FINESSE: Radiation pressure effects and a quantum kat


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

Daniel Brown
ddb@star.sr.bham.ac.uk
DCC: G1400205
ISC Meeting, Hannover – 03/2014

Overview

- FINESSE with radiation pressure effects (at last…). So what can it do now?
- Modelling quantum noise effects
- Efficient simulating with pykat

UNIVERSITY OF BIRMINGHAM

History

Started in 1997 by Andreas Freise as side project during his PhD

Used extensively worldwide - http://www.gwoptics.org/finesse/impact.php

Open sourced in 2012 - http://kvasir.sr.bham.ac.uk/redmine/projects/finesse

Ten Simple Rules for the Open Development of Scientific Software http://www.ploscompbiol.org/article/info%3Adoi%2F10.1371%2Fjournal.pcbi.1002802

Rule 1: Don't Reinvent the Wheel

Rule 2: Code Well

Rule 3: Be Your Own User

Rule 4: Be Transparent

Rule 5: Be Simple

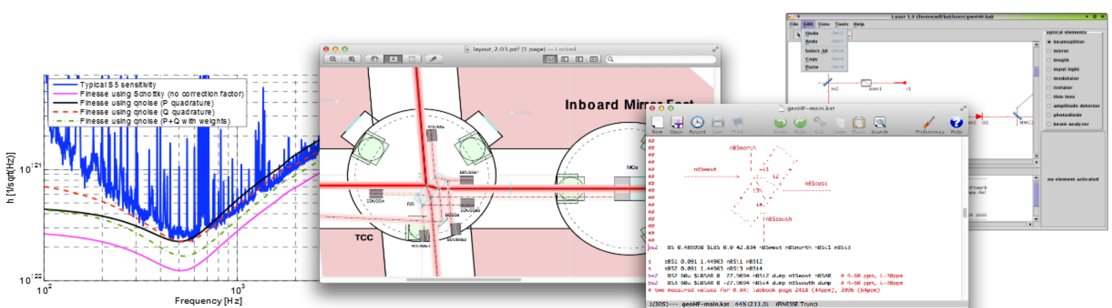
Rule 6: Don't Be a Perfectionist

Rule 7: Nurture and Grow Your Community

Rule 8: Promote Your Project

Rule 9: Find Sponsors

Rule 10: Science Counts



UNIVERSITY OF BIRMINGHAM

History

v0.99.9
- Open sourced FINESSE

v1.0
- Fixing and completing HOM features

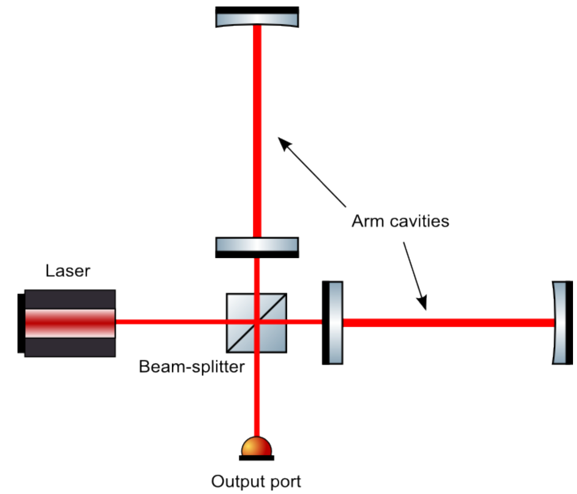
v1.1
- Internal rewrite of solver
- "Sidebands of sidebands"
- Beamsplitter maps

v1.2
- Initial implementation of radiation pressure effects
- Suspend mirrors and beamsplitters
- Longitudinal, rotational and higher order surface motions
- Testing of two-photon formalism for computing quantum noise effects

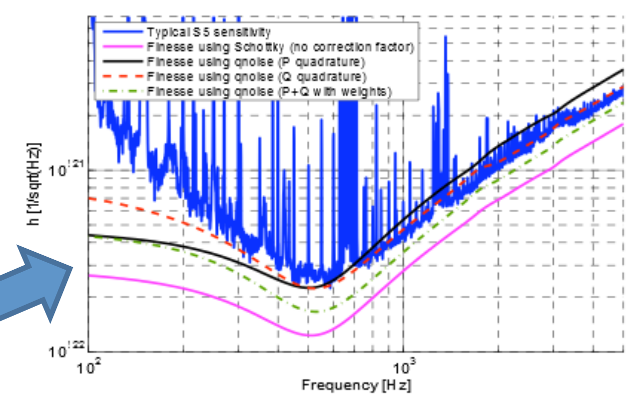
We're here ➡

And in the future v2.0, in time for GWADW 2014 as promised by Andreas last year…

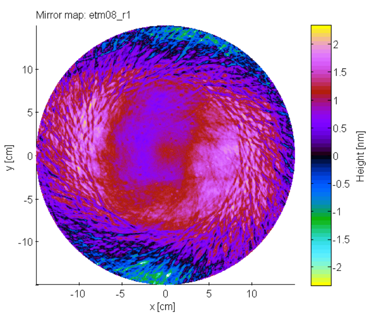
What is FINESSE?




```
%% INTERFEROMETER COMPONENTS

l L0 1 0 n1
s s0  1 n1 nbsp1
bs BSP 0.01 0.99 0 45 nbsp1 dump nbsp3 dump


s s01 1 nbsp3 n2

bs BS0 0.5 0.5 59.6 45 n2 n3 n4 n5     # Beam Splitter

const T_ITM 7e-3 # 7000ppm transmission from ET book
const T_ETM 0E-6 # 6ppm transmission from ET book

s sNin  1           n3 n6
m1 IMN $T_ITM 0 0   n6 n7
s sNarm 10000       n7 n8
m1 EMN $T_ETM 0 180 n8 dump

s sWin  1           n4 n9
m1 IMW $T_ITM 0 0   n9 n10
s sWarm 10000       n10 n11
m1 EMW $T_ETM 0 180 n11 dump
```



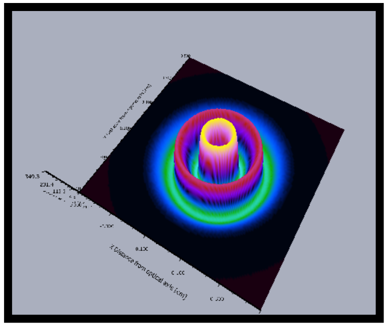
What does it do?



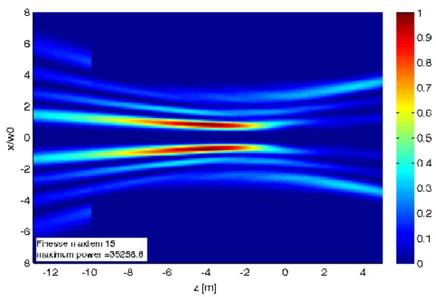
Surface and bulk distortions

- Thermal effects
- Manufacturing errors
- Surface maps

Ideal beam



- Gaussian beam
- Higher order modes like Laguerre-Gaussian (LG) beams



Finite optics

- Beam clipping
- Offsets

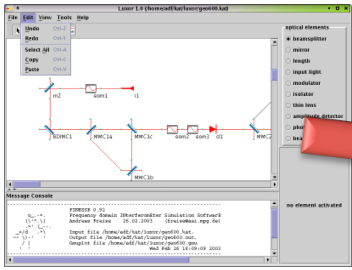
Then typically we want to compute:

- Noise couplings
- Transfer functions
- Control signals
- And more…

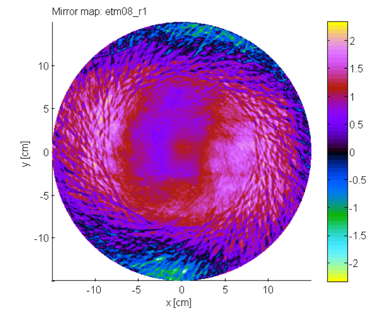
And how they are affected by distorted beams

UNIVERSITY OF BIRMINGHAM

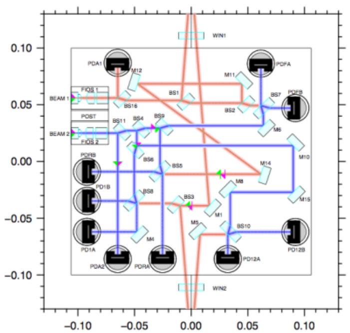
Ecosystem



Luxor
www.gwoptics.org/finesse/luxor.php

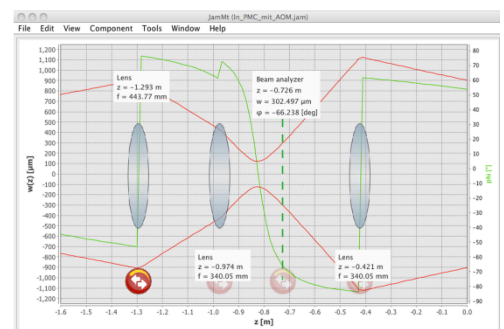


Matlab: SimTools
www.gwoptics.org/simtools/



OptoCad
http://home.rzg.mpg.de/~ros/

pyKat
www.gwoptics.org/pykat/



JamMT
http://www.sr.bham.ac.uk/dokuwiki/doku.php?id=geosim:jammt

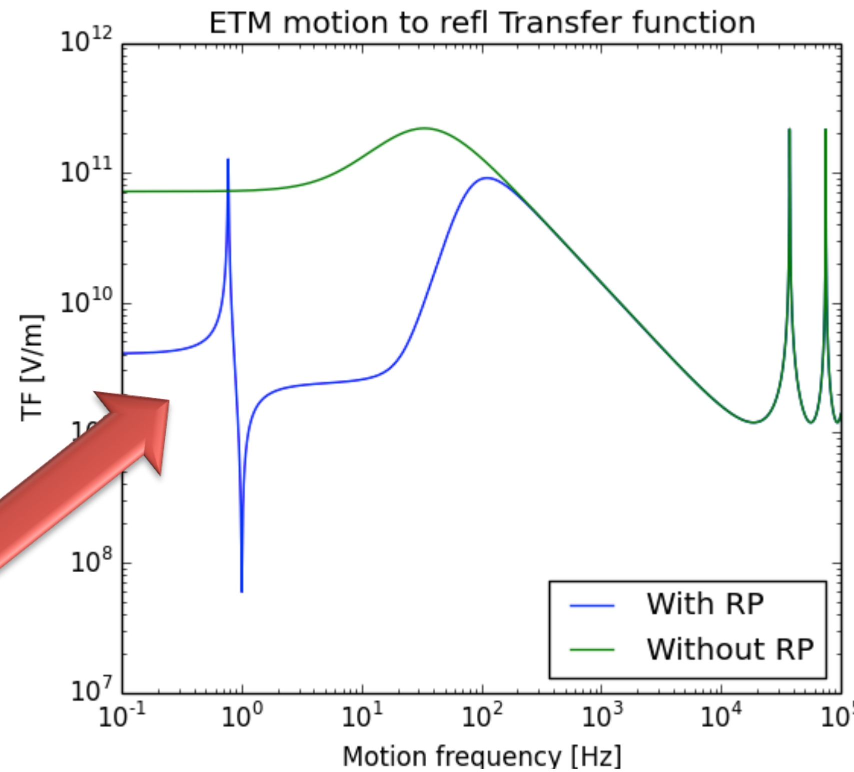
1. Implement radiation pressure effects

Radiation pressure effects

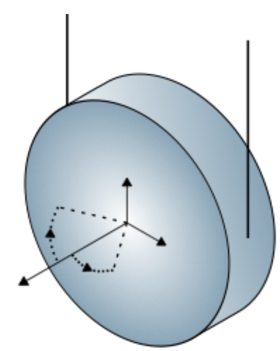
Radiation pressure creates opto-mechanical coupling

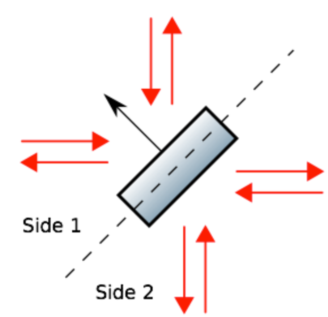
Yaw, pitch and longitudinal motions are coupled with optical fields, so this affects:

- Quantum noise transfer functions
- Displacement noise transfer functions
- Control signals
- Stability – Angular Sidles-Siggs instability for example



Need tool that can model both thermal distortions (HOM) along with radiation pressure effects for commissioning and design work

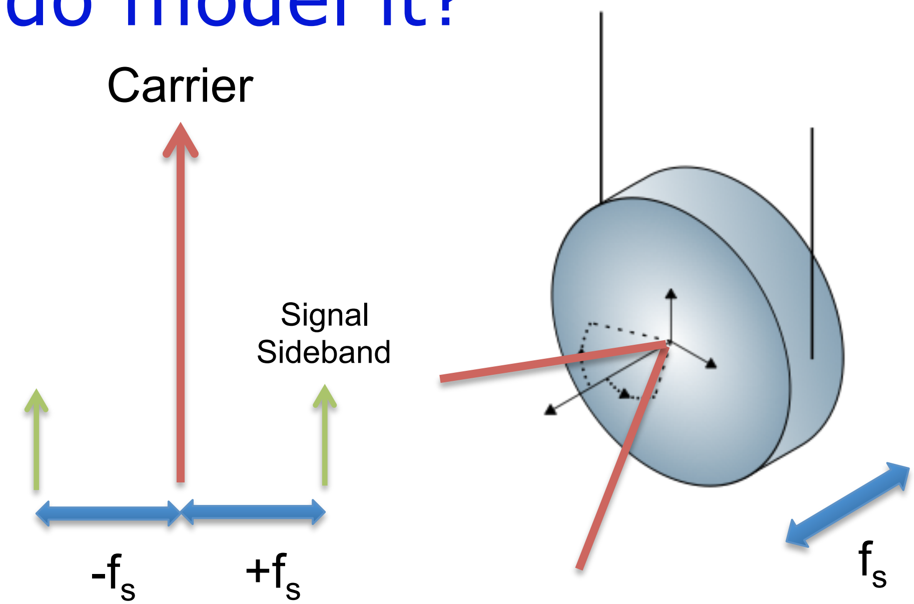




How do we do model it?

To linearise the problem we have to make a few assumptions…

- That each carrier is separated by MHz frequencies
- That high frequency beats between optical fields contribute negligibly to the force due to free mass response
- That motions of an optic are less than the wavelength of the light
- Motions are in the DC to audio frequency range
- Audio sidebands are much smaller in amplitude than carrier fields
- That all DC forces are counteracted by some control system



General surface motion to optical field coupling

$$K^{o}_{nmn'm'} = \iint_{\infty}^{\infty} u_{n'm'}(x,y) e^{i2kz_o(x,y)} u^{*}_{nm}(x,y)\,dx\,dy,$$

$$K^{s}_{nmn'm'} = \iint_{\infty}^{\infty} u_{n'm'}(x,y) z_s(x,y) u^{*}_{nm}(x,y)\,dx\,dy,$$

$$a^{\pm}_{s,jnm} = \frac{irkA^{\pm}_{s}}{\cos(\alpha)} \sum_{n',m'} a_{c,jn'm'} (K^{s}K^{o})_{nmn'm'}$$

Incoming carrier

UNIVERSITY OF BIRMINGHAM

What do we need to solve?

Longitudinal motion to optical field coupling

Surface motion is just a constant, no x/y dependence

$z_s(x,y) = Z_s$

First compute surface motion distortion, just identity matrix in this case

$$K^s_{nmn'm'} = \delta_{nn'}\delta_{mm'},$$

$$a^{\pm}_{s,jnm} = \frac{irk}{\cos(\alpha)} Z^{\pm}_s \sum_{n',m'} a_{c,jn'm'} K^o_{nmn'm'}$$

Surface motion amplitude

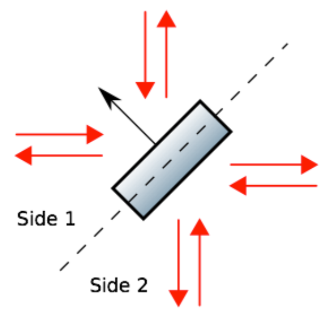
Static surface distortion

r = mirror reflectivity
k = wave number
α = angle of incidence

Optical field to longitudinal coupling

Power fluctuations at signal frequency f_s

$$P_s = \sum_j \sum_{n,m} \left(a^+_{s,jnm} a^*_{c,jnm} + a^{-*}_{s,jnm} a_{c,jnm} \right)$$



Compute power fluctuations in ALL incoming and outgoing beams

$$F_s = \frac{R\cos(\alpha)}{c}\left(-P_{s,1i} - P_{s,1o} + P_{s,2i} + P_{s,2o}\right)$$

$$Z_s = H_s \sum_n^{N_F} F_{s,n}$$

Mechanical transfer function

Final motion at frequency f_s is then sum of all forces acting on it

Higher order motions…

Rotational motion to optical field coupling

$$a^{\pm}_{s,jnm} = \frac{irk}{\cos(\alpha)} \sum_{n',m'} a_{c,jn'm'} \left(\theta^{\pm}_x (K^x K^o)_{nmn'm'} + \theta^{\pm}_y (K^y K^o)_{nmn'm'}\right)$$

Θ_x = Yaw
Θ_y = Pitch

Optical field to rotational motion coupling

Consider center of intensity oscillations to compute torque on a suspended mirror

$$\Delta x = \frac{1}{P} \iint_{-\infty}^{\infty} xE(x,y,t)E^*(x,y,t)dxdy,$$
$$= \Delta x_{DC} + \Delta x_s + O(a_s^2) + O(|f > f_s|).$$

Only interested in oscillations at frequency f_s

$$\Delta x_s = \frac{1}{P} \sum_j \sum_{n,m} \gamma^x_{nm} a^+_{s,jnm} + \gamma^{x*}_{nm} a^{-*}_{s,jnm} + c.c$$

$$\gamma^x_{nm} = K^x_{n(n+1)} a^*_{c,j(n+1)m} + K^x_{n(n-1)} a^*_{c,j(n-1)m}$$

$$\gamma^y_{nm} = K^y_{m(m+1)} a^*_{c,jn(m+1)} + K^y_{m(m-1)} a^*_{c,jn(m-1)}$$

Analytically can solve surface motion coupling Integral, find coupling integral is Hermitian

$$K^x_{nn'} = \frac{\sqrt{\max(n,n')}}{2} w_x(z) e^{i\Psi_x(z)},$$
$$K^x_{n'n} = K^{x*}_{nn'}.$$

φ_z = Gouy phase
ω_x = Beam size

$$\tau_{rp} = \frac{P}{c}\Delta x_s. \qquad \theta_x = H_{\theta_x}(\omega_s)\tau_{rp}$$

How to model optical springs

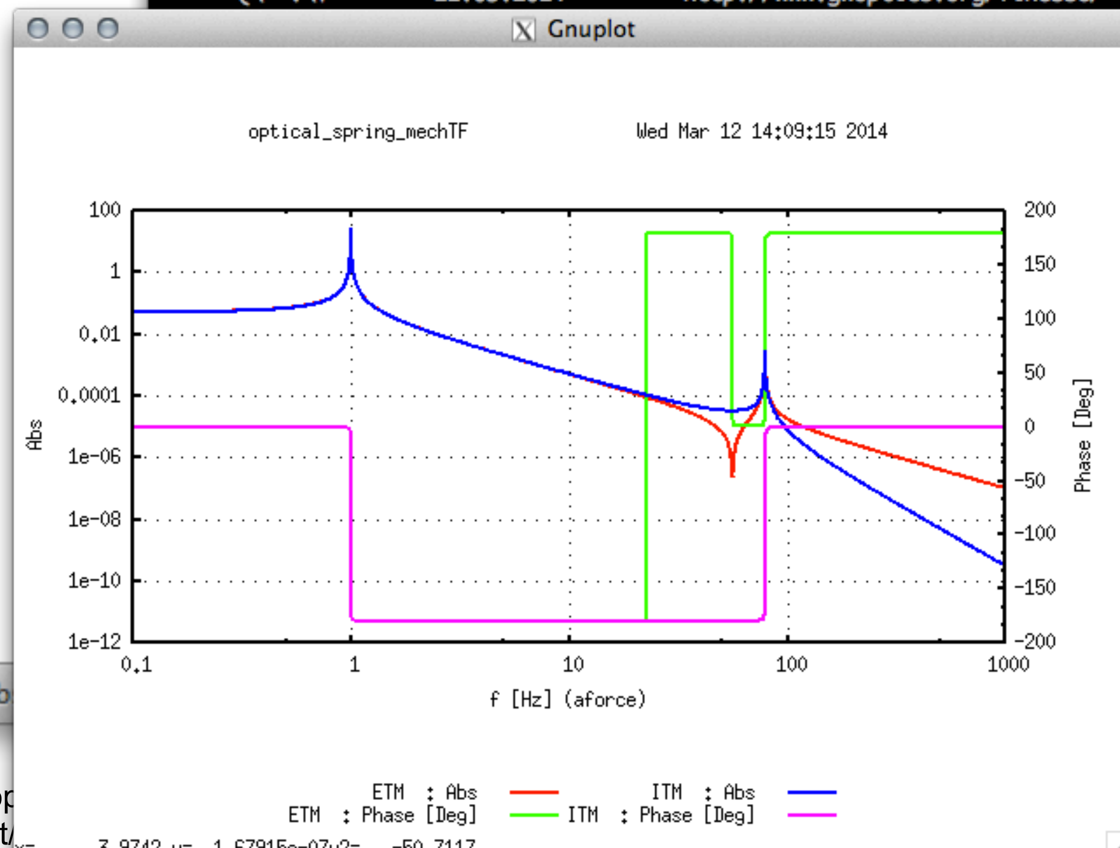
optical_spring_mechTF.kat

```
tf sus 1 0 p 1 100000

l l1 3 0 n1
m ITM 0.9937 0.0063 0 n1 n2
s cav1 1 n2 n3
m ETM 1 0 -0.048 n3 n4

attr ITM M 0.25 zmech sus
attr ETM M 0.25 zmech sus

fsig aforce ETM Fz 1 0 1

xd zETM ETM z
xd zITM ITM z

xaxis aforce f log 0.1 1k 1000
yaxis log abs:deg
```

Line: 7 | Plain Text | Soft Tab

1. bash — Python | bash

```
ddb@godel kats$ kat optical_spring_mechTF.kat

------------------------------------------------------------------------
                FINESSE 1.2.beta       (build 1.2.beta-53-g5d1deba)
     o_.-=.     Frequency domain INterferomEter Simulation SoftwarE
    (\'".\|     11.03.2014         http://www.gwoptics.org/finesse/
```



Measurement of radiation-pressure-induced op
Corbitt, et. al. 2006. http://pra.aps.org/abstract/

…and angular RP effects

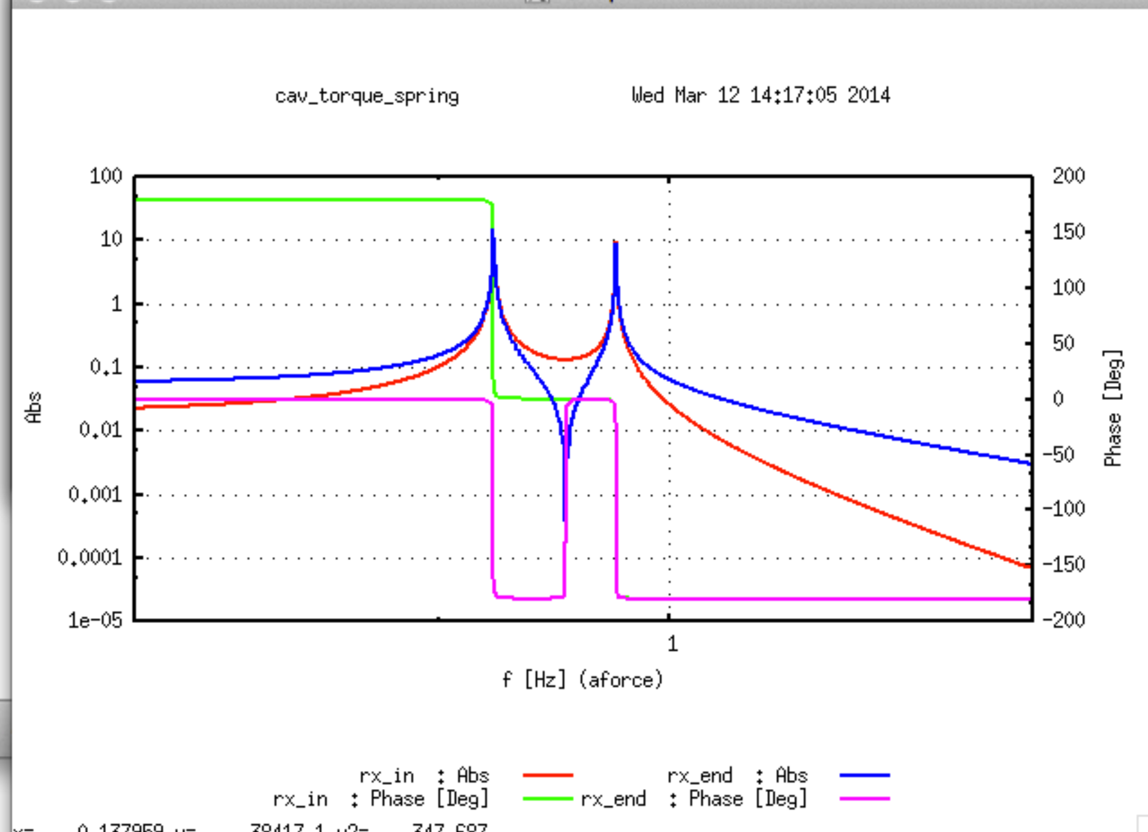
cav_torque_spring.kat

```
tf rxpend 1 0 p 0.6 1000

l l1 100 0 n1
m Min 0.99 0.01 0 n1 n2
s cav 4000 n2 n3
m Mend 1 0 0 n3 n4

attr Min Ix 1 rxmech rxpend
attr Mend Ix 1 rxmech rxpend

attr Min Rc -2076
attr Mend Rc 2076

cav c1 Min n2 Mend n3

fsig aforce Mend Frx 1 0 1

xd rx_in Min rx
xd rx_end Mend rx

xaxis aforce f log 2e-1 3 1000
yaxis log abs:deg
maxtem 1
```

Line: 22:18 Plain Text Soft Tabs:

1. bash — Python | bash

```
ddb@godel kats$ kat cav_torque_spring.kat
```



Computing PIs

Method used in FINESSE is based on "A general approach to optomechanical parametric instabilities" (Evans 2010)

Parametric gain → $$\mathcal{R}_m = \Re\mathfrak{e}\left[\frac{\Delta A_m}{A_m}\right]$$ ← Open loop transfer function of the surface motion

OLTF is computed internally already in Finesse and is extracted from the inverted signal-motion interferometer matrix for the surface motion. This can be output from a model using the new command:

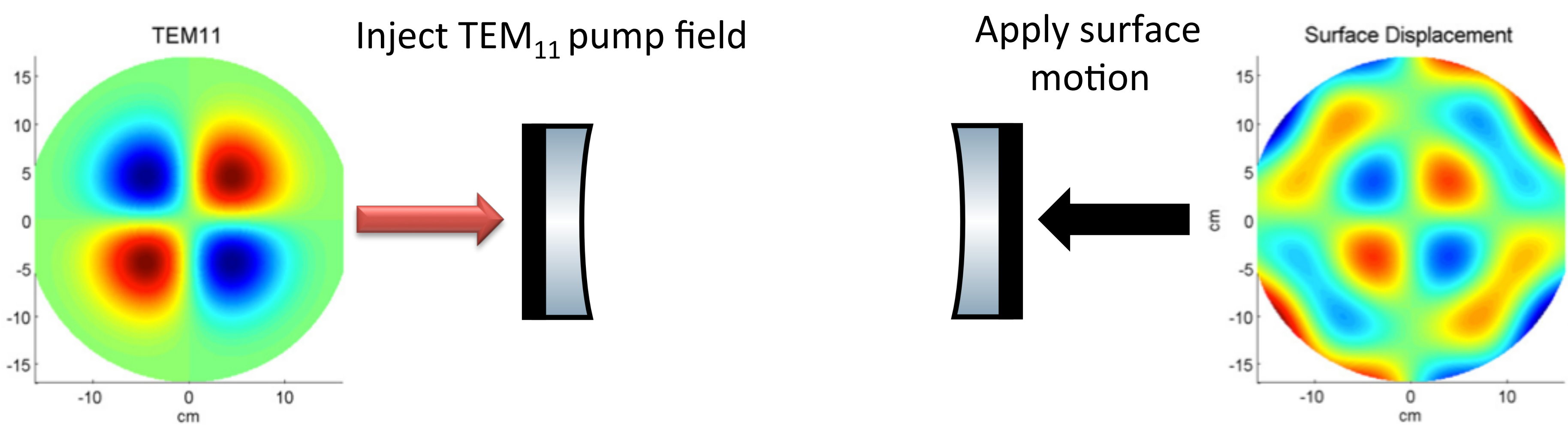
```
oltfd name component motion
```

Detector name

Component name

Motion to output

Example



Cavity Parameters, base on aLIGO arm cavities:

$P = 1MW$

$T_{itm} = 0.014$

$T_{etm} = 10^{-5}$

$L = 3994.5m$

$M = 40$ kg

Task

In the frequency range 20-50kHz we want to output the parametric gain of this surface motion.

How do we do that in Finesse?

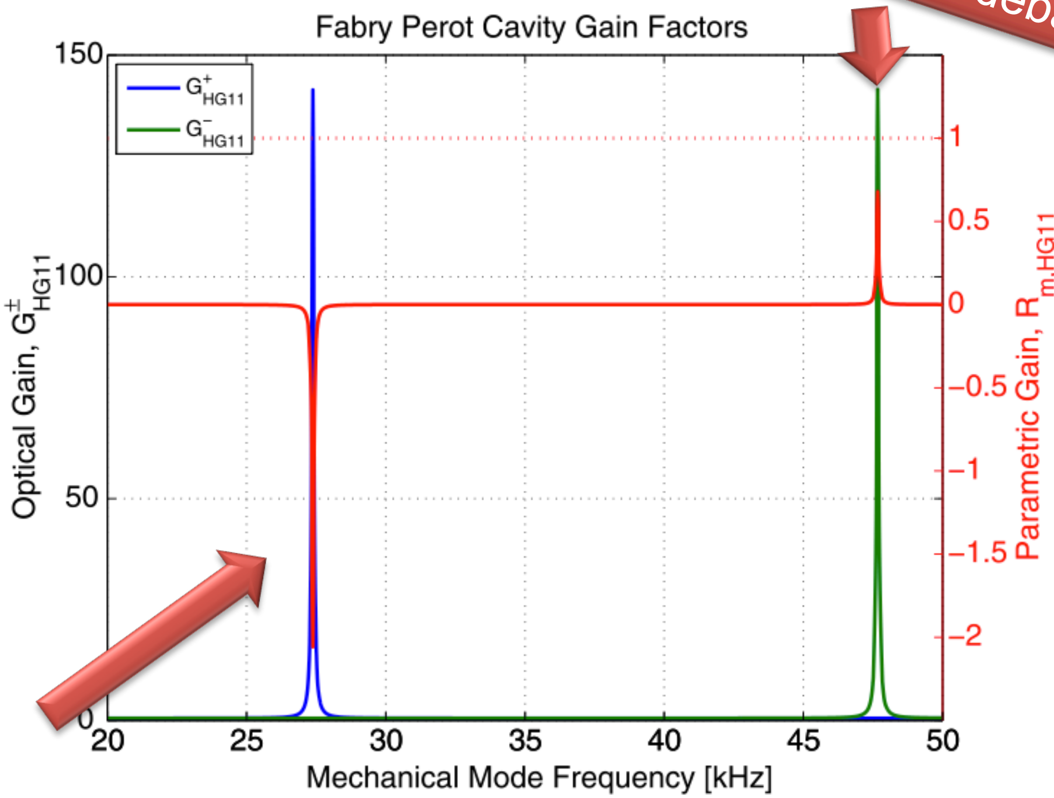
Example

New commands for doing PI modelling, don't need many…

`tf tf1 1 0 p 30k 1E7`	Create force to motion amplitude transfer function
`smotion m2 surf_mod.map tf1`	Reads in map and sets as a surface motion
`oltfd oltfd1 m2 s0`	Outputs the first surface motion at mirror m2

Example – Paper results

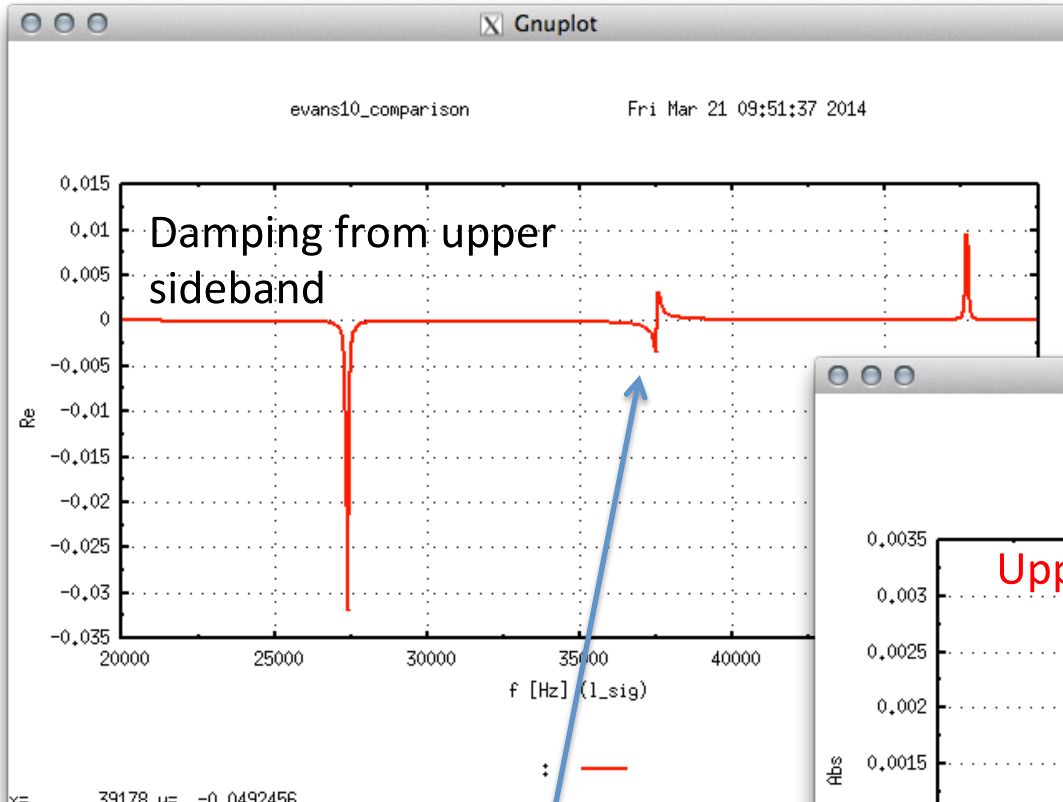
Note lower sideband positive gain…



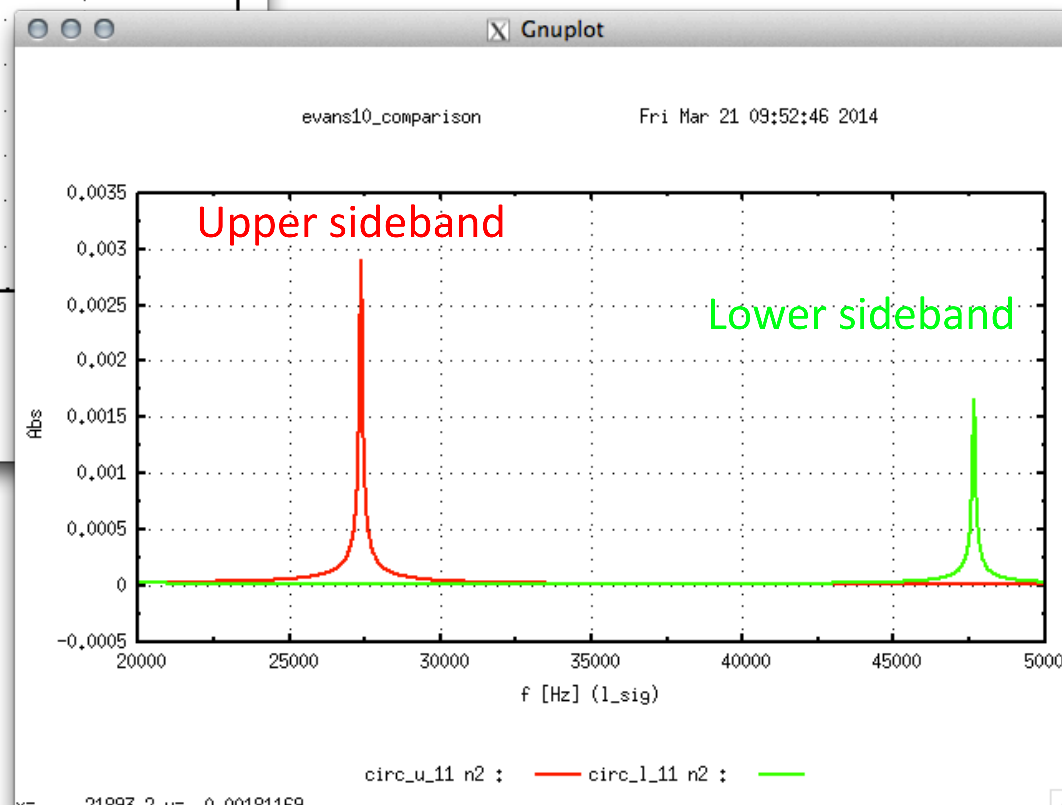
… positive sideband negative gain

UNIVERSITY OF BIRMINGHAM

Example – Finesse results



Positive parametric gain, but not instable



We compute all couplings mode couplings upto order 2 hence extra features.

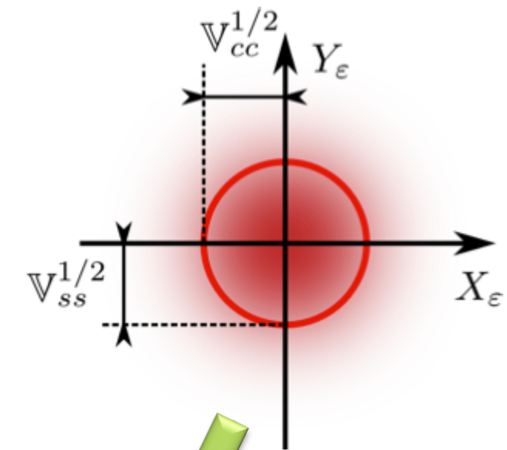
Scaling slightly different due to normalisation of map differences

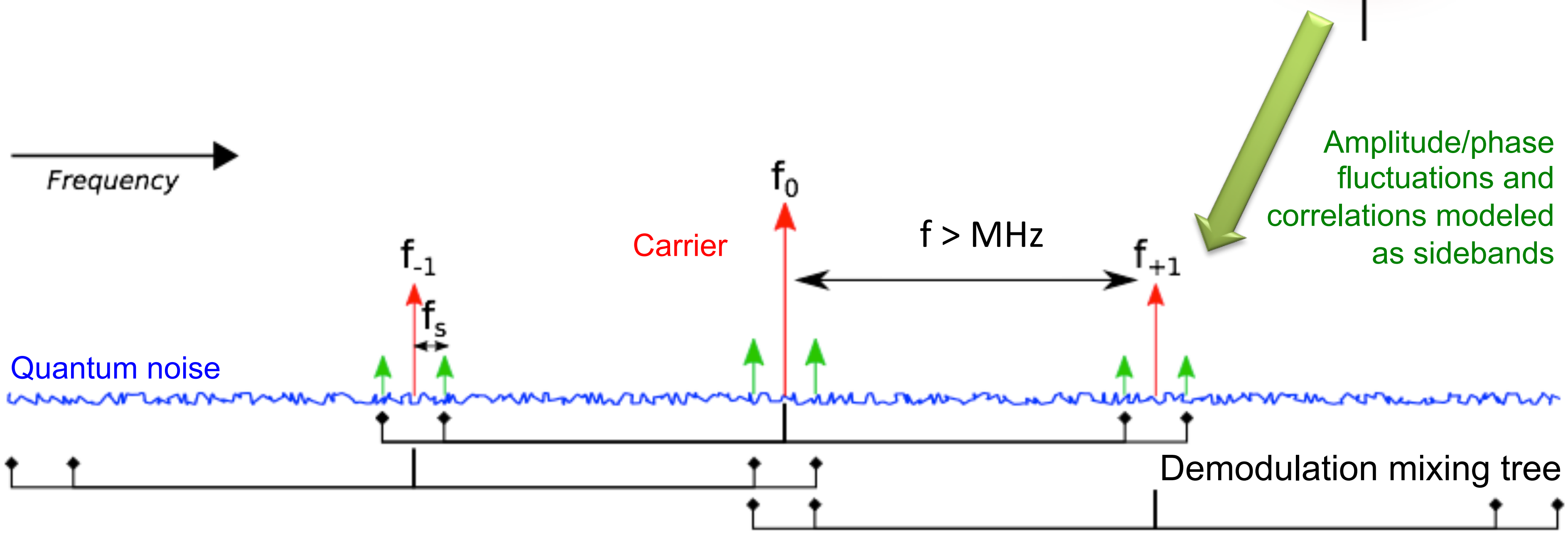
2. Implement quantum noise

What about quantum noise?

We implemented the two-photon formalism in FINESSE to compute noise at a photodiode detectors

- Need to easily include many noise sources
- Handle HOM correctly
- Noise PSD computation needs to take into account multiple carrier fields and their contribution to the noise when demodulated





Quantum noise calculation

Output quantum sidebands

Input quantum sidebands

$$\hat{M}\vec{b} = \vec{a}$$

Interferometer matrix

Selection vector

$$\hat{M}\vec{b}\vec{b}^\dagger\hat{M}^\dagger = \vec{a}\vec{a}^\dagger$$

$$\vec{b}\vec{b}^\dagger = \hat{M}^{-1}\vec{a}\vec{a}^\dagger\hat{M}^{-1\dagger}$$

Covariance matrix of noise

$$\langle\vec{b}\vec{b}^\dagger\rangle\vec{s} = \hat{M}^{-1}\langle\vec{a}\vec{a}^\dagger\rangle\hat{M}^{-1\dagger}\vec{s}$$

Matrix of noise sources

- HOM scale number of losses dramatically
- Previous method used in Optickle too slow
- Fill selection vector depending on value we want to compute

PRELIMINARY RESULTS

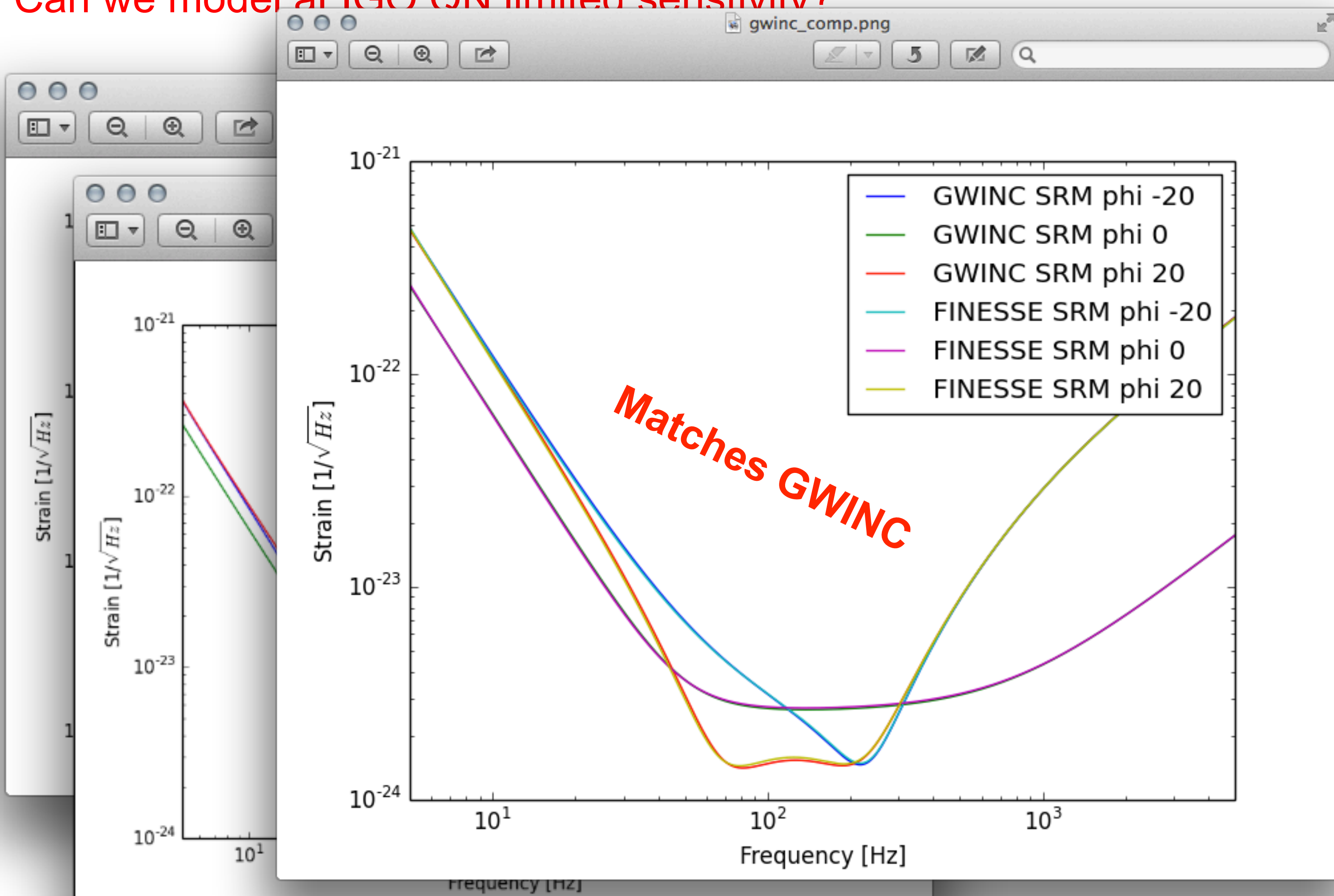
Now some examples to give an idea of what you can model so far…

We're confident the numbers are correct we just want to be sure before letting it out into the public that is passes all the tests we throw at it.

UNIVERSITY OF
BIRMINGHAM

Some toy aLIGO examples…

Can we model aLIGO QN limited senstivity?



e new

e
s 0 node

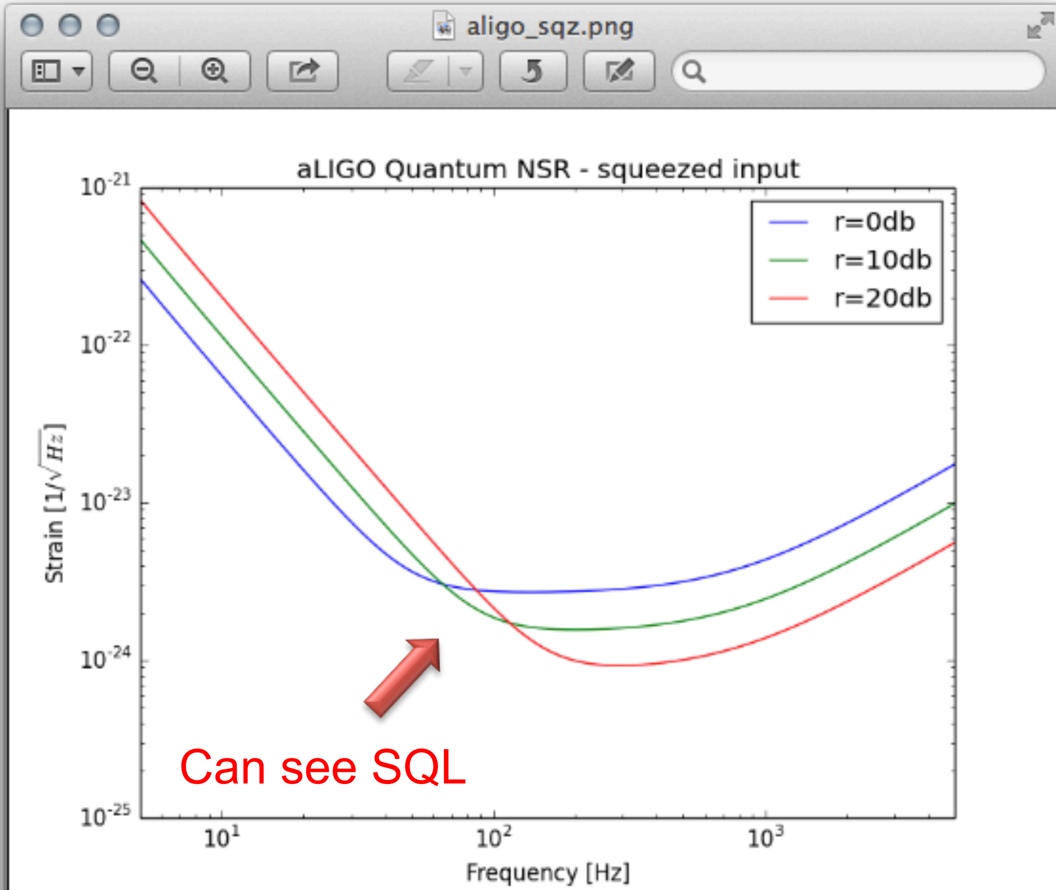
oise for a
e signal.

C

nt with
rs, like SRM

UNIVERSITY OF BIRMINGHAM

How about some squeezing?



Easy enough, just add a squeezed source at the dark port…

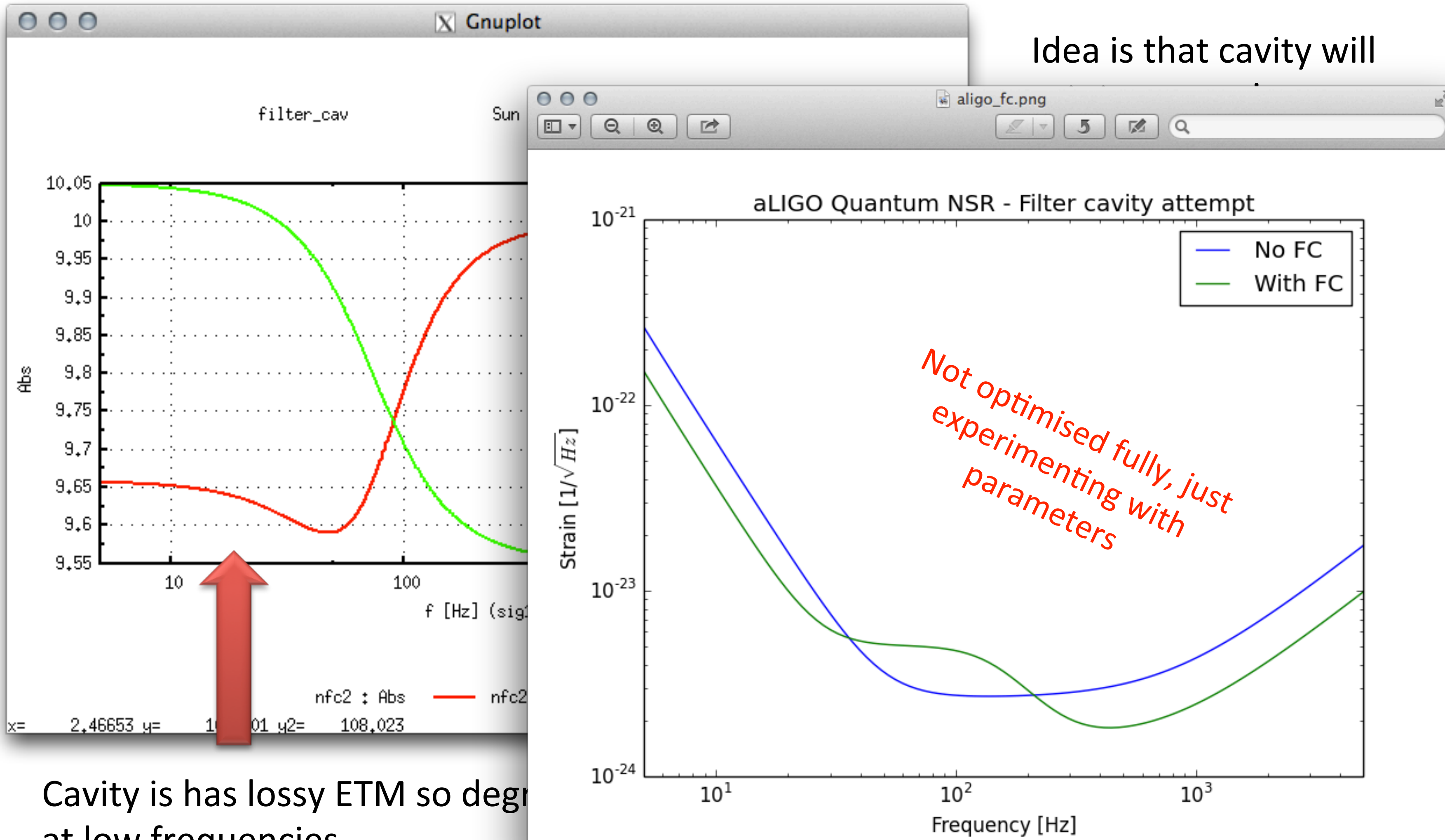
Squeezing db and angle

```
sq sq1 0 10 90 nfc1
```

Carrier frequency to squeeze

…Filter cavities?

Idea is that cavity will



Cavity is has lossy ETM so degr
at low frequencies

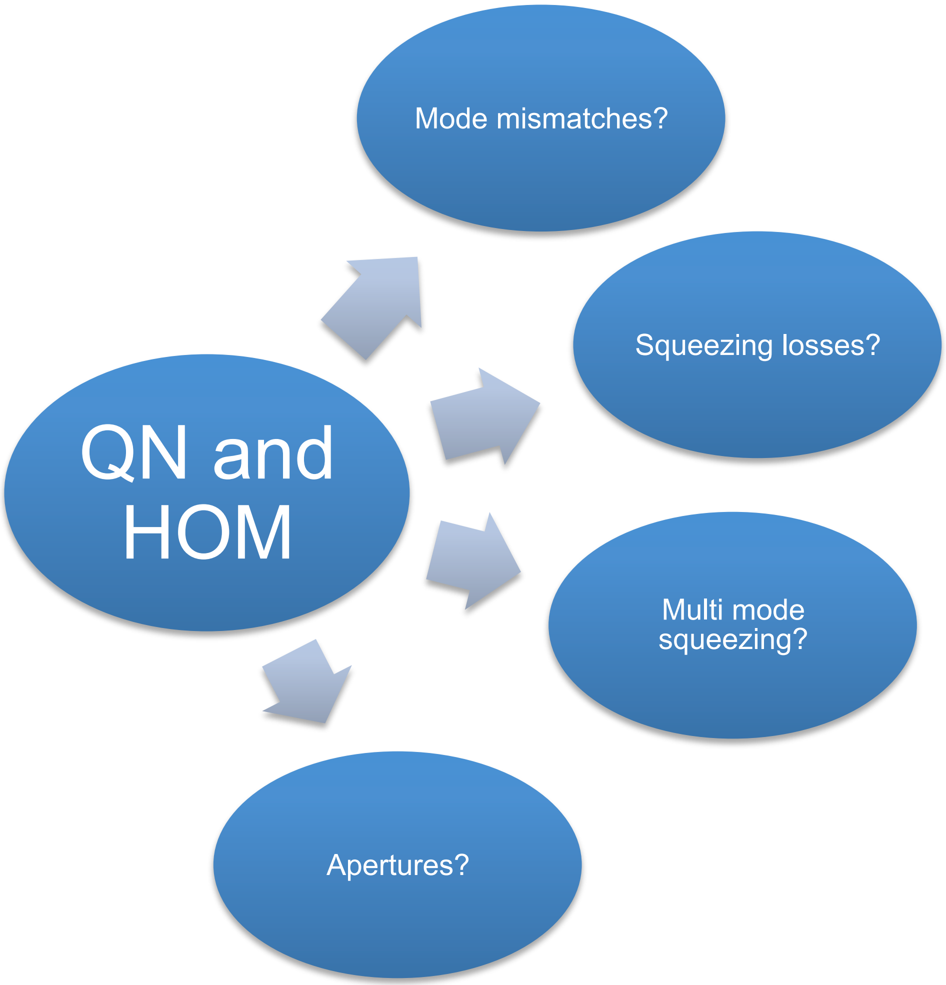
UNIVERSITY OF BIRMINGHAM

So what next with quantum noise?

Still open questions about how to implement higher order modes and quantum noise…

Modulator components currently assume vacuum noise in and vacuum noise out, e.g. can't inject squeezed field through one…

Can we model it well enough…?



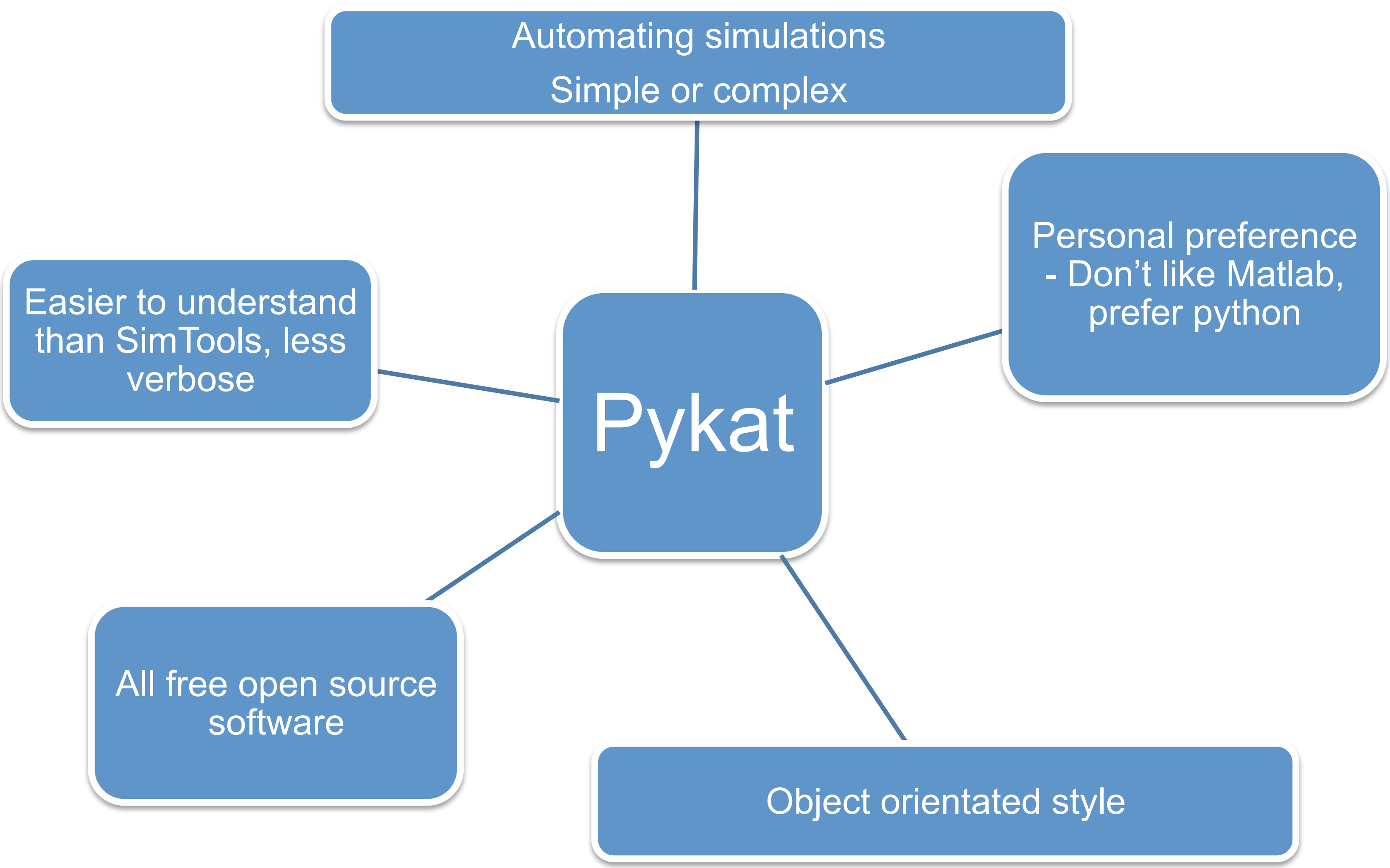
pykat

```
                                                 ..-
                          _                     '(
                          \`.|\.__...-""""-_." )
       ..+-----.._        /  ' `             .-'
   . '            `:      7/* _/._\    \   (
  (        '::;;+;;:      `-"' =" /,`"" `) /
  L.        \`:::a:f           c_/     n_'
  ..`--...___`.  .    ,
   `^-....____:   +.      www.gwoptics.org/pykat
```

A python interface for FINESSE

It’s still pretty new and has more features to be added but is usable today.

Why use pykat?



The general idea

Build optical setups

Kat file

Kat file

Kat file

Can use block layout
as used in SimTools

Load files into pykat

- Run multiple simulations quickly
- Apply optimisation routines
- More complex analysis

How does it work

Load strings of Finesse commands…

```
code = """
l l1 1 0 0 n2
m m1 0.5 0.5 0 n2 n3
s s2 10 1 n3 n4
m m2 0.5 0.5 0 n4 n5

pd circ n3
"""

kat = finesse.kat()

kat.parseCommands(code)
```

…or load code from kat files

```
kat = finesse.kat()

Kat.loadKatFile(“file.kat”)
```

By loading or parsing Finesse code we “fill” the kat object with the components, detectors and commands so that we can interact with them

How does it work

Object orientated access to component, detectors and commands…

```
kat.l1.P = 1 # set laser power to 1W

kat.m1.phi = 45 # set tuning of mirror to 45 degrees

# set curvatures of mirrors
kat.m1.Rcx = -1000.0
kat.m1.Rcy = -1000.0
kat.m2.Rcx =  1000.0
kat.m2.Rcy =  1000.0

# change global settings of the simulation
kat.maxtem = 3
kat.yaxis = “abs:deg”
```

How does it work

Once parameters have been defined we run the simulation…

```
kat.add(xaxis("lin", [0, 360], kat.m2.phi, 100))

# run current setup, output object contains all the data for
# that run
out1 = kat.run()

# Change whatever you want…
kat.m1.R = 0.2
kat.m1.T = 0.8

# and run the simulation again getting a separate output object
out2 = kat.run()
```

How does it work

Output objects contain all the information from detectors, from this we can plot data or use computed results to feed into a new simulation

```
# the values of xaxis used
print out.x

# Select outputs by detector name. If you output complex data
# with ‘yaxis re:im or abs:deg’ the output is converted to
# complex numbers
print out[“circ”]

print out.ylabels, out.xlabel

# Quick plot for data
out.plot()
```

Now for some hands on demos…

Can download latest pykat from www.gwoptics.org/pykat using Git

Or install it via pip “pip install pykat”

Recommended usage is with the interactive ipython shell http://ipython.org/