Finesse

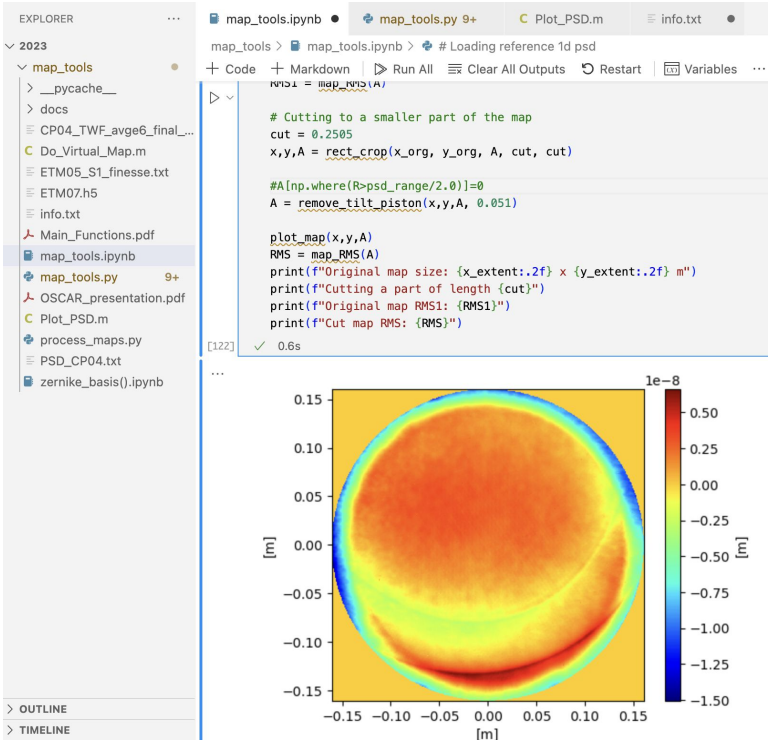
Miron van der Kolk

Who am I?

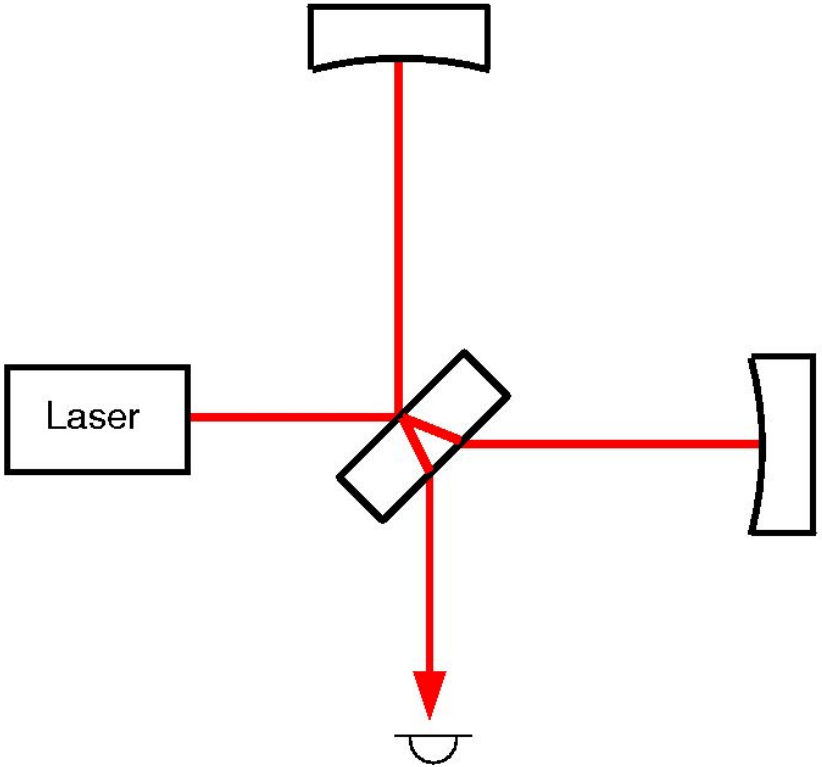
- ‘Scientific programmer’, full-time Finesse maintainer since September
- Take care of the non-science stuff:
  - Usability
  - Bug fixing
  - Releases
  - Code quality and structure
- Please contact me for any questions/issues/suggestions
- Don’t understand all the science (yet)

What is Finesse?

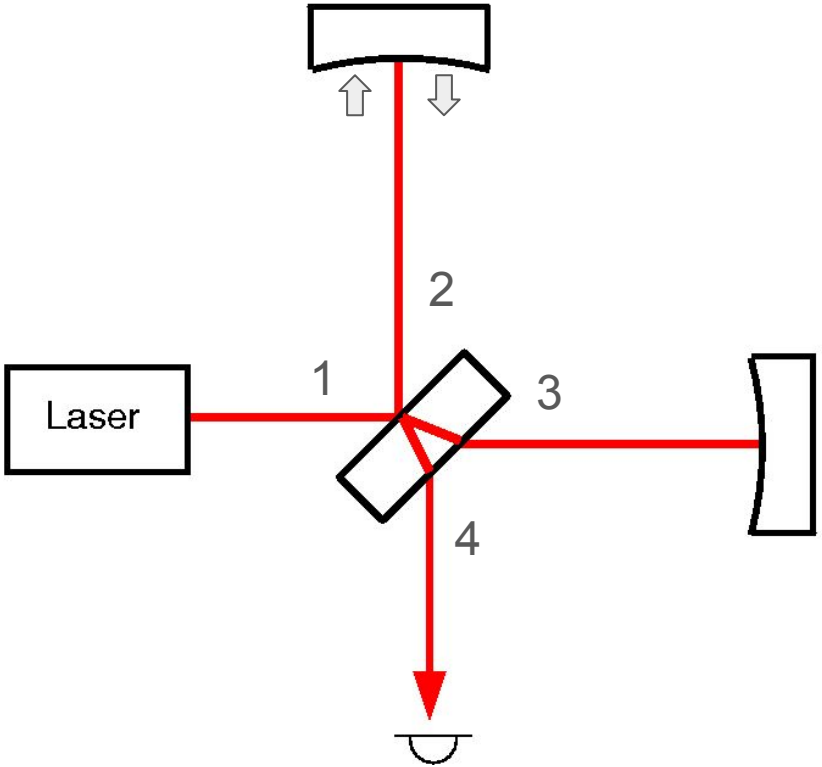
- **F**requency domain **in**terf**e**rometer **s**imulation **s**oftwar**e**
- We simulate interferometers in Python (Cython for speed)
- Use jupyter environments for iterative/interactive investigation



Interferometer as graph



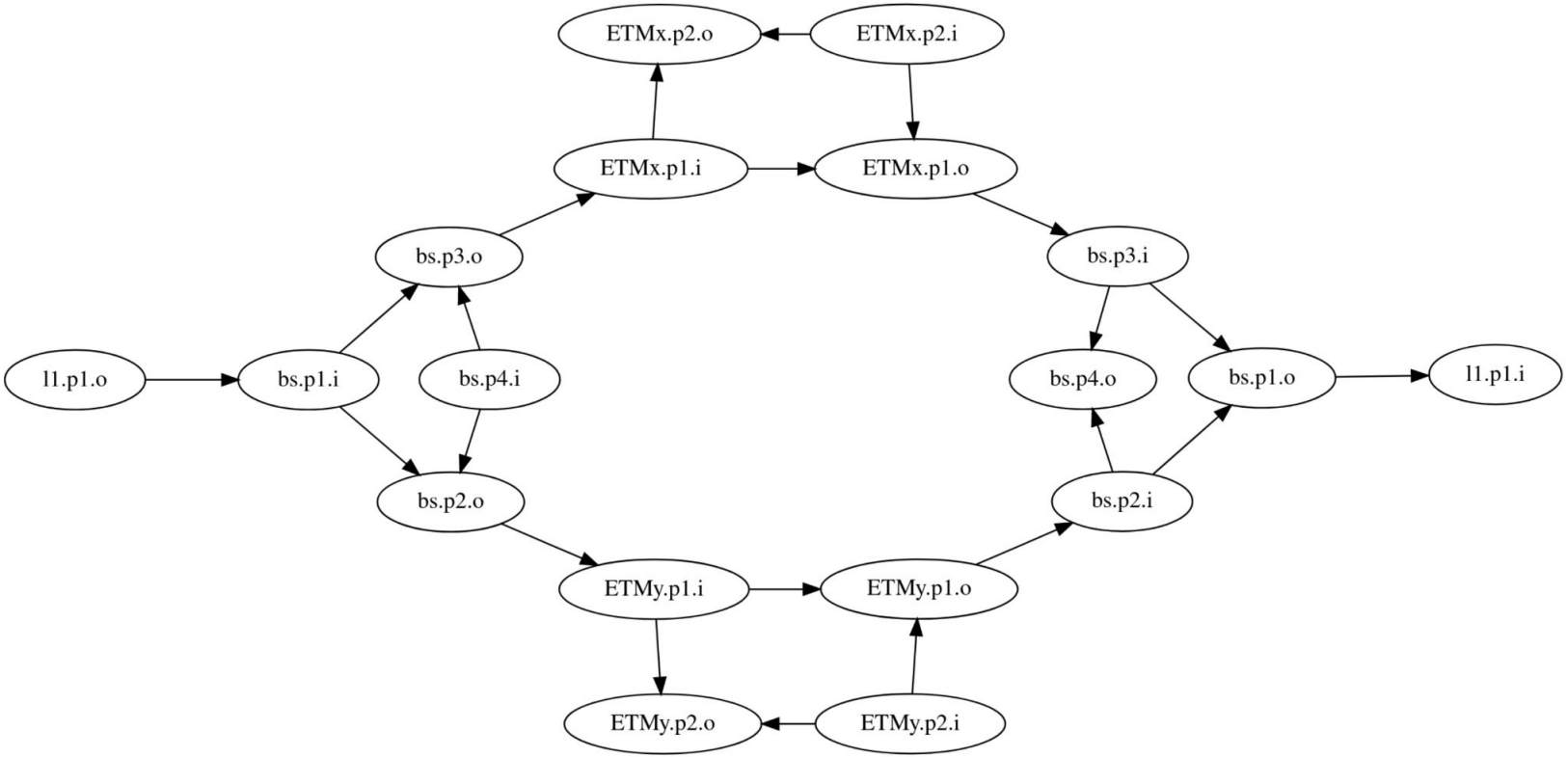
Interferometer as graph



Interferometer as graph

- Nodes/Ports: Optical, Signal or Mechanical
- Edges/Spaces: Matrices transforming the light vector
- Place detectors wherever you want!

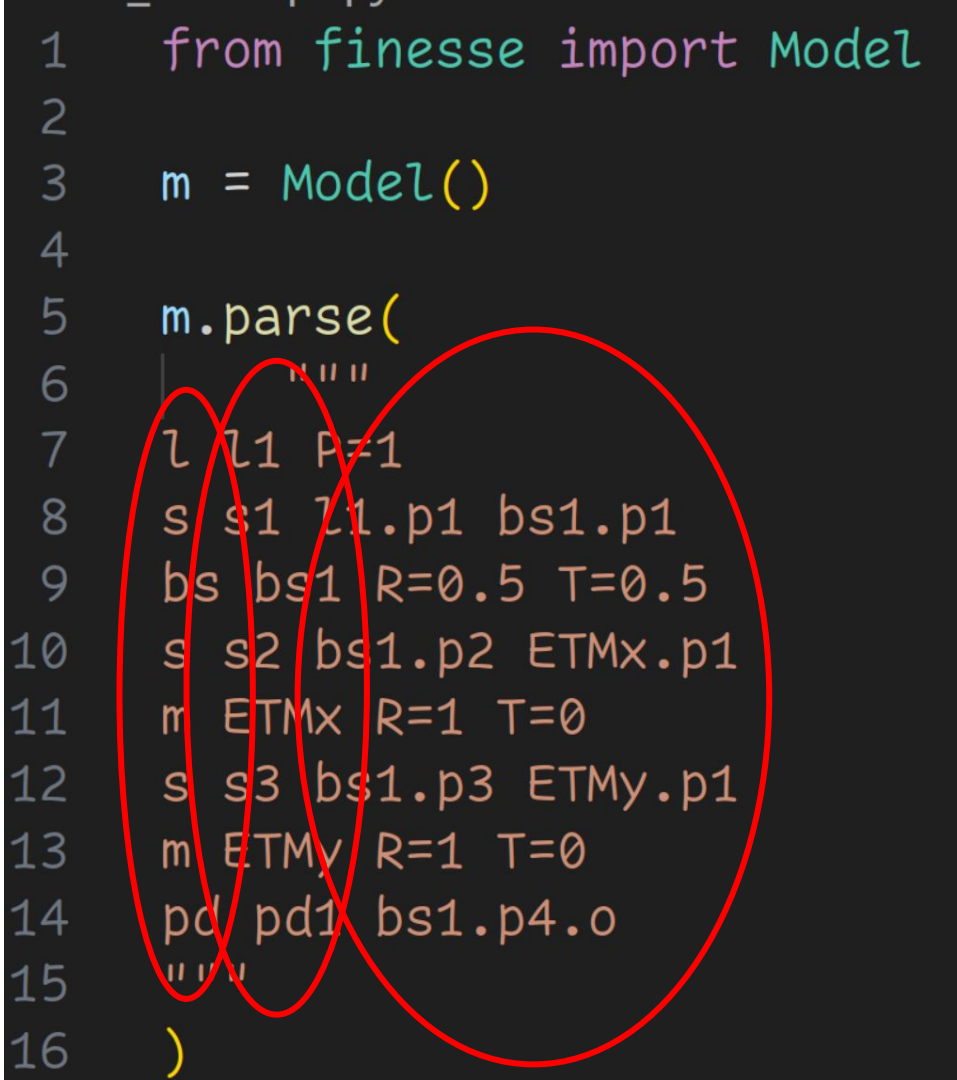
Interferometer as graph



Katscript

- Component type
- Component name
- Component specification

- Declarative (order-independent)
- Edges are spaces
- Convenience method exist



Python Alternative

- Comfortable with programming
- More advanced features

```
from finesse import Model
from finesse.components import (Laser,
                                Beamsplitter,
                                Mirror)
from finesse.detectors import PowerDetector

m = Model()

l1 = Laser("l1")
bs = Beamsplitter("bs", R=0.5, T=0.5)
m.link(l1, bs)

ETMx = Mirror("ETMx", R=1, T=0)
m.link(bs.p3, ETMx)

ETMy = Mirror("ETMy", R=1, T=0)
m.link(bs.p2, ETMy)

pd1 = PowerDetector("pd1", m.bs.p4.o)
```

Detectors

- PowerDetector: physical power
- AmplitudeDetector: amplitude and phase of the light
- CCD: beam shape
- MathDetector: outputs result of symbolic equation

Actions: making solutions

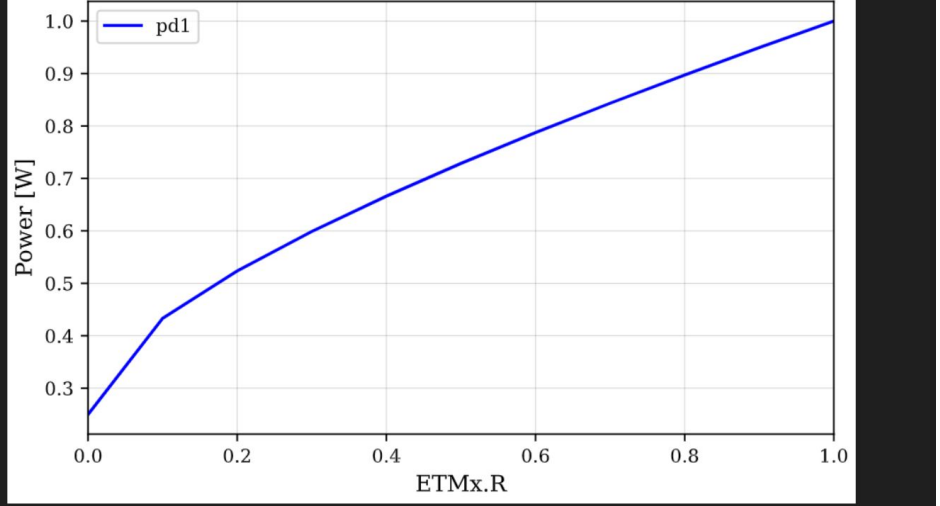
- Xaxis: most basic
- Modify model and calculate values at detectors
- FrequencyResponse, Beam Tracing, Minimize

```
from finesse.analysis.actions import Xaxis

sol = m.run(Xaxis(m.ETMx.R, "lin", 0, 1, 10))
```

Solutions

- Contain output of detectors
- Default plot method

```
import finesse
from finesse.analysis.actions import Xaxis

finesse.init_plotting()
sol = m.run(Xaxis(m.ETMx.R, "lin", 0, 1, 10))
sol.plot()
print(sol["pd1"])
```




```
[0.25       0.43311388 0.5236068  0.59886128 0.66622777 0.72855339
 0.78729833 0.84333001 0.8972136  0.94934165 1.        ]
```