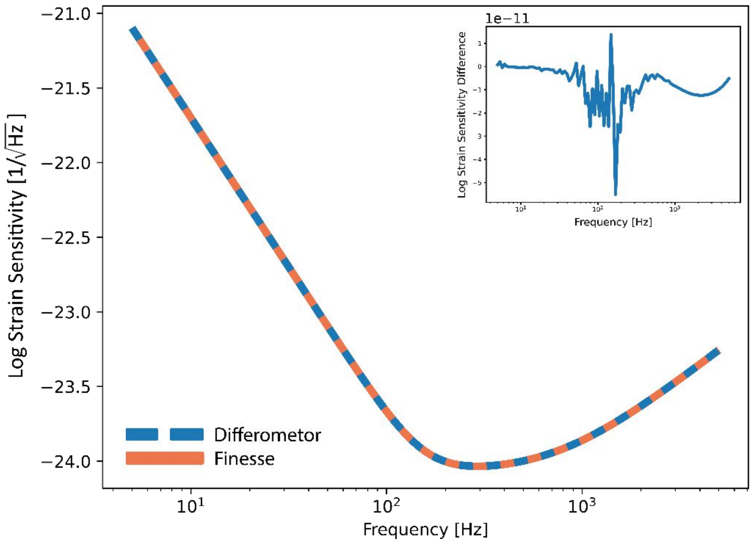
- Free Mass (only z node)

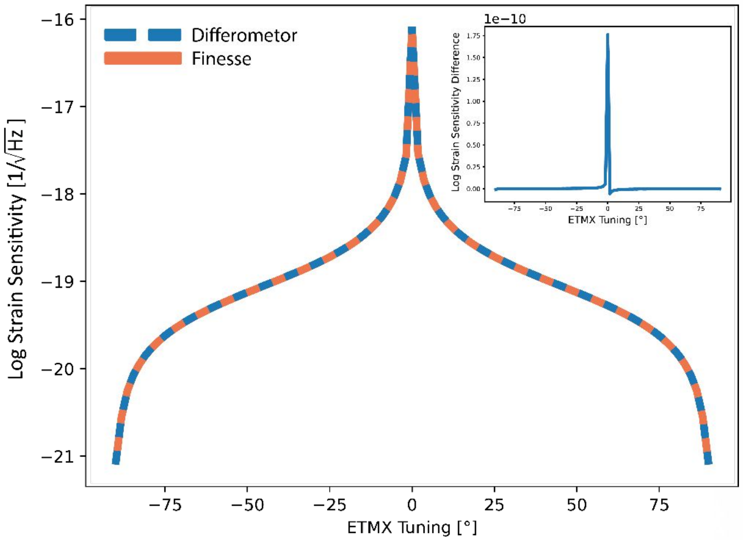
❌ **Higher Order Modes**

❌ **Control Loops**

❌ ...

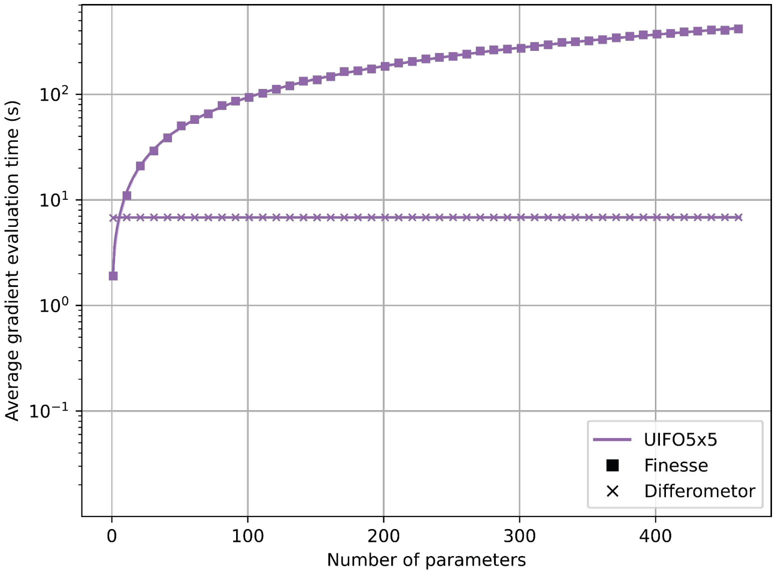
# Accuracy - Close Agreement with Finesse

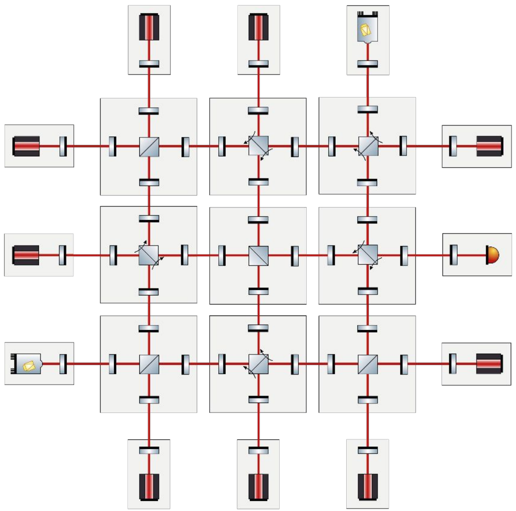



Simplified aLIGO

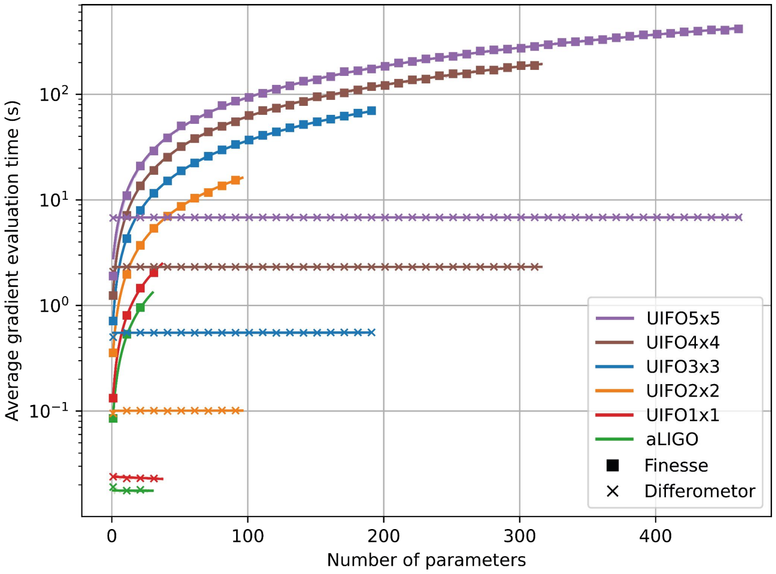
# Gradient Computation - Speed Gains through Autodiff

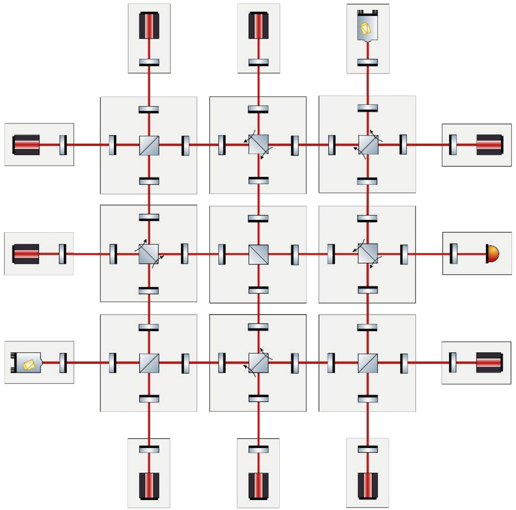


Nvidia Quadro RTX 6000 GPU vs. Intel Xeon Gold 6130 CPU

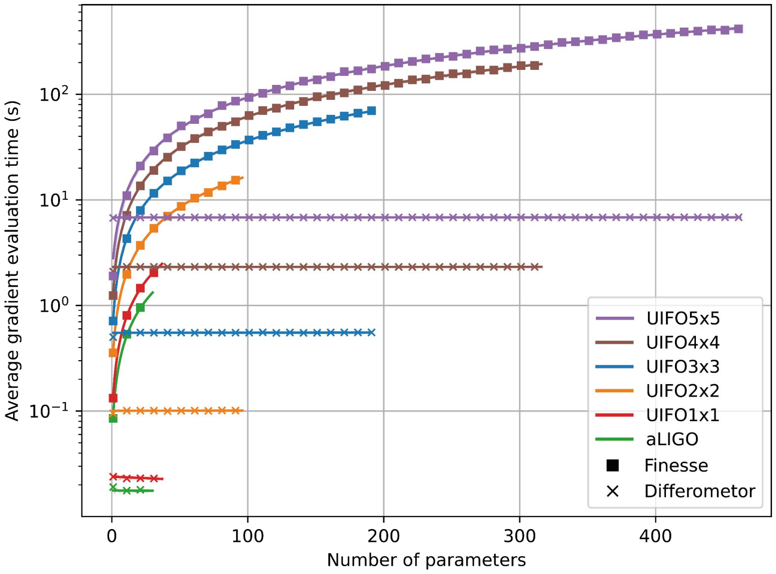
# Gradient Computation - Speed Gains through Autodiff

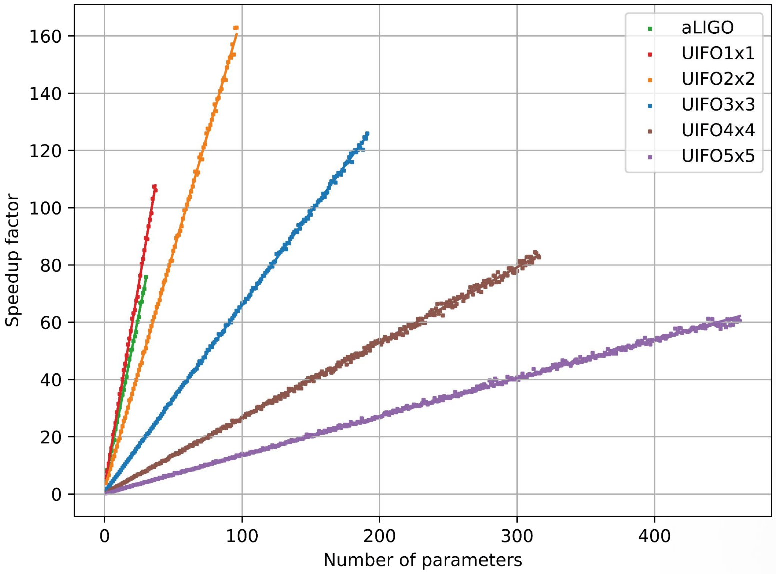


Nvidia Quadro RTX 6000 GPU vs. Intel Xeon Gold 6130 CPU

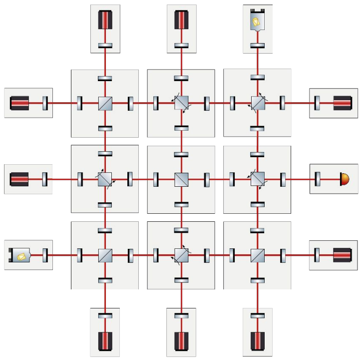
# Gradient Computation - Speed Gains through Autodiff




Nvidia Quadro RTX 6000 GPU vs. Intel Xeon Gold 6130 CPU

# Gradient Computation - Speed Gains through Autodiff



Generate Random UIFO

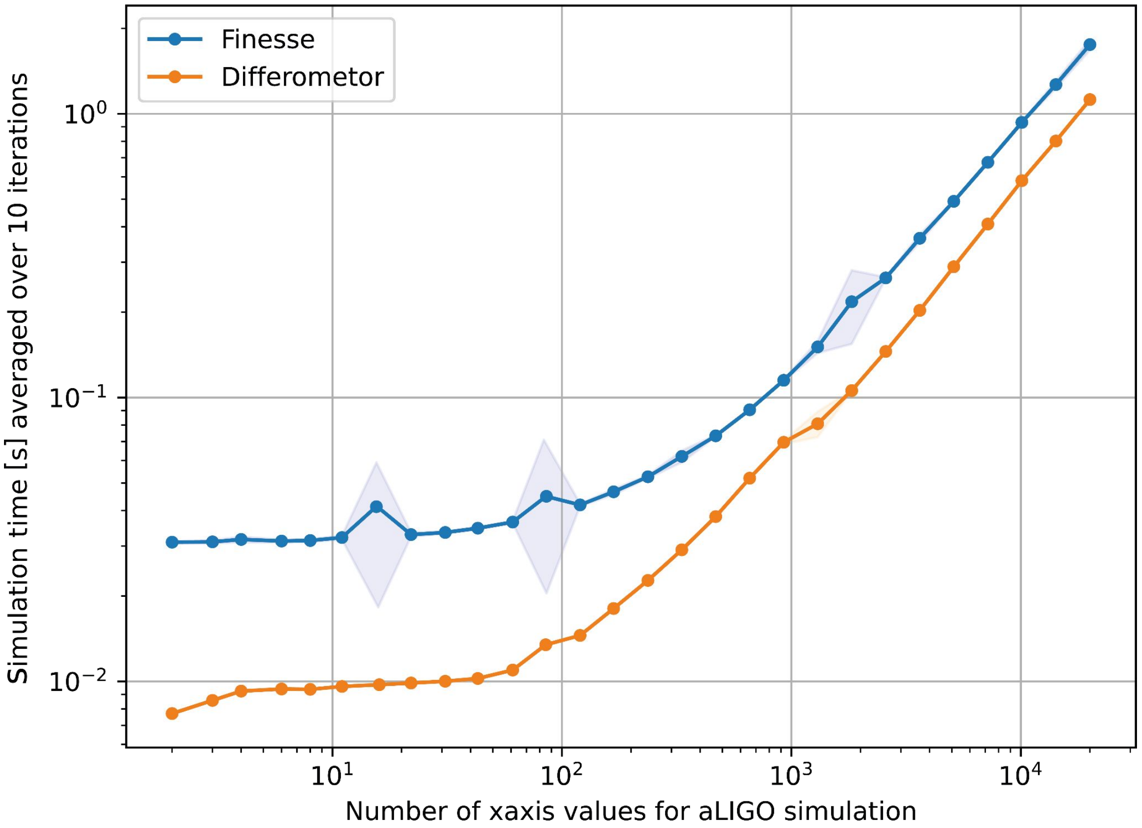
Update Parameters

Simulate & Compute Objective

**Calculate Gradients of Loss w.r.t. Parameters**

JIT Compilation

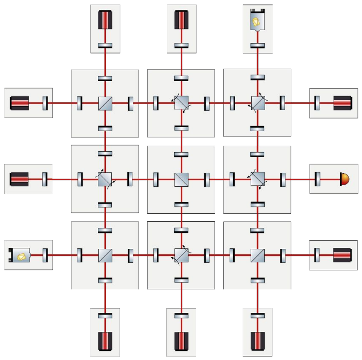
# Simulation - Speed Gains through GPU and JIT




Simplified aLIGO

Nvidia Quadro RTX 6000 GPU
vs. Intel Xeon Gold 6130 CPU

# Simulation - Speed Gains through GPU and JIT



Generate Random UIFO

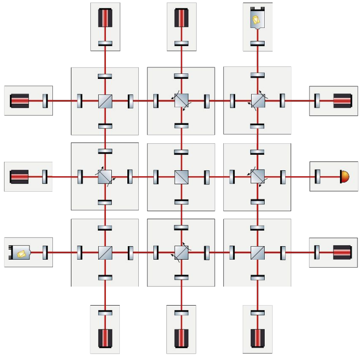
Update Parameters

**Simulate & Compute Objective**

Calculate Gradients of Loss w.r.t. Parameters

JIT Compilation

# Differometor Compilation is Slow



Generate Random UIFO

Update Parameters

Simulate & Compute Objective

Calculate Gradients of Loss w.r.t. Parameters

**Simplified aLIGO:**

- **Differometor: 40.4 s**
- **Finesse: 1.4 s**

**JIT Compilation**

# What are our plans?

# Mario's ERC Grant

erc
European Research Council

## Artificial Scientific Discovery of advanced Quantum Hardware with high-performance Simulators

Fact Sheet

### Project description

DE EN ES FR IT PL

#### Quantum experiment design with AI and high-performance simulators

The rapid advancements in generating, controlling, and measuring complex quantum systems have ushered in a new technological era based on quantum superposition and entanglement. However, designing these advanced experiments and hardware has grown too intricate for human researchers to handle alone. To unlock the vast potential of quantum mechanics, AI has been introduced into experiment design, but current methodologies remain limited. This creates a significant barrier to fully exploiting quantum physics' transformative possibilities. The ERC-funded ARTDISQ project aims to break these limitations by leveraging JAX, a cutting-edge computational framework, to build high-performance physical simulators. These simulators will power AI-driven breakthroughs in quantum-enhanced gravitational wave detection, advanced imaging techniques, and revolutionary quantum hardware design.

Hide the project objective

Project Information

**ArtDisQ**
Grant agreement ID: 101165179

**DOI**
10.3030/101165179

**EC signature date**
6 December 2024

**Start date**
1 January 2025

**End date**
31 December 2029

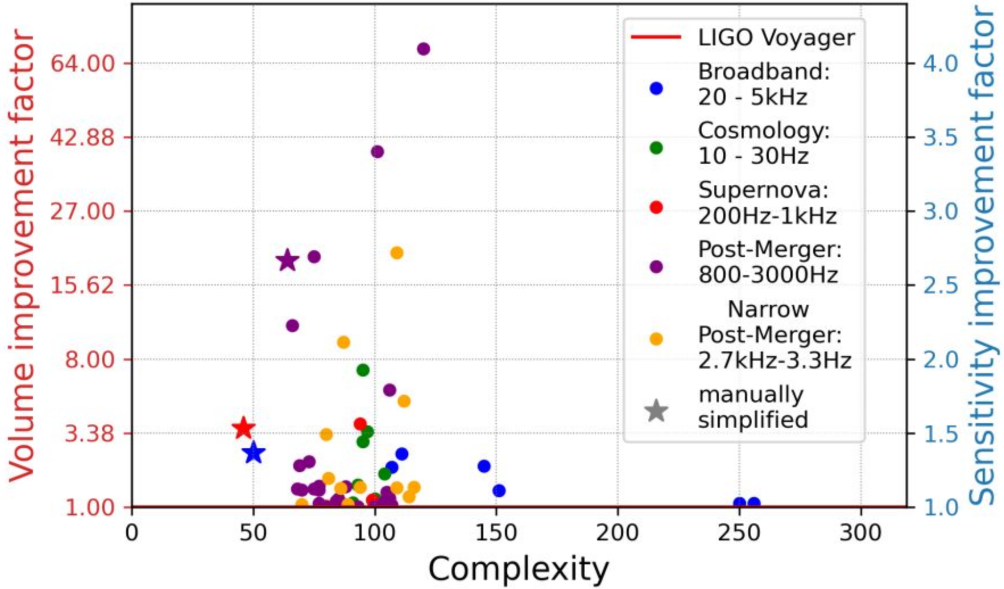
**Funded under**
European Research Council (ERC)

**Total cost**
€ 1 499 221,25

**EU contribution**
€ 1 499 221,00

# Gravitational Wave Detector Zoo 2.0

?
?
?

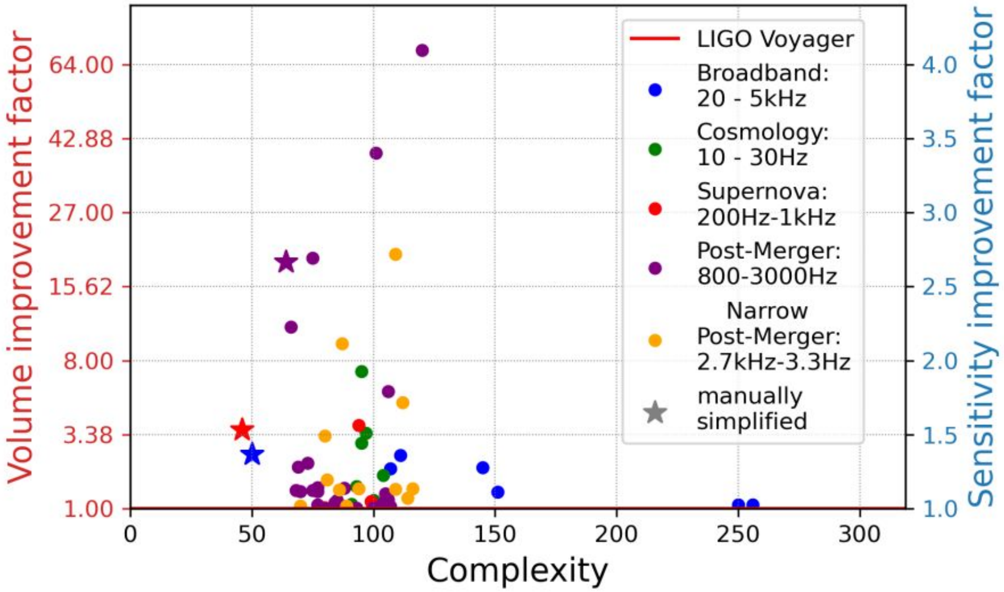



Krenn et al., Phys. Rev. X 15 (2025)

# Gravitational Wave Detector Zoo 2.0

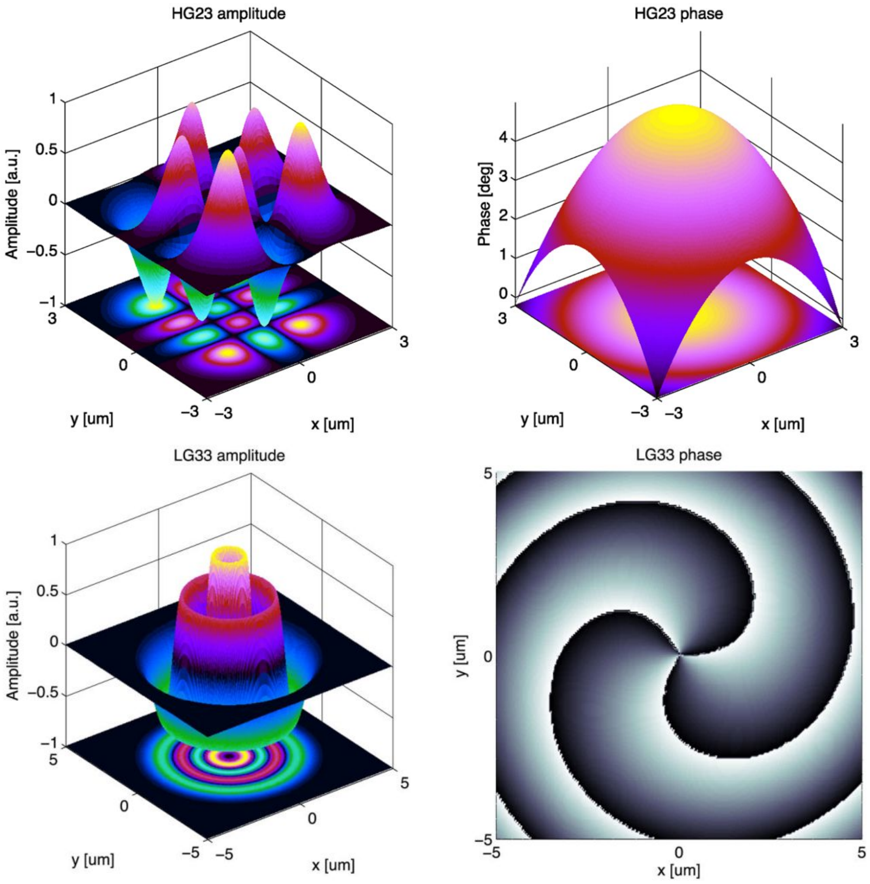
? ? ?

What are your thoughts on computational design?



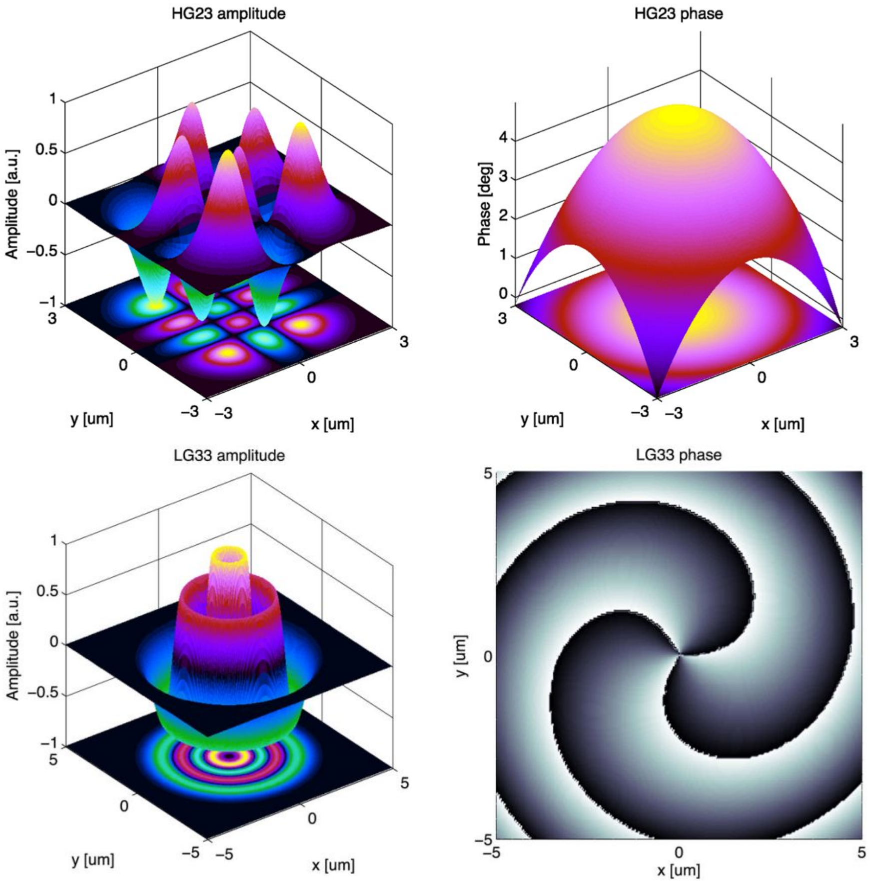

Krenn et al., Phys. Rev. X 15 (2025)

# New Features - e.g. Higher-Order Modes




Bond, C. et al. Living Rev Relativ 19, 3 (2016).

# New Features - e.g. Higher-Order Modes




Could Differometor help in some of your projects?

Which features should we implement next?

Bond, C. et al. Living Rev Relativ 19, 3 (2016).

# pip install differometer - try it out!

https://github.com/artificial-scientist-lab/Differometer

- Differometer is a new differentiable interferometer simulator implemented in JAX
- It enables large speed gains in interferometer optimizations
- We use it for the computational design of new gravitational wave detectors

# pip install differometor - try it out!

https://github.com/artificial-scientist-lab/Differometor

- Differometor is a new differentiable interferometer simulator implemented in JAX
- It enables large speed gains in interferometer optimizations
- We use it for the computational design of new gravitational wave detectors

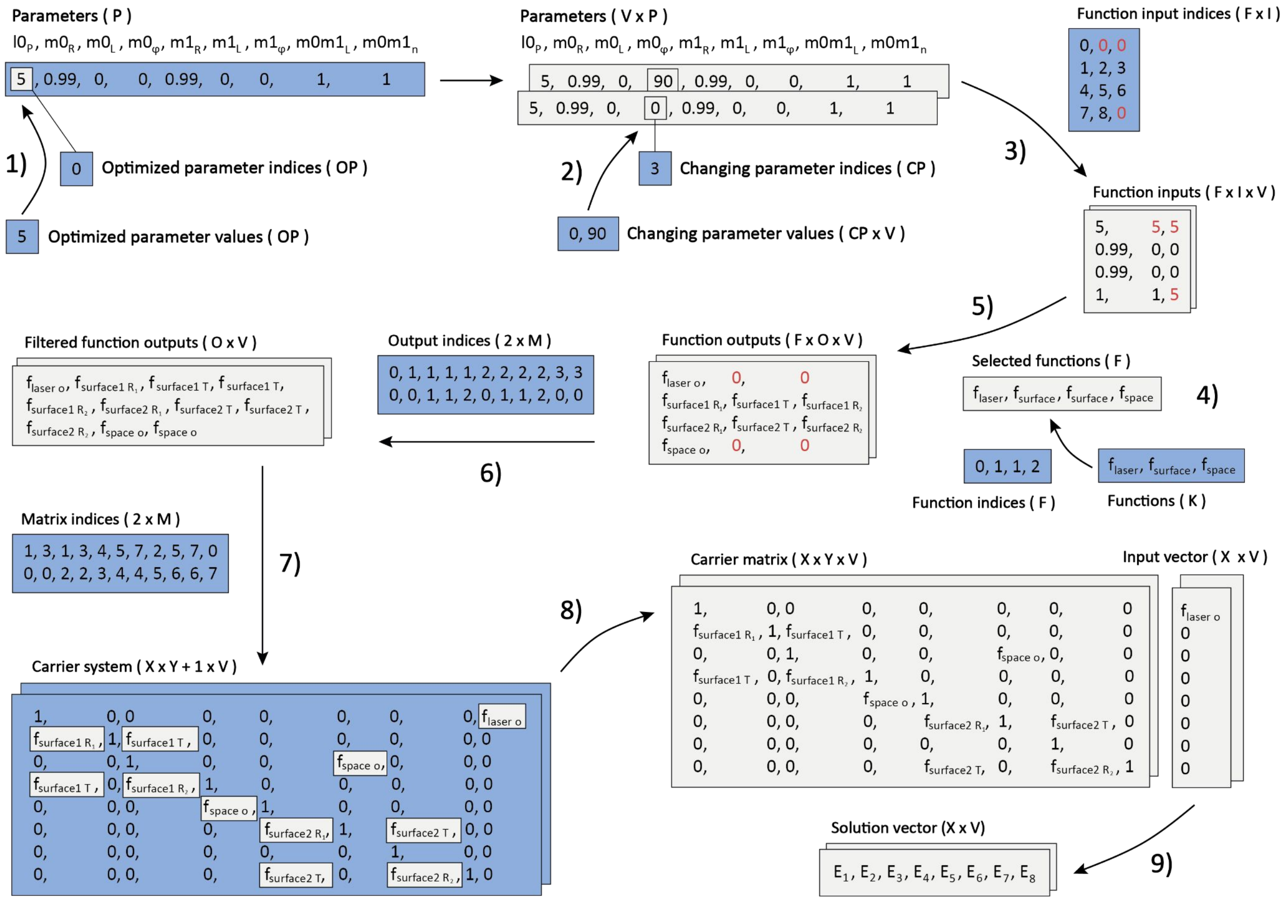
**Thank you for listening!**
**Any questions?**

# Backup

# Lines of Code Comparison

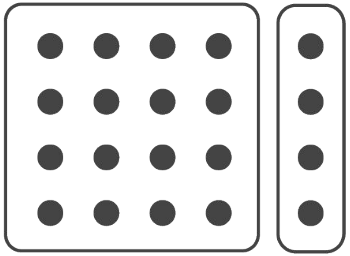
| | Finesse 3 | Differometor |
|---|---|---|
| Files | 269 | 8 |
| Blank | 14794 | 450 |
| Comment | 21736 | 646 |
| Code | 49152 | 2123 |

# Carrier System

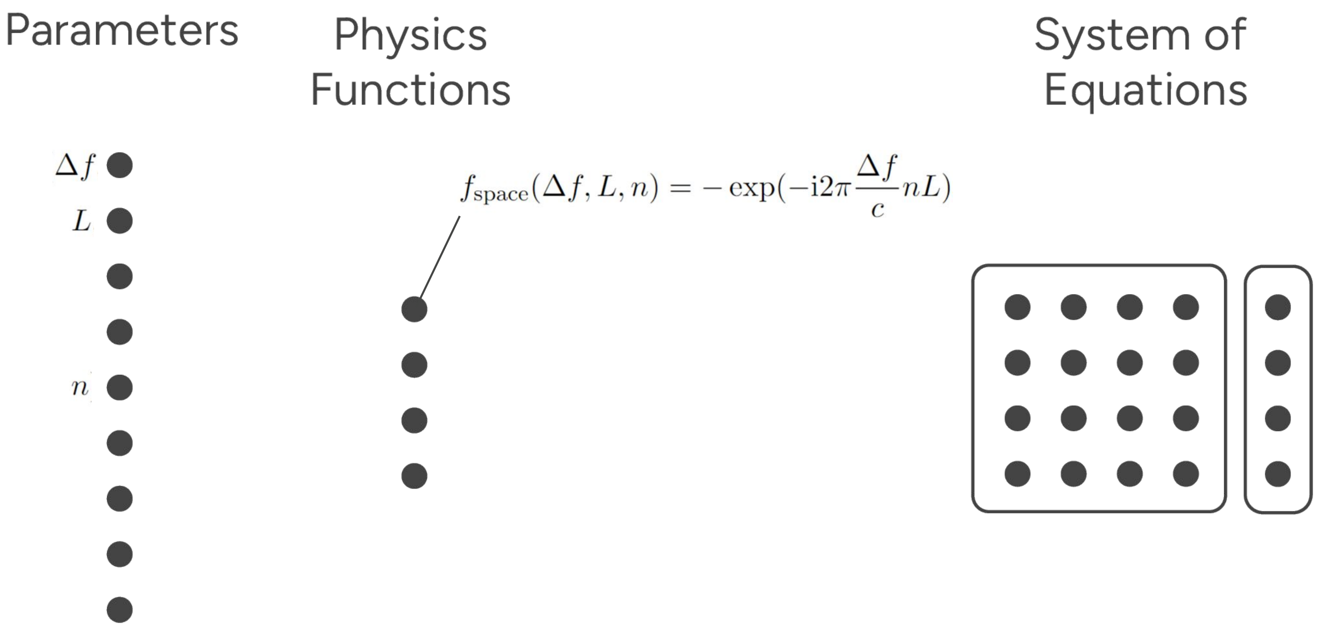


# The Simulator Engine

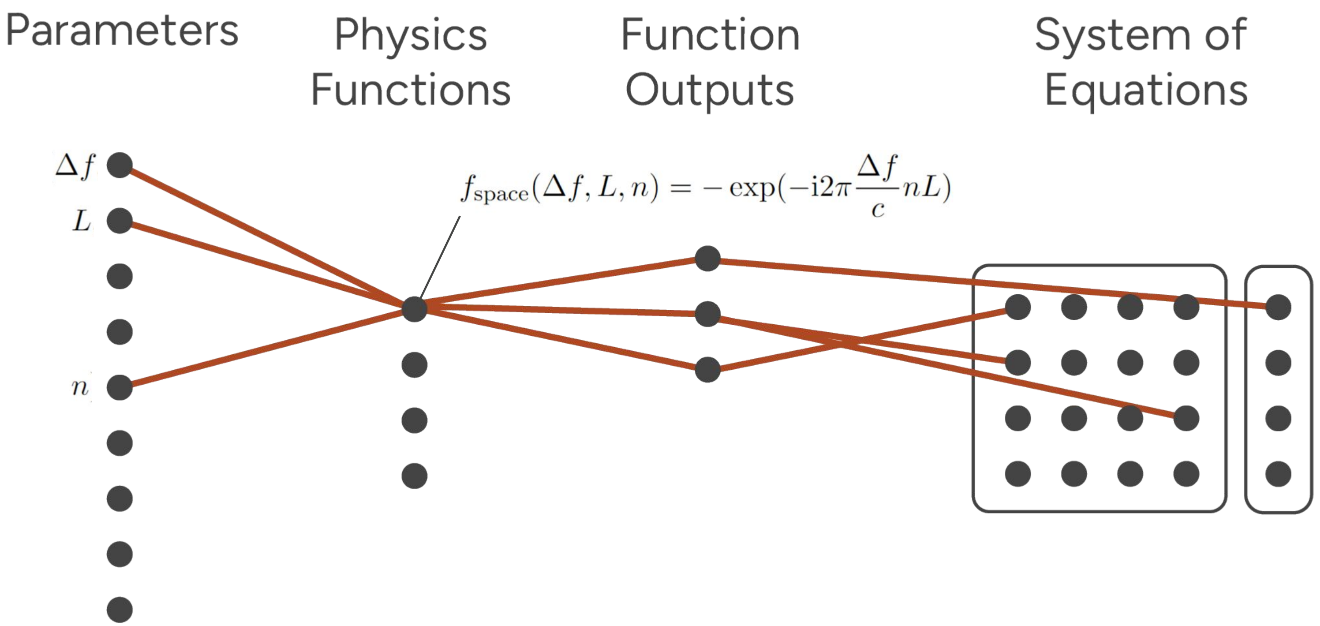
System of Equations



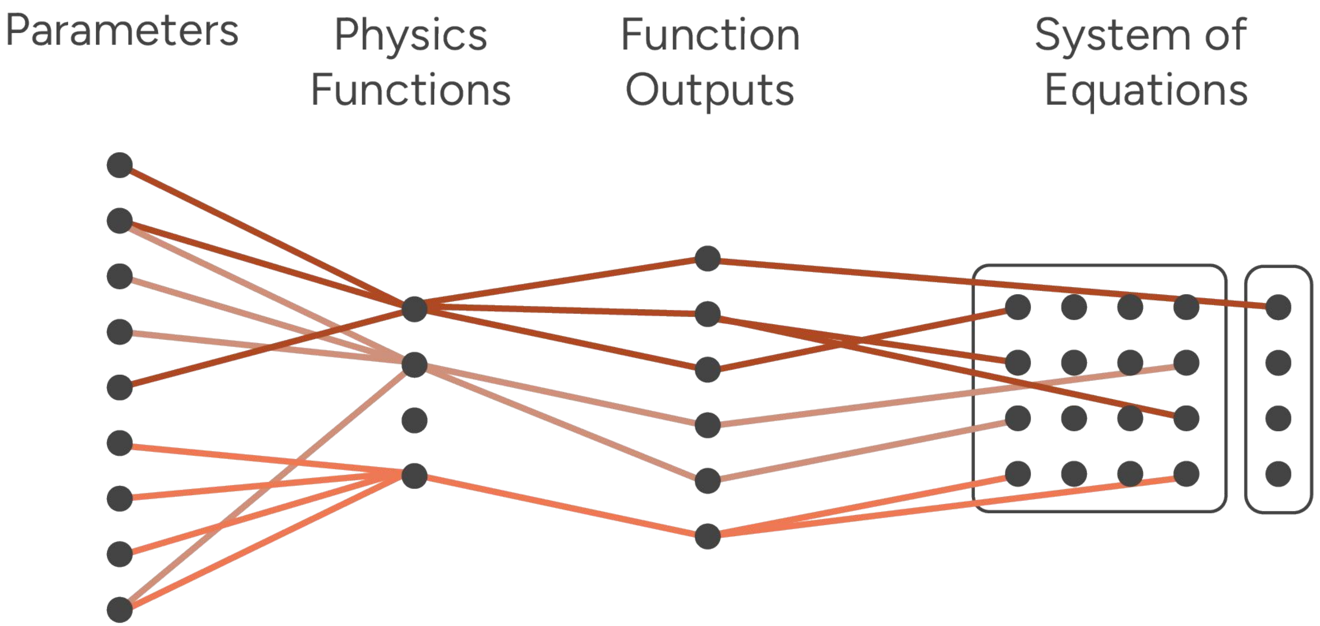
# The Simulator Engine



# The Simulator Engine



# The Simulator Engine



# Differometor - Subsystems

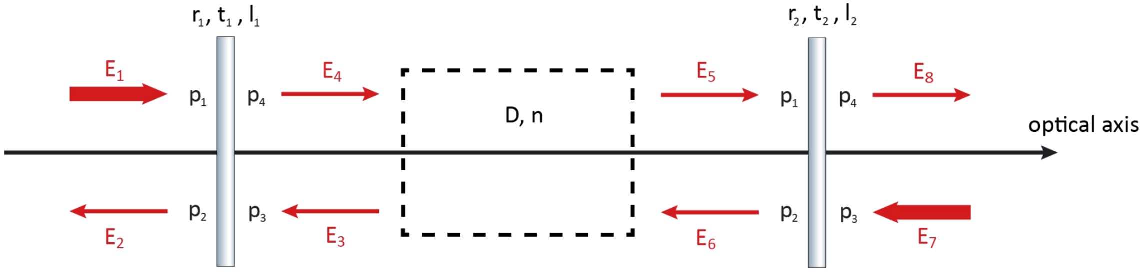
**Plane Wave Propagation**

Signal Sidebands

Quantum Noise

Optomechanics

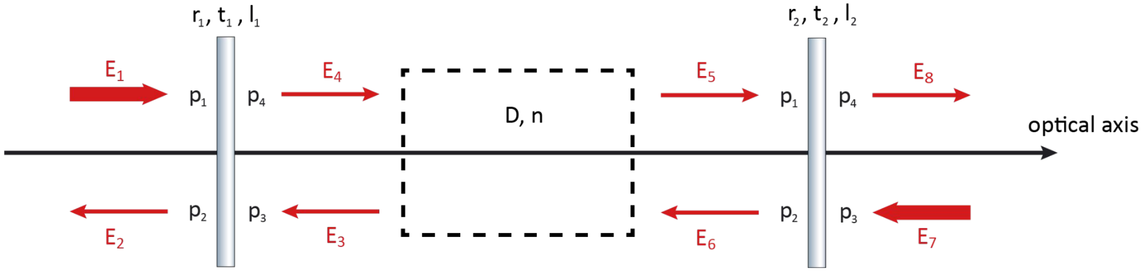
# Plane Wave Propagation




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

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

# Plane Wave Propagation




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

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

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

# Differometor - Subsystems

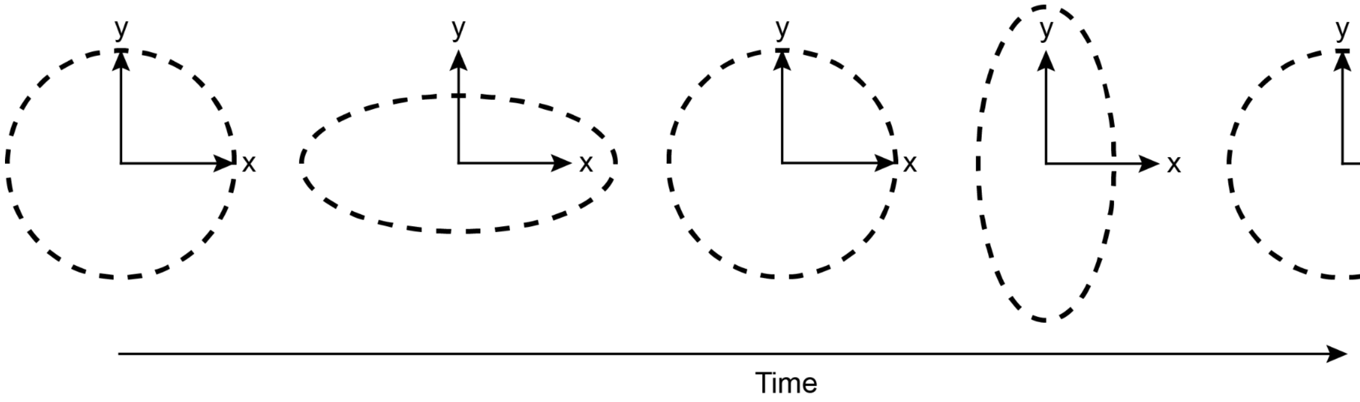
Plane Wave Propagation

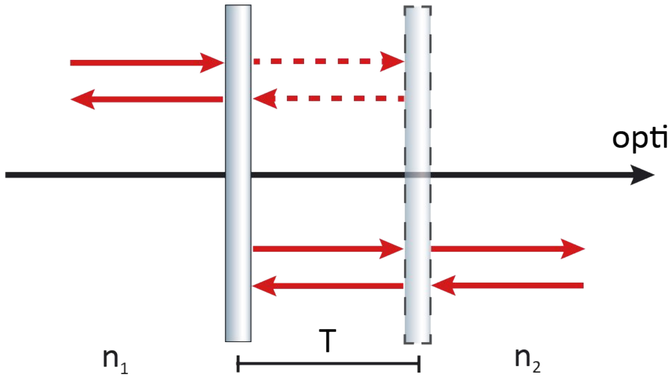
**Signal Sidebands**

Quantum Noise

Optomechanics

# The Signal System



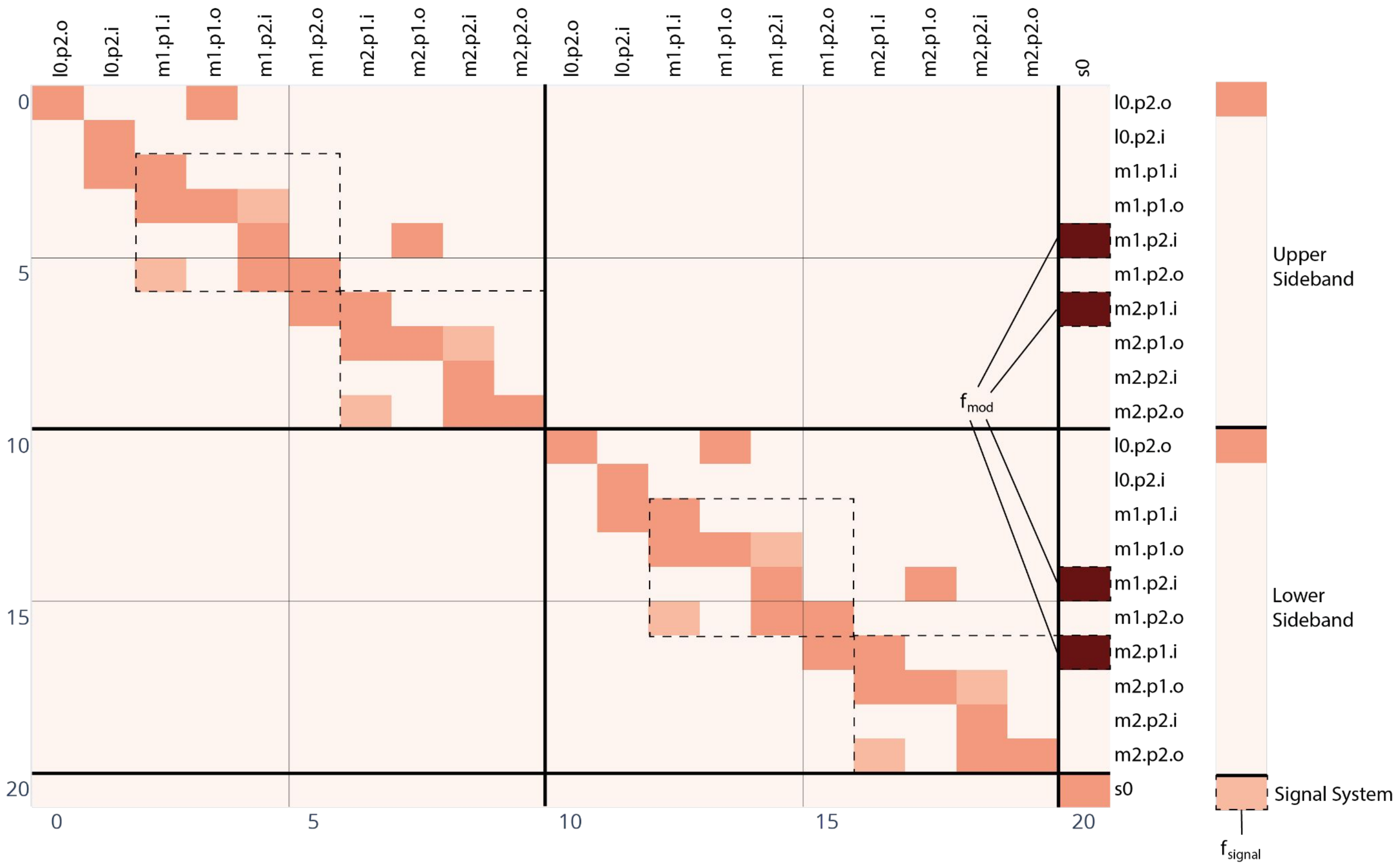




$$E = E_0 \exp(\mathrm{i}(\omega_0 t + \varphi_0 + \phi(t)))$$

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

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

# The Signal System

# Differometor - Subsystems

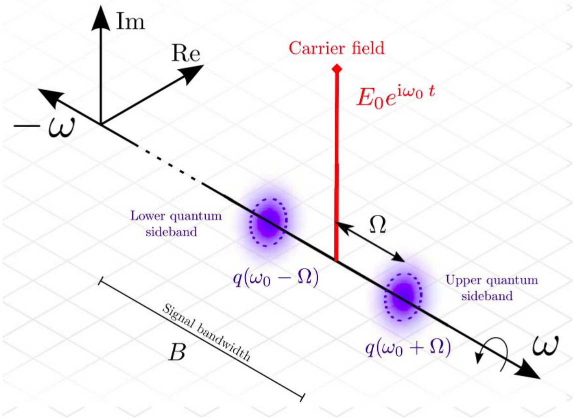
Plane Wave Propagation

Signal Sidebands

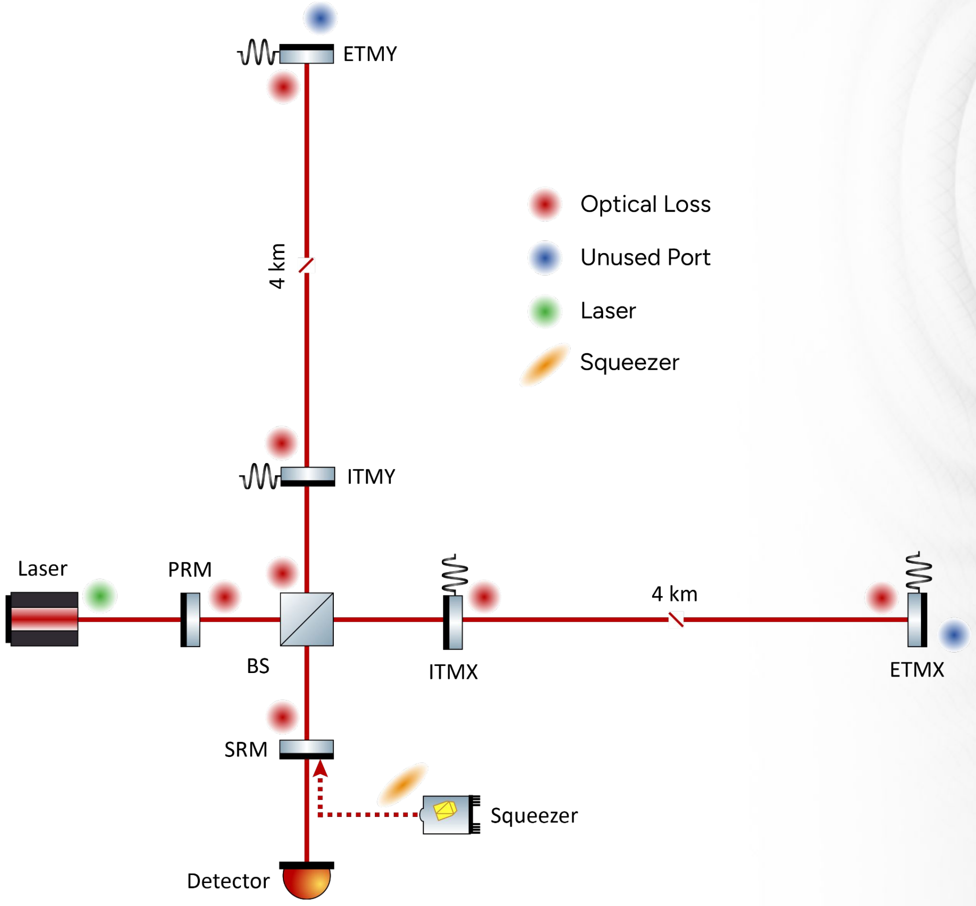
**Quantum Noise**

Optomechanics

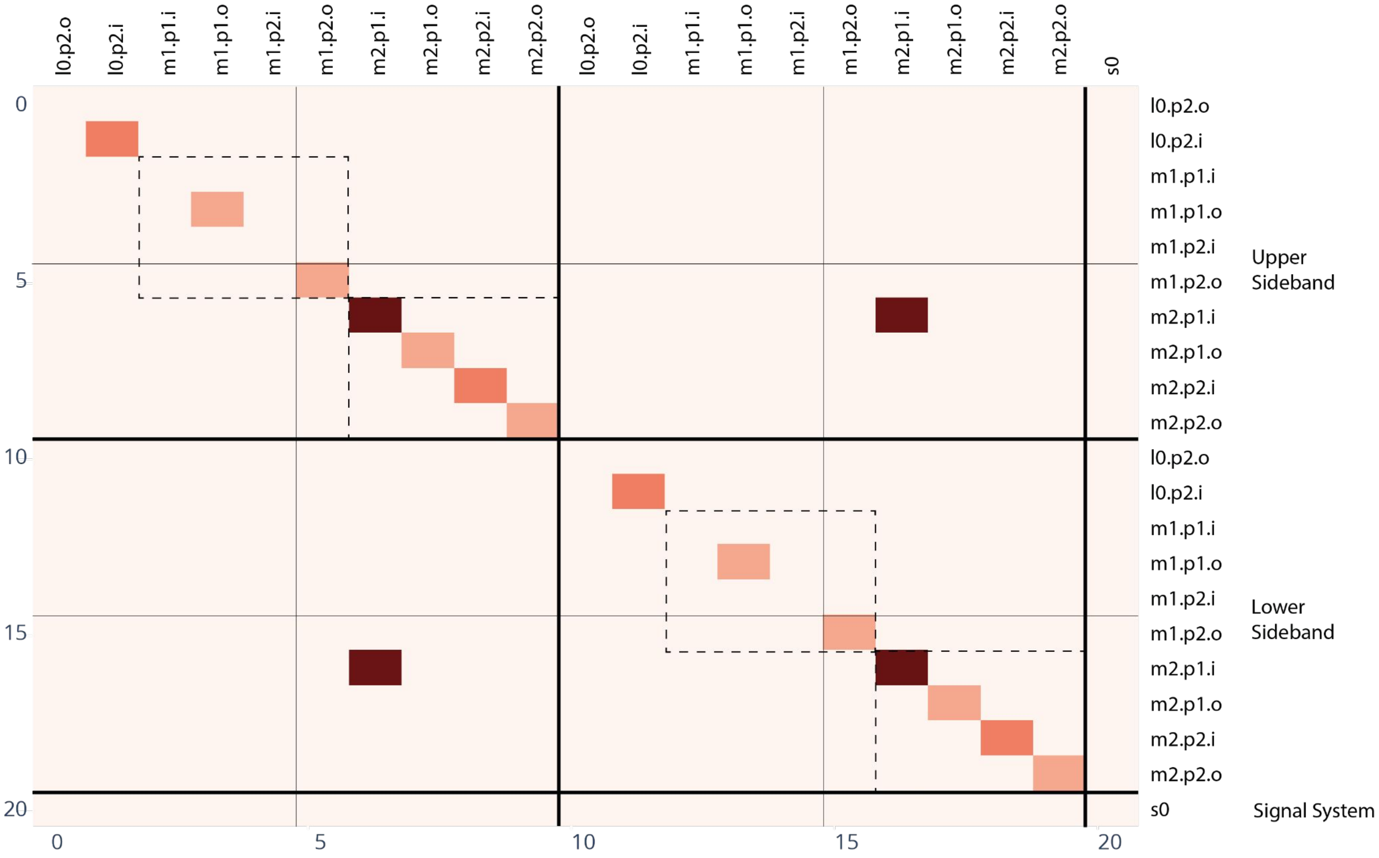
# Quantum Noise




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



# Quantum Noise

# Differometor - Subsystems

Plane Wave Propagation

Signal Sidebands

Quantum Noise

**Optomechanics**

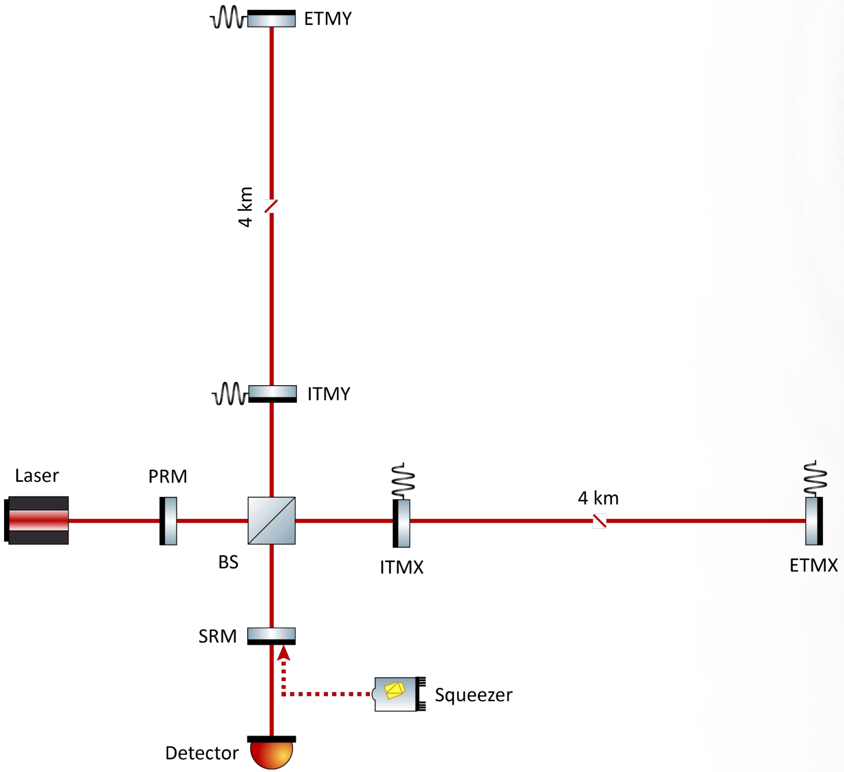
# Optomechanics

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

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

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

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



# Optomechanics