notes

April 23, 2025

1 Using HOM Spacing Measurements to Check Various Thermal Actuator Coupling Factors

1.1 Laying out the problem

We have a slurry of known parameters in the IFO's cold state (no Ring Heaters or self-heating), and with some measurements of the Higher Order Mode spacing (HOM) we can constrain or calculate the values of coupling factors of self-heating and ring heater power to surface curvature changes.

Table of physical parameters we know in the **cold** state:

Parameter	Symbol	Value	Unit
ITMY Radius of Curvature	$R_{\text{i, c}}$	1939.2	m
ETMY Radius of Curvature	$R_{\text{e, c}}$	2246.9	m
ARM Length	L	3994.47	m

1.2 Cavity g-factors

We can relate the cavity g-factor (a measure of the test masses' curvature relative to the cavity length) to the HOM spacing data.

Here is a list of symbols used in the derivation:

Parameter	Symbol
ITM g-factor	g_i
ITM g-factor, cold-state	$g_{i,c}$
ETM g-factor	g_e
ETM g-factor, cold-state	$g_{e,c}$
Round-trip Gouy Phase	ψ_{rt}
Curvature, cold-state (ITM, ETM)	$D_{i,c}, D_{e,c}$
Curvature, hot-state (ITM, ETM)	D_i, D_e
Self-heating curvature coupling (ITM, ETM)	A_i, A_e
Ring-heater curvature coupling	B
Self-heating power (ITM, ETM)	$P^i_{\text{self}}, P^e_{\text{self}}$
Ring-heater power (ITM, ETM)	$P^i_{\text{rh}}, P^e_{\text{rh}}$

To start we relate the HOM spacing to the cavity g-factor:

$$f_{\mathrm{HOM}} = \frac{\psi_{rt}}{2\pi} f_{\mathrm{FSR}}$$
$$\psi_{rt} = 2\arccos\left(\sqrt{g_i g_e}\right)$$
$$g_i g_e = \cos^2\left(\pi \frac{f_{\mathrm{HOM}}}{f_{\mathrm{FSR}}}\right)$$

Then we express the test mass g-factors in terms of the thermal contributions

$$g_i = 1 - LD_i, \quad D_i = D_{i,c} + A_i P^i_{\mathrm{self}} + BP^i_{\mathrm{rh}}$$
$$g_e = 1 - LD_e, \quad D_e = D_{e,c} + A_e P^e_{\mathrm{self}} + BP^e_{\mathrm{rh}}$$

Multiplying them together and assuming second order terms are small:

$$g_i g_e = 1 - L\left(D_{i,c} + D_{e,c} + A_i P^i_{\mathrm{self}} + A_e P^e_{\mathrm{self}} + B(P^i_{\mathrm{rh}} + P^e_{\mathrm{rh}})\right)$$
$$+L^2\left(D_{i,c}D_{e,c} + D_{i,c}(BP^e_{\mathrm{rh}} + A_e P^e_{\mathrm{self}}) + D_{e,c}A_i P^i_{\mathrm{self}} + \ (A_i A_e + (A_i + A_e)B + B^2 \text{ terms})\right)$$

The terms on the right side are assumed to be small and negligible.

$$g_i g_e = 1 - L(D_{i,c} + D_{e,c}) + L^2 D_{i,c}D_{e,c} - L\left(A_i P^i_{\mathrm{self}} + A_e P^e_{\mathrm{self}} + BP^e_{\mathrm{rh}}\right)$$
$$+L^2\left(D_{i,c}(BP^e_{\mathrm{rh}} + A_e P^e_{\mathrm{self}}) + D_{e,c}A_i P^i_{\mathrm{self}}\right)$$

We can gather terms by groups of cold state terms, B depedent terms, and self-heating terms:

$$g_i g_e = X_c - BLP^e_{\mathrm{rh}}(1 - LD_{i,c}) + X_s = X_c + X_s - BLP^e_{\mathrm{rh}} g_{i,c}$$
$$\cos^2\left(\pi \frac{f_{\mathrm{HOM}}}{f_{\mathrm{FSR}}}\right) = X_c + X_s - BLP^e_{\mathrm{rh}} g_{i,c}$$

where,

$$X_c = 1 - L(D_{i,c} + D_{e,c}) + L^2 D_{i,c}D_{e,c} = g_{i,c}g_{e,c}$$
$$X_s = -L\left(A_i P^i_{\mathrm{self}}(1 - LD_{e,c}) + A_e P^e_{\mathrm{self}}(1 - LD_{i,c})\right) = -L\left(A_i P^i_{\mathrm{self}} g_{i,c} + A_e P^e_{\mathrm{self}} g_{e,c}\right)$$

1.3 Fitting data to recover B and constrain X_s

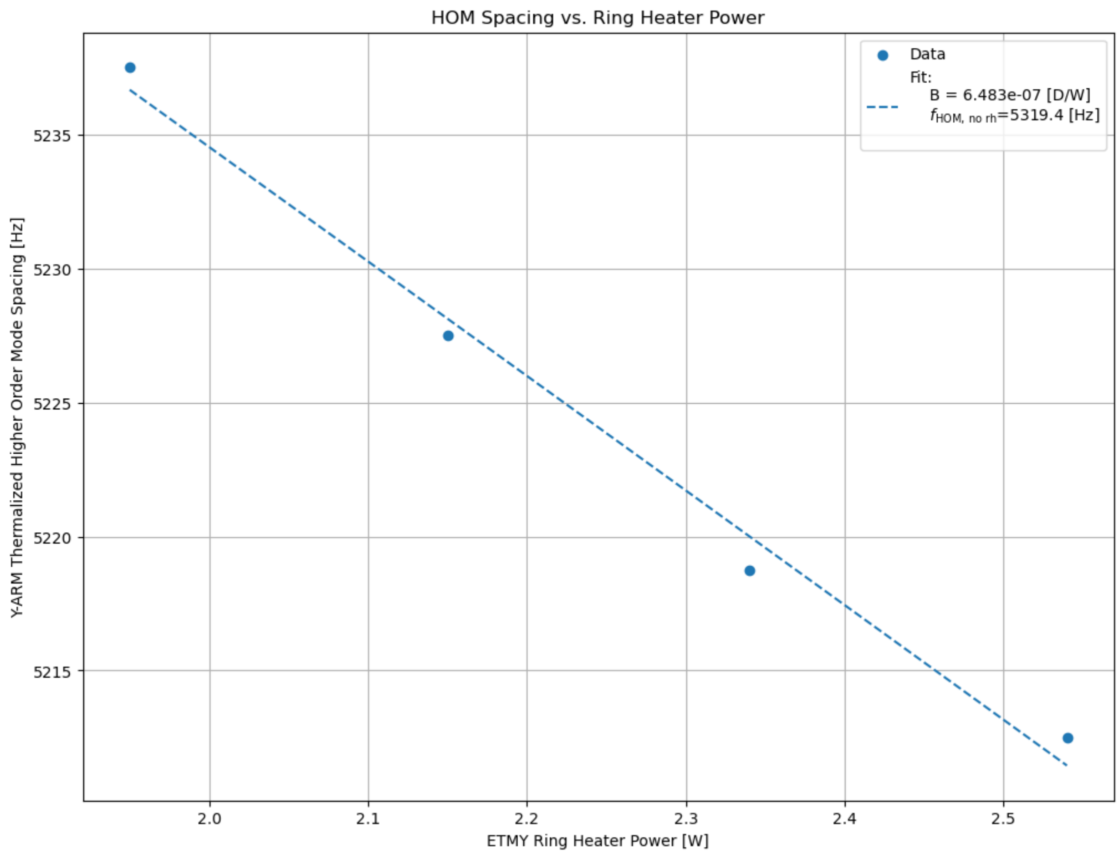
By fitting a line ($y = mx + b$) to the HOM data as a function of P^e_{rh}, we can recover the HOM change due to self heating as well as the coupling of ring-heater power to curvature (B):

$$\cos^2\left(\pi \frac{f_{\mathrm{HOM}}}{f_{\mathrm{FSR}}}\right) = X_c + X_s - BLP^e_{\mathrm{rh}} g_{i,c}$$
$$y = \cos^2\left(\pi \frac{f_{\mathrm{HOM}}}{f_{\mathrm{FSR}}}\right), \ m = -BLg_{i,c}, \ x = P^e_{\mathrm{rh}}, \ b = X_c + X_s$$

[3]:
```python
import numpy as np
import scipy.constants as scc
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
```

[4]:
```python
# known parameters
L = 3994.47 # m
FSR = scc.c / (2 * L) # Hz
static_etm_roc = 2246.90 # m
static_itm_roc = 1939.20 # m
static_etm_curv = 1 / static_etm_roc
static_itm_curv = 1 / static_itm_roc

# measured parameters
etmy_rh_power = np.array([1.95, 2.15, 2.34, 2.54])
HOM_LG10 = np.array([10.475e3, 10.455e3, 10.4375e3, 10.425e3])

# calculate the slope and intercept
def func(X, m, b):
    return m*X + b

gige = np.cos(np.pi * HOM_LG10 / 2 / FSR)**2
popt, pcov = curve_fit(func, etmy_rh_power, gige)
slope, intercept = popt

g_ic = 1 - L * static_itm_curv
B = slope / (-L * g_ic)

X_c = (1 - L * static_itm_curv)*(1 - L * static_etm_curv)
X_s = intercept - X_c
self_heating_curv = -X_s/L
HOM_noRH = FSR*np.arccos(np.sqrt(intercept))/np.pi
HOM_cold = FSR*np.arccos(np.sqrt(X_c))/np.pi
label = "Fit:\n" + f"    B = {B:.3e} [D/W]\n    " + \
        r"$f_\text{HOM, no rh}$" + \
        f"={HOM_noRH:.1f} [Hz]\n"

fig, axs = plt.subplots(figsize=(12,9))
axs.scatter(etmy_rh_power, HOM_LG10 / 2, label='Data')
axs.plot(np.linspace(etmy_rh_power[0], etmy_rh_power[-1], 100),
         FSR*np.arccos(np.sqrt(func(np.linspace(etmy_rh_power[0],␣
 ↪etmy_rh_power[-1], 100), *popt)))/np.pi,
         ls = '--',
         label=label)

axs.set_xlabel("ETMY Ring Heater Power [W]")
axs.set_ylabel("Y-ARM Thermalized Higher Order Mode Spacing [Hz]")
```

```
axs.grid(True, 'both', 'both')
axs.set_title("HOM Spacing vs. Ring Heater Power")
axs.legend()
fig.savefig("./figures/rh_power_to_surf_curv_fit.pdf")
```



2