Finesse 3

the new generation

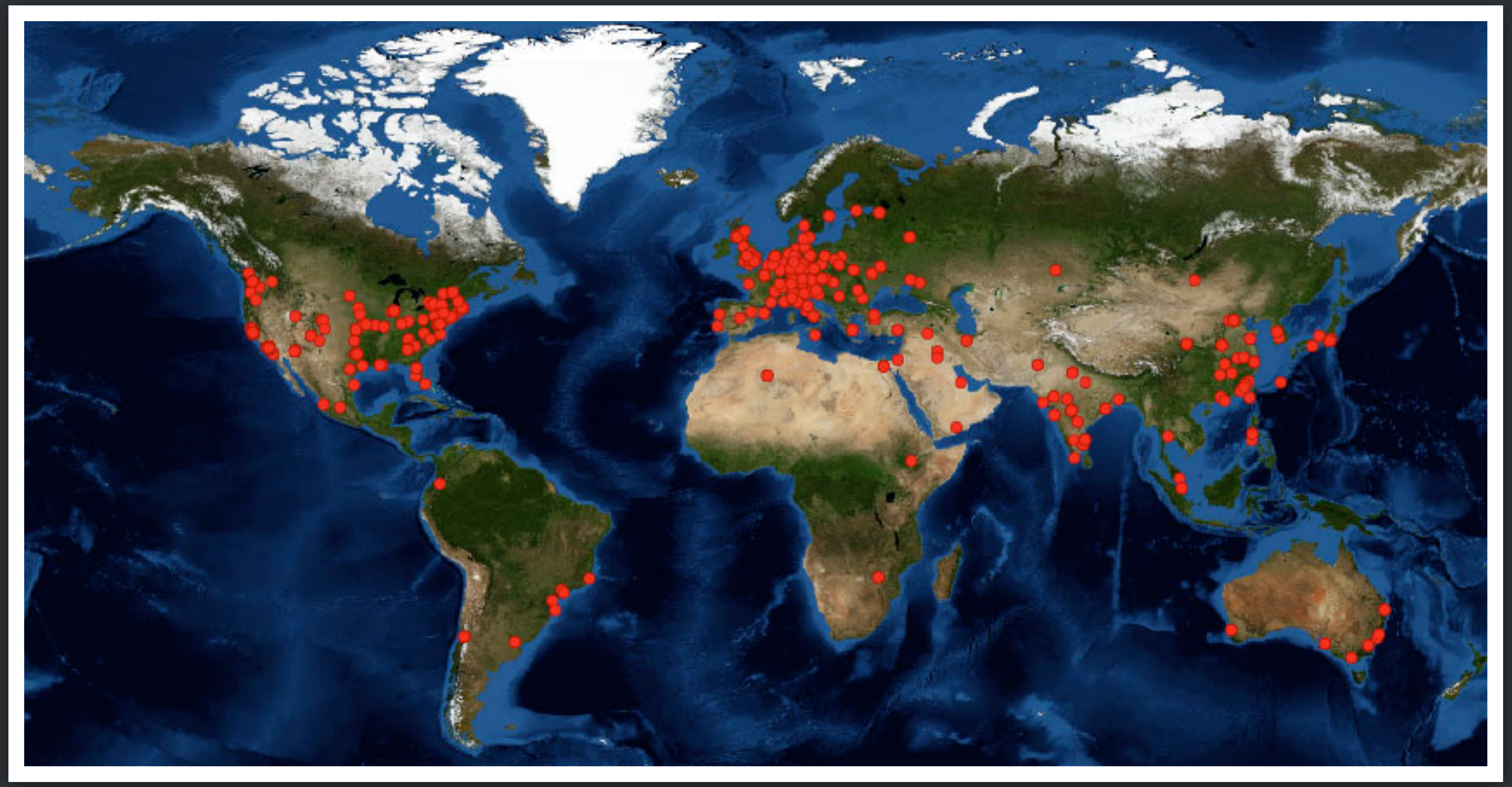
a short introduction to our latest version of FINESSE

Daniel Brown, Andreas Freise for the Finesse team, 05.06.2023

Nikhef VU VRIJE UNIVERSITEIT AMSTERDAM

Long history short

- Started1997, PhD side project
- Used extensively worldwide
- Open sourced in 2012
- Continuously used and developed



What can FINESSE do?

FINESSE can simulate:

- Beam shapes
- Optical losses
- Quantum noise
- Squeezing
- Radiation pressure effects
- Diverse detectors
- Error signals
- Transfer functions

… so long as the model is frequency domain (i.e. static or quasi-static), paraxial, and suits modelling using a modal basis.

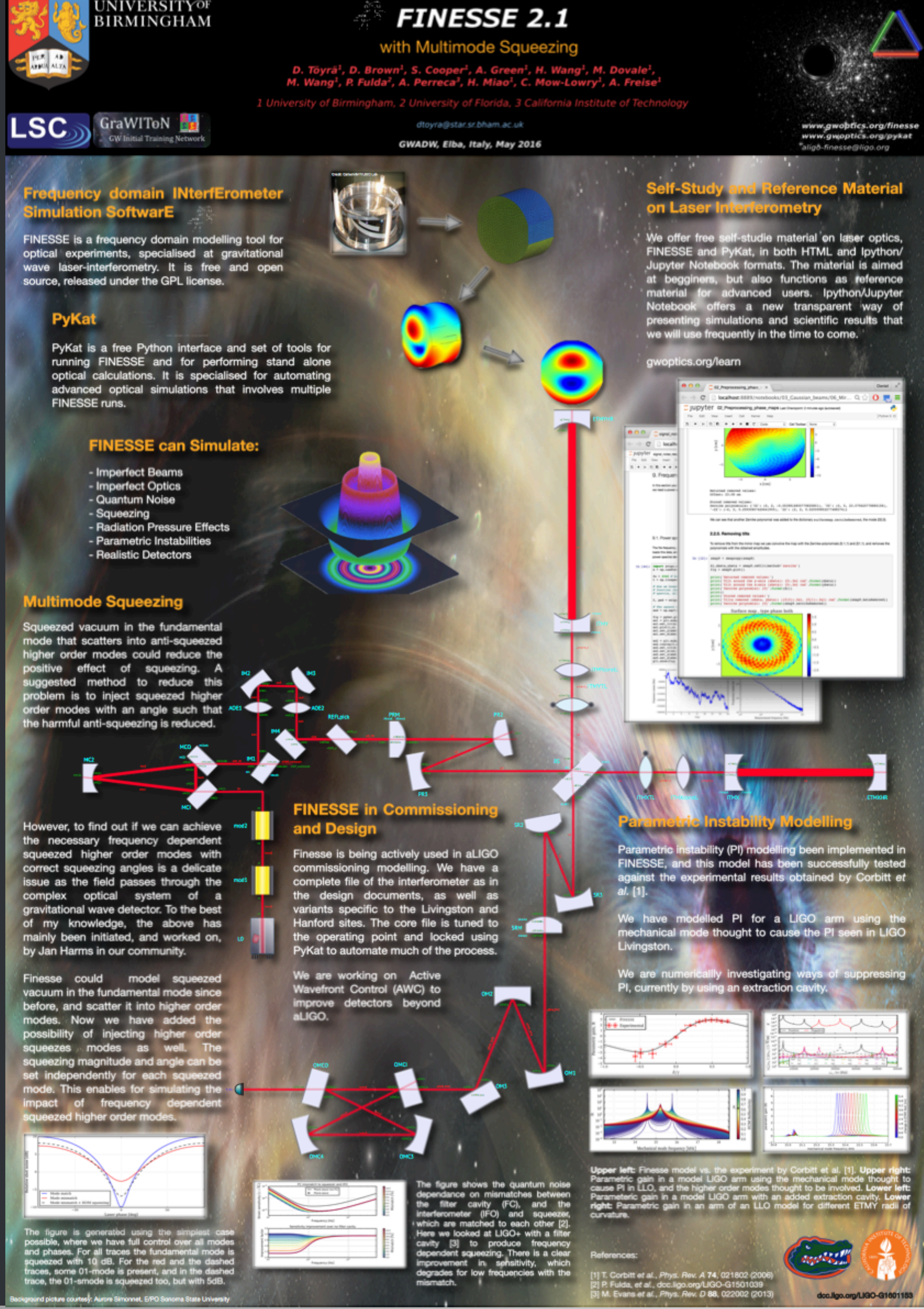
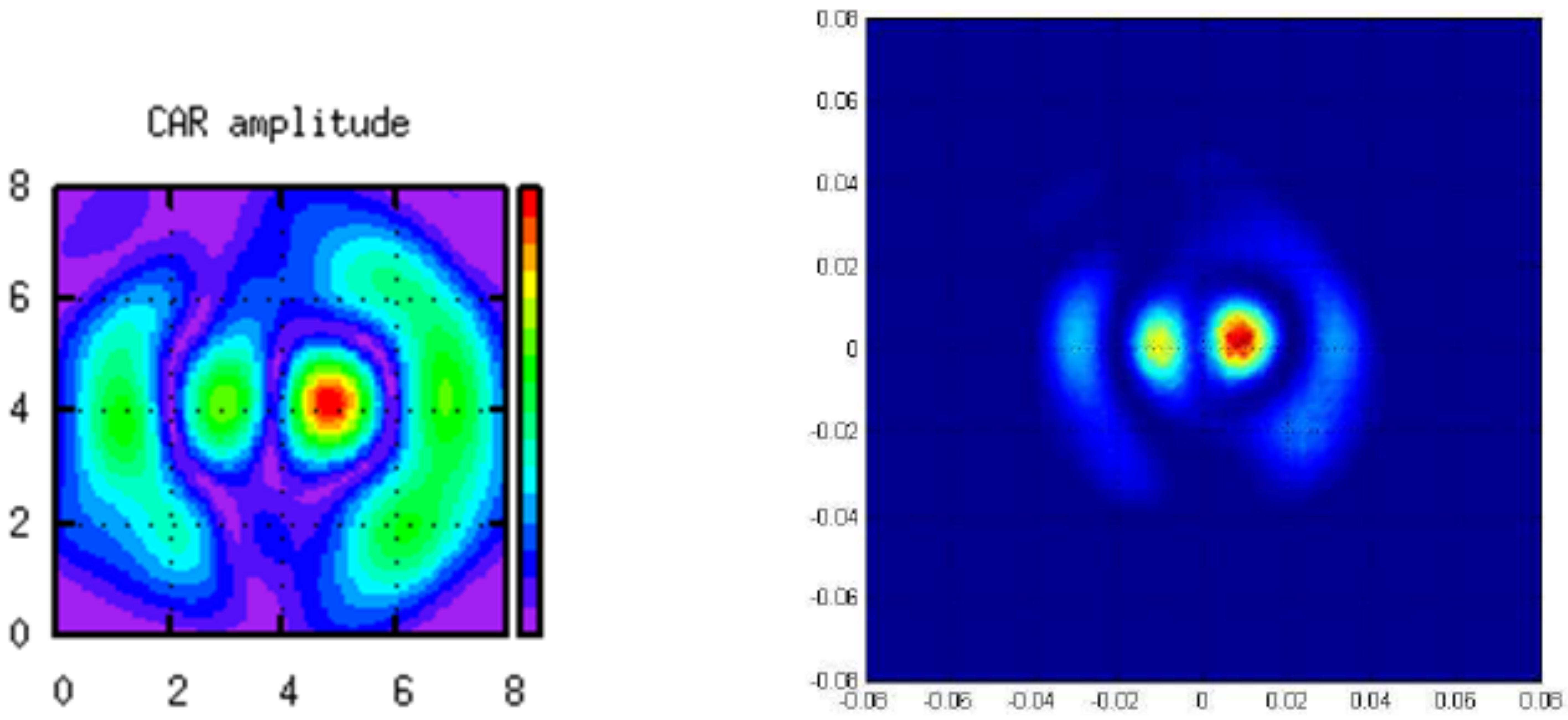


Image: 'Finesse 2.1' poster by D. Töyrä, LIGO-G1601153

Using FINESSE: match simulation to **defective** interferometers

The FINESSE team always spend significant effort to model imperfect interferometers. This was not yet crucial for the first generation of detectors, but has become essential for current and future instruments. Many FINESSE features, conventions and habits have been designed with this in mind.

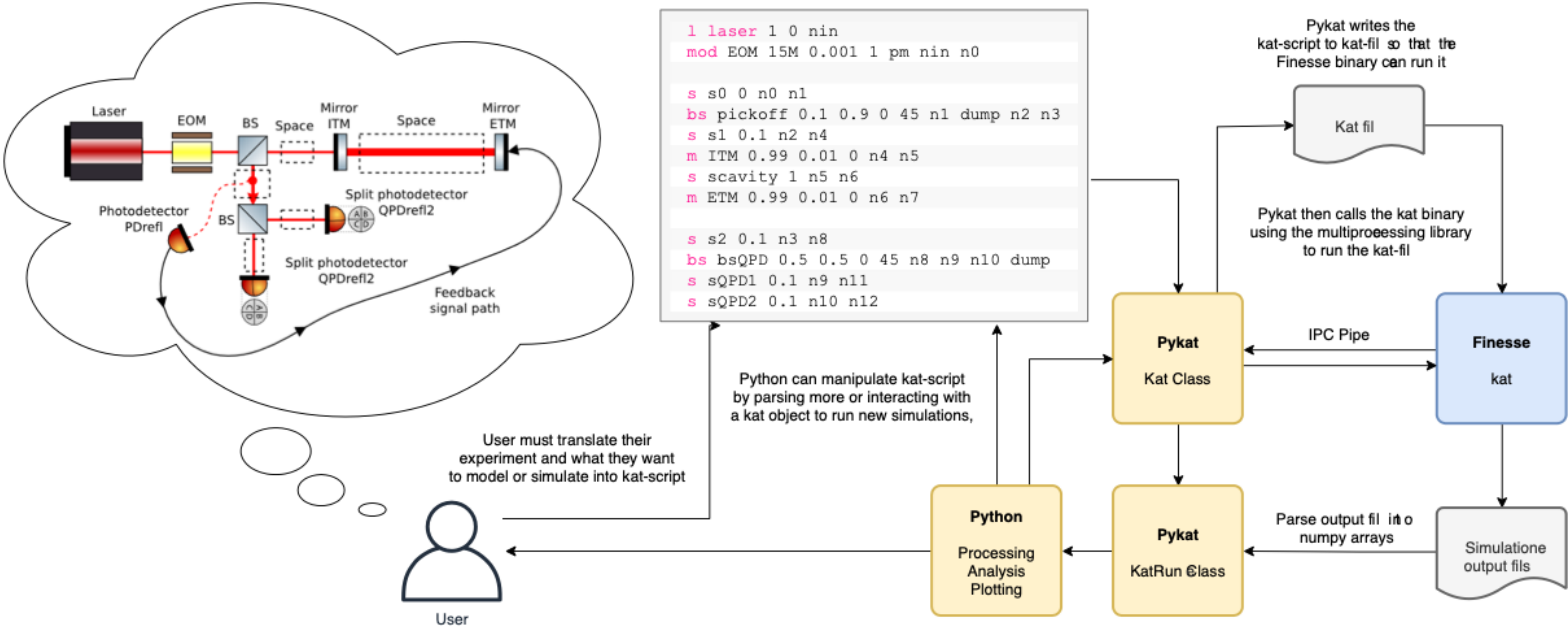


Modal model (Finesse) **FFT propagation model (DarkF)**

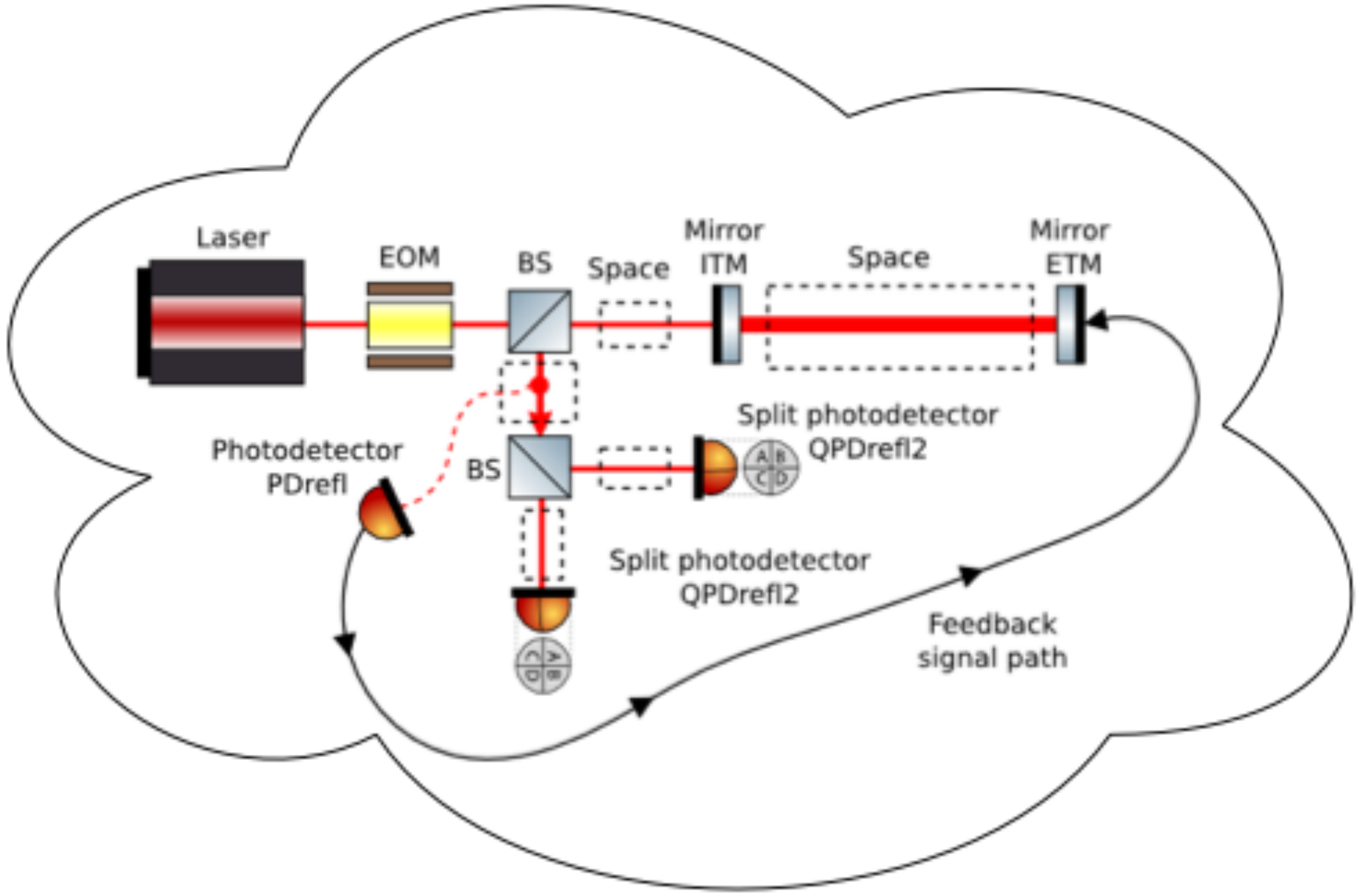
[J. Marque et al 2009]

FINESSE 2 + Pykat

e



e




```
l laser 1 0 nin
mod EOM 15M 0.001 1 pm nin n0

s s0 0 n0 n1
bs pickoff 0.1 0.9 0 45 n1 dump n2 n3
s s1 0.1 n2 n4
m ITM 0.99 0.01 0 n4 n5
s scavity 1 n5 n6
m ETM 0.99 0.01 0 n6 n7

s s2 0.1 n3 n8
bs bsQPD 0.5 0.5 0 45 n8 n9 n10 dump
s sQPD1 0.1 n9 n11
s sQPD2 0.1 n10 n12
```

Python can manipulate kat-script by parsing more or interacting with a kat object to run new simulations,

User must translate their experiment and what they want to model or simulate into kat-script

User

Python

Processing
Analysis
Plotting

Finesse 3

Combines both Finesse 2 and Pykat for efficiency and performance

We are now encouraging the use of FINESSE 3 over previous versions!

The new FINESSE manual is not yet fully complete but is constantly evolving on the web page.

It contains many examples and documentation for various functions and commands.

If a part of the manual is not yet complete and you really need it, ask in the chat channel and we will aim to fill it in faster.

Finesse

3.0a8-5-g3aee1c2d

Search docs

- Introduction to Finesse
- Getting started
- **Examples**
  - An optical cavity
  - Pound-Drever-Hall locking
  - Near unstable cavities
  - Extracting geometrical parameters
  - Radio-frequency modulation and Bessel functions
  - Radiation pressure
  - Squeezing and homodyne detection
  - Optical springs
  - Quantum-noise limited sensitivity of Advanced LIGO
  - Aligning and translating a beam with steering mirrors
  - Cavity eigenmodes
  - Locking actions
  - Frequency dependant squeezing
  - Defining cavity eigenmodes with the cavity command
- Using Finesse
- Physics of Finesse

/ Getting started / Installing Finesse / First Install

First Install

This section provides information on getting started with your first installation of FINESSE. Although other installation methods are available, we recommend following this guide to set up your initial environment to avoid known issues.

Python Installation

We recommend using Miniconda to install the Python ecosystem. We have created Conda packages to automate and easily install FINESSE in one command.

From the section titles, pick the one that suits your condition.

I am a Python user and have Anaconda/Conda installed already (Windows, OSX, or Linux)

I am a Python user and have my own environment already set up that I want to use

For Windows users without Conda installed

For MacOS users without Conda installed

For Linux users without Conda installed

I am a Python user and have Anaconda/Conda installed already (Windows, OSX, or Linux)

If you already use Conda on your system, then installation is very easy! You can proceed to Installing Finesse and Jupyter with Conda.

Python vs KatScript

- FINESSE 2 only worked with KatScript (which limited what could be done).
- PyKat added a Python wrapper which essentially wrote KatScript for you.
- FINESSE 3 is firstly a Python program, it has a Python interface for everything. KatScript is a wrapper around the full Python programming interface.
- This means that you can use both to make models and run simulations. KatScript is often easier or more compact, Python is more powerful.
- Some features are not (yet) supported in KatScript and can only be used through Python (e.g surface maps).

KatScript

```
model = finesse.script.parse("""
laser l1 P=1
space s1 l1.p1 m1.p2
mirror m1 R=0.5 T=0.5
power_detector_dc P m1.p2.o
""")
```

Python

```
# or you can use a Python interface...
from finesse.components import Laser, Space, Mirror
from finesse.detectors import PowerDetector

model_python = finesse.Model()
model_python.add(Laser('l1', P=1))
model_python.add(Mirror('m1', R=0.5, T=0.5))
model_python.add(
    PowerDetector(
        'P',
        model_python.m1.p2.o
    )
)
model_python.add(Space(
    's1',
    model_python.l1.p1,
    model_python.m1.p1
  )
)
```

Nikhef VU VRIJE UNIVERSITEIT AMSTERDAM

Changes to KatScript

Finesse 2

```
m2 PRAR 0 $L_PRAR 0 nPR1 nPRsub1
s sPRsub 0.1003 $nsilica nPRsub1 nPRsub2
m1 PR $T_PR $L_PR 0 nPRsub2 nPR2
#attr PR Rc -1477         # Measured cold IFO PR RoC [VIR-0029A-15]
attr PR Rc -1430          # Design value to have good matching (sho
attr PRAR Rc -3.62         # Measured PR AR RoC [VIR-0029A-15]

# Space between PR and POP. Length from TDR.
s lPR_POP 0.06 1 nPR2 nPOP1

# Pick off plate. The angle of incidence and the physical distanc
# propagates inside POP are computed from thickness of 3.5 cm [TD
# tilt [TDR], and refractive index of $nsilica. POP AR is wedged,
# the AR-reflectivity is set as a loss.
bs2 POP_AR 0 $L_POP2 0 6.0 nPOP1 nPOPunused1 nPOPsub1 nPOPsub3
s sPOPsub 0.03549 $nsilica nPOPsub1 nPOPsub2
bs2 POP $R_POP1 0 0 4.135015 nPOPsub2 nPOPsub4 nPOP2 nB4
s sB4_att 0 nB4 nB4att
bs B4_attenuator 0.7344 0.2656 0 0 nB4att dump nB4b dump
```

Finesse 3

```
m PRAR R=0.0 L=160u Rc=-3.62
s sPRsub PRAR.p2 PR.p1 L=0.1003 nr=nsilica
m PR T=0.04835 L=30u Rc=-1430.0

# Space between PR and POP. Length from TDR.
s lPR_POP PR.p2 POP_AR.p1 L=0.06

# Pick off plate. The angle of incidence and the physical dista
# propagates inside POP are computed from thickness of 3.5 cm [
# tilt [TDR]. POP AR is wedged, thus one AR-reflectivity is set
# POP reflectivities [VIR-0027A-15]

bs POP_AR R=0.0 L=125u alpha=6.0
s sPOPsub POP_AR.p3 POP.p1 L=0.03549 nr=nsilica
bs POP R=184u L=0.0 alpha=4.135015
# B4' port is POP.p4, attenuated B4 is B4_attenuator.p3

s sB4_att POP.p4 B4_attenuator.p1
bs B4_attenuator R=0.7344 T=0.2656
```

Changes to KatScript

FINESSE 3

- Main syntax style stays the same, i.e. m=mirror, one line per component
- No 'attribute' command any more
- Values are always assigned using the parameter name
- No need to specify nodes explicitly
- Instead spaces connect components directly (using 'ports')

```
m PRAR R=0.0 L=160u Rc=-3.62
s sPRsub PRAR.p2 PR.p1 L=0.1003 nr=nsilica
m PR T=0.04835 L=30u Rc=-1430.0

# Space between PR and POP. Length from TDR.
s lPR_POP PR.p2 POP_AR.p1 L=0.06

# Pick off plate. The angle of incidence and the physical dist
# propagates inside POP are computed from thickness of 3.5 cm
# tilt [TDR]. POP AR is wedged, thus one AR-reflectivity is se
# POP reflectivities [VIR-0027A-15]

bs POP_AR R=0.0 L=125u alpha=6.0
s sPOPsub POP_AR.p3 POP.p1 L=0.03549 nr=nsilica
bs POP R=184u L=0.0 alpha=4.135015
# B4' port is POP.p4, attenuated B4 is B4_attenuator.p3

s sB4_att POP.p4 B4_attenuator.p1
bs B4_attenuator R=0.7344 T=0.2656
```

Changes to KatScript

- You can do math with numbers, variables and references in every command
- New 'degree of freedom' (dof) command
- New `readout' commands to work with `dof' for sensing matrices and noise projection

FINESSE 3

```
# Useful frequencies
################################################################################
var fsrN (0.5 * c0 / LN.L)
var fsrW (0.5 * c0 / LW.L)
var fsrPRC (0.5 * c0 / lPRC)
var fsrSRC (0.5 * c0 / lSRC)
var f1_arm (125.5 * fsrN - 300.0)  # Definition of f1, TDR section 2.3
var f1_SRC (3.5 * fsrN)
var f1_PRC (3.5 * fsrN)

# DOFs
################################################################################
# position
dof DARM NE.dofs.z -1 WE.dofs.z +1
dof CARM NE.dofs.z +1 WE.dofs.z +1
dof MICH NI.dofs.z -1 NE.dofs.z -1 WI.dofs.z +1 WE.dofs.z +1
dof PRCL PR.dofs.z +1
dof SRCL SR.dofs.z -1

# Detectors
################################################################################
readout_dc B1 OMC1_2.p3.o output_detectors=true
readout_dc B2 B2_attenuator.p3.o output_detectors=true
readout_dc B4 B4_attenuator.p3.o output_detectors=true
readout_dc B7 NEAR.p2.o output_detectors=true
readout_dc B8 WEAR.p2.o output_detectors=true
```

Actions

- `Actions' are new Python functions to run FINESSE tasks.
- The actions pre-define all task before they are run. This allows FINESSE 3 to optimise the model (i.e. remove all tuning options that are not required by any action).
- Each action can do either a single simple task or execute a complex task.
- Most users would just use existing actions, but they are easy to write/expand with a bit of Python knowledge.

FINESSE 3

```python
self.model.run(
    Series(
        # Switch off the modulators and remove SR and PR by misaligning them.
        Change(
            {"eom6.midx": 0,"eom8.midx": 0,"eom56.midx": 0,
                "SR.misaligned": True,"PR.misaligned": True,
                "SRAR.misaligned": True,"PRAR.misaligned": True,
            }
        ),
        # Maximize arm power
        Maximize("B7_DC", "NE_z.DC", bounds=[-180, 180], tol=1e-14),
        Maximize("B8_DC", "WE_z.DC", bounds=[-180, 180], tol=1e-14),
        # Minimize dark fringe power
        Minimize("B1_DC", "MICH.DC", bounds=[-180, 180], tol=1e-14),
        # Bring back PR
        Change({"PR.misaligned": False}),
        # Maximize PRC power
        Maximize("CAR_AMP_BS", "PRCL.DC", bounds=[-180, 180], tol=1e-14),
        # Bring in SR
        Change({"SR.misaligned": False}),
        # Maximize SRC power
        # B4_112 requires 56MHz
        Change({"SRCL.DC": 0, "eom56.midx": midx}),
        Maximize("B4_112_mag", "SRCL.DC", bounds=[-180, 180], tol=1e-14),
    ),
)
```

Nodes and ports

Nodes are quite different in FINESSE 3! Each component can have multiple *ports*, each port has multiple *nodes.*

How many and what nodes there are at a port depends on the physical type: *Optical* or *Signal*.

Optical nodes have an input and an output, optical fields can travel in both directions. Optical ports are typically named `pN` where N is the node number, pN.i and pN.o are the input and output optical fields from the component.

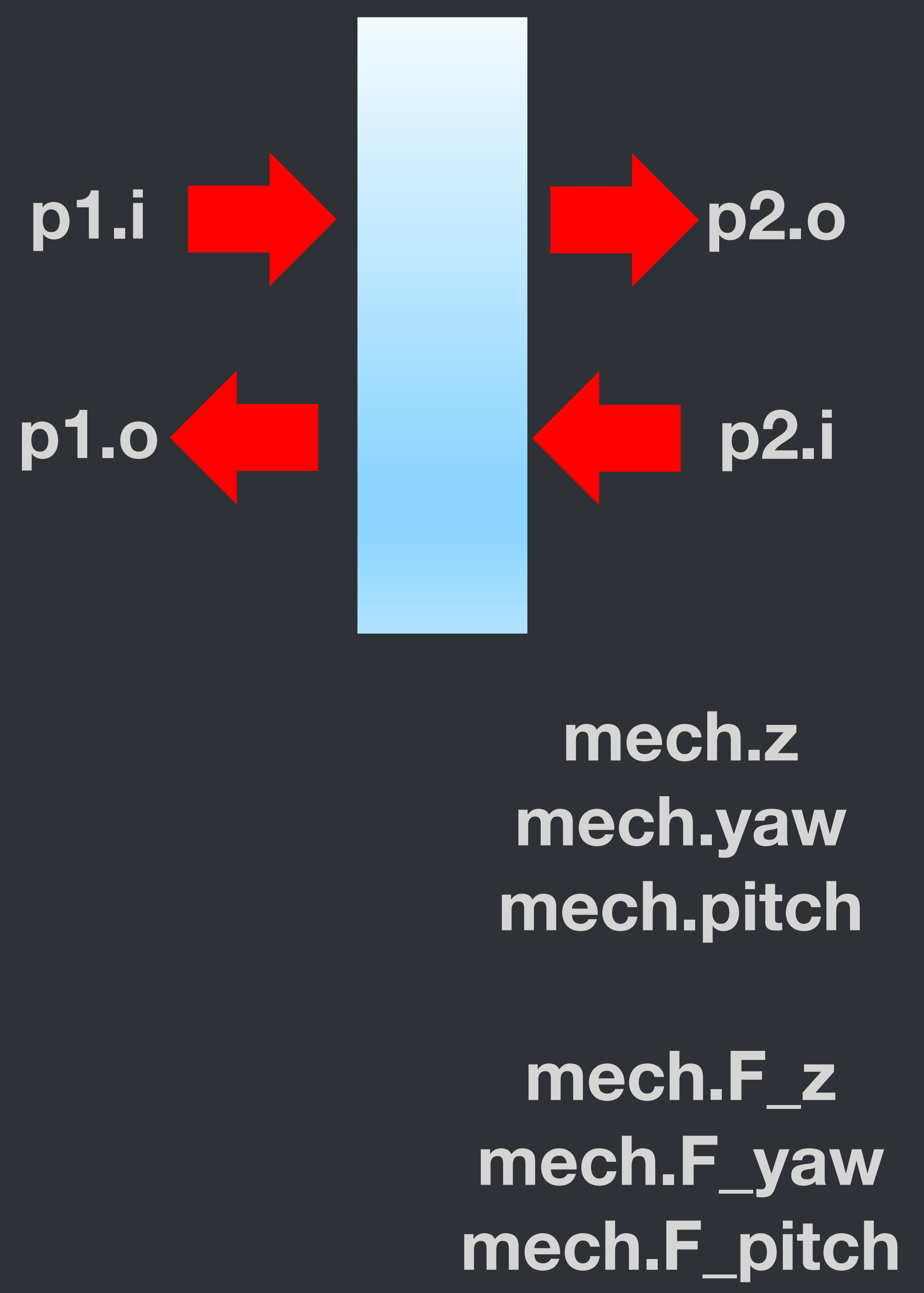
Signal nodes represent electrical or mechanical states in the system (these are where you can inject and read signals from, like a GW signal or a small mirror oscillation).

```
model = finesse.script.parse("""
laser l1 P=1
""")
model.l1.nodes
```

✓ 0.0s

```
OrderedDict([('l1.p1.i', <OpticalNode l1.p1.i @ 0x1678eb8b0>),
             ('l1.p1.o', <OpticalNode l1.p1.o @ 0x1678eb490>),
             ('l1.amp.i', <SignalNode l1.amp.i @ 0x1678eb850>),
             ('l1.phs.i', <SignalNode l1.phs.i @ 0x1678eb940>),
             ('l1.frq.i', <SignalNode l1.frq.i @ 0x1678eb730>),
             ('l1.dx.i', <SignalNode l1.dx.i @ 0x1678eb7c0>),
             ('l1.dy.i', <SignalNode l1.dy.i @ 0x1678eba00>),
             ('l1.yaw.i', <SignalNode l1.yaw.i @ 0x1678eb9d0>),
             ('l1.pitch.i', <SignalNode l1.pitch.i @ 0x1678eb8e0>),
             ('l1.mech.z', <SignalNode l1.mech.z @ 0x1678eba60>),
             ('l1.mech.x', <SignalNode l1.mech.x @ 0x1678eba90>),
             ('l1.mech.y', <SignalNode l1.mech.y @ 0x1678ebbe0>),
             ('l1.mech.yaw', <SignalNode l1.mech.yaw @ 0x1678ebaf0>
             ('l1.mech.pitch', <SignalNode l1.mech.pitch @ 0x1678eb
```

Make more examples



Links to resources

- FINESSE 3 main page:
  https://finesse.ifosim.org/
- FINESSE 3 code repository:
  https://gitlab.com/ifosim/finesse/finesse3
- FINESSE 3 anaconda package:
  https://anaconda.org/conda-forge/finesse
- Chat channel for FINESSE 3:
  https://matrix.to/#/#finesse:matrix.org
- IFOsim logbooks
  https://logbooks.ifosim.org/
- Interferometer techniques for gravitational wave detection
  https://link.springer.com/article/10.1007/s41114-016-0002-8
- GWIC 3G 'Simulations and Control', see
  https://dcc.ligo.org/LIGO-G1800565
- IFOsim mailing list:
  https://grouper.ligo.org/mailinglists/ifosim