1 FPMI frequency response to gravitational wave perturbation*

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
import scipy.signal as sig
import os
from ifo_configs import mich_freq_resp as MICH
from ifo_configs import fpmi_freq_resp as FPMI
from ifo_configs import N_shot, bode_amp, bode_ph
plt_style_dir = 'stash/'
if os.path.isdir(plt_style_dir) == True:
    plt.style.use(plt_style_dir + 'ppt2latexsubfig.mplstyle')
plt.rcParams["font.family"] = "Times New Roman"
line_width=7.5
```

I often find it helpful to revisit that which we will need to build upon before moving forward, especially if one has not done this derivation more than a couple of times. Sooooo...

1.1 Last time when working with IFO configs:

Michelson frequency response to gravitational wave

The Michelson interferometer by it's design is a measurement device that can detect small changes in light phase between it's two perpendicular arms.

A gravitational wave offers a unique phase differential that can be characterized mathematically by the following:

$$\phi_X - \phi_Y = \int_{t-2L/c}^{t} \Omega\left[1 + \frac{1}{2}h(t)\right]dt - \int_{t-2L/c}^{t} \Omega\left[1 - \frac{1}{2}h(t)\right]dt \tag{1}$$

This eventually led us to the time independent phase response to a monochromatic gravitational wave ($h(t)$):

$$\Delta\phi(\omega) = h_0 \frac{2L\Omega}{c} e^{-iL\omega/c} \frac{\sin(L\omega/c)}{L\omega/c} \tag{2}$$

Now, we are going to look at how Fabry Perót cavitites can help our Michelson become a gravitational wave observatory

2 Derivation

Let's start with the simple Fabry Perót cavity. The following are equations that characterize the circulating and reflected fields (both critical to measuring the phase response of the FP cavity to GWs):

$$E(t) = t_1 E_{in} + r_1 r_2 E(t - 2T) e^{-i\Delta\phi(t)} \tag{3}$$

$$E_r(t) = -r_1 E_{in} + t_1 r_2 E(t - 2T) e^{-i\Delta\phi(t)} \tag{4}$$

$T = L/c$ is the time it takes light to reach the end of the cavity and $\Delta\phi(t)$ is the phase rotation.

We can define the static phase rotation (no GW passing through) as :

$$\Delta\phi = 2kL = 4\pi L/\lambda_{opt} \tag{5}$$

And if L is tuned just right $2kL = 2\pi n$ so the cavity is just tuned for resonance

If we put a gravitational wave in the mix we redefine this phase rotation as such that:

$$\Delta\phi = \frac{\omega_0}{2} \int_{t-\frac{2L}{c}}^{t} h(t')dt' \tag{6}$$

This assumes that the static phase rotation satisfies $2\omega_0 L/c = 2\pi n$. Which is the same thing that we said above but with different symbols (because we're fancy ;D)

Say that we have something that does throw the cavity slightly off resonance.. doesn't have to be a gravitational wave... but that's what we hope for. ANYWAY...

If the $\Delta\phi$ becomes such that the cavity is thrown off resonance we get a time dependent intra-cavity field:

$$E(t) = \bar{E} + \delta E(t) \tag{7}$$

and if the phase rotation ($\Delta\phi$) is super small... which is pretty much guaranteed with gravy waves, we can say:

$$e^{i\Delta\phi} = 1 - i\Delta\phi \tag{8}$$

Using equations 7 and 8 in 3 we get:

$$\bar{E} + \delta E(t) = t_1 E_{in} - r_1 r_2 \bar{E} + r_1 r_2 \delta E(t - 2T) - i r_1 r_2 \bar{E} \Delta\phi(t)) \tag{9}$$

We can parse this into time dependent and time independent terms:

$$\bar{E} = t_1 E_{in} - r_1 r_2 \bar{E} \tag{10}$$

$$\delta E(t) = r_1 r_2 \delta E(t - 2T) - i r_1 r_2 \bar{E} \Delta\phi(t) \tag{11}$$

Since the time dependent phase information is encoded in 11 we will take the laplace transform of this equation to yield:

$$\delta E(s) = -i \frac{r_1 r_2 \bar{E}}{1 - r_1 r_2 e^{-2sT}} \Delta\phi(s)$$

YAS! we are now one step closer to getting a useful expression for the phase response. But again.. what does this last equation mean? That last equation is how the change in the electric field directly relates to a small perturbation in phase (which could be either a small change in laser frequency or length modulation)

Now.. we're not done yet because that last expression does not tell us the entire story yet.. we want to see how this effects the phase differential with the **reflected** electric field.

To do this.. we have to combine equations 3 and 4. (*an easy way to do this is to get rid of the $ r_2 E(t - 2T) e^{-i Δφ(t)}$ term*) :

$$E_r(t) = \frac{t_1}{r_1}E(t) - \frac{t_1^2 + r_2^2}{r_1}E_{in}$$

if the cavity is unperturbed:

$$\bar{E}_r = \left(\frac{r_2(r_1^2 + t_1^2) - r_1}{t_1}\right)\bar{E}$$

and if we perturb the cavity we see that the change in the intra-cavity field is directly related to the change in the reflected field:

$$\Delta\phi_r(s) \equiv \frac{\delta E(s)}{\bar{E}} = \frac{t_1^2 r_2}{(t_1^2 + r_1^2)r_2 - r_1}\frac{\Delta\phi(s)}{1 - r_1 r_2 e^{-2sT}}$$

This implies that there is an additional frequency dependent factor in your phase shift and this translates into your FPMI transfer function as:

$$H_{FPMI}(\omega_g) = \frac{2\Delta\phi_r(\omega_g)}{h(\omega_g)} = \frac{t_1^2 r_2}{(t_1^2 + r_1^2)r_2 - r_1}\frac{H_{\mathrm{MI}}(\omega_g, L)}{1 - r_1 r_2 e^{-2i\omega_g L/c}}$$

Whew.... that was a lot.... now let's code it up

Since we can seperate the calculation into two.. I'm going to parse out the calculation between the constant Fabry Perót term and the term with the frequency dependence. But first, lets set up our parameters for our FPMI:

[2]:
```
# Some parameters
cee = np.float64(299792458)
OMEG = np.float64(2*np.pi*cee/(1064.0*1e-9))
L = np.float64(4000.0)
nu = np.arange(1, 1000000, 1)
nat_nu = [np.float64(i*2*np.pi) for i in nu]
h_0 = np.float64(1)

PHI_0 = np.pi/2 #[rad]
P_IN = 25

T_1 = .014
```

```
#T_1 = 25e-6
T_2 = 50e-6
R_1 = 1-T_1
R_2 = 1-T_2

t_1 = T_1**.5
r_1 = R_1**.5
r_2 = R_2**.5
```

Now we can compute:

$$H_{FPMI}(\omega_g) = \frac{t_1^2 r_2}{(t_1^2 + r_1^2)r_2 - r_1} \cdot \frac{H_{\mathrm{MI}}(\omega_g, L)}{1 - r_1 r_2 e^{-2i\omega_g L/c}}$$

[3]:
```
FPMI?
```

```
Signature: FPMI(freq, r_1, t_1, r_2, L, phi_0, P_in, OMEGA, low_pass=False)
Docstring:
FABRY PEROT MICHELSON FREQUENCY RESPONSE CALCULATOR
freq : standard (gravitational wave) frequency [Hz]
r_1, t_1, r_2: Assuming arm symmetry where the ITM has r_1, t_1 coefficients and␣
 ↪the ETM has a r_2 reflectivity coefficient. Also assumes no loss. [arb]
OMEGA: OPTICAL angular frequency [rad Hz]
Length: Michelson ifo arm length [m]
phi_0 : static differential arm length tuning phase [rad]
File:      ~/Documents/git/SU/dissertation/code/ifo_configs.py
Type:      function
```

[4]:
```
H_FPMI = FPMI(nu, r_1, t_1, r_2, L, PHI_0, P_IN, OMEG)
```

We estimate the FP's pole frequency

$$1 - r_1 r_2 e^{-2i\omega_g L/c} = 0$$

therefore when:

$$e^{-i\omega_g L/c} = \frac{1}{\sqrt{r_1 r_2}}$$

we acquire the pole frequency ω_{pole} as indicated in the low pass

$$f_{\mathrm{pole}} = \frac{1}{4\pi\tau_s} = \frac{c}{4\pi L}\frac{1 - r_1 r_2}{\sqrt{r_1 r_2}} = \frac{\nu_{\mathrm{FSR}}}{2\pi}\frac{1 - r_1 r_2}{\sqrt{r_1 r_2}} = \frac{\nu_{\mathrm{FSR}}}{\mathcal{F}}$$

ALso, understanding that the cavity Finesse can be defined as

$$\mathcal{F} = \frac{\pi\sqrt{r_i r_e}}{1 - r_i r_e}$$

we also can invert for a high value of finesse

$$\mathcal{F} >> \pi$$

:

$$r_i r_e \approx 1 - \frac{\pi}{\mathcal{F}}$$

[5]:
```
f_pole = 1/(((4*np.pi*L)*np.sqrt(r_1*r_2))/(cee*(1-r_1*r_2)))
def fpmi_lp(freq, cav_pole):
    return 1/(1 + 1j*(freq/cav_pole))#*np.exp(1j*freq/cav_pole))
H_FPMI_LP = fpmi_lp(nu, f_pole)
```

Might as well compare it to our Michelson response:

$$H_{\mathrm{MI}}(\omega_g) = \frac{2L\Omega}{c} e^{-iL\omega/c} \frac{\sin(L\omega/c)}{L\omega/c}$$

[6]:
```
MICH?
```

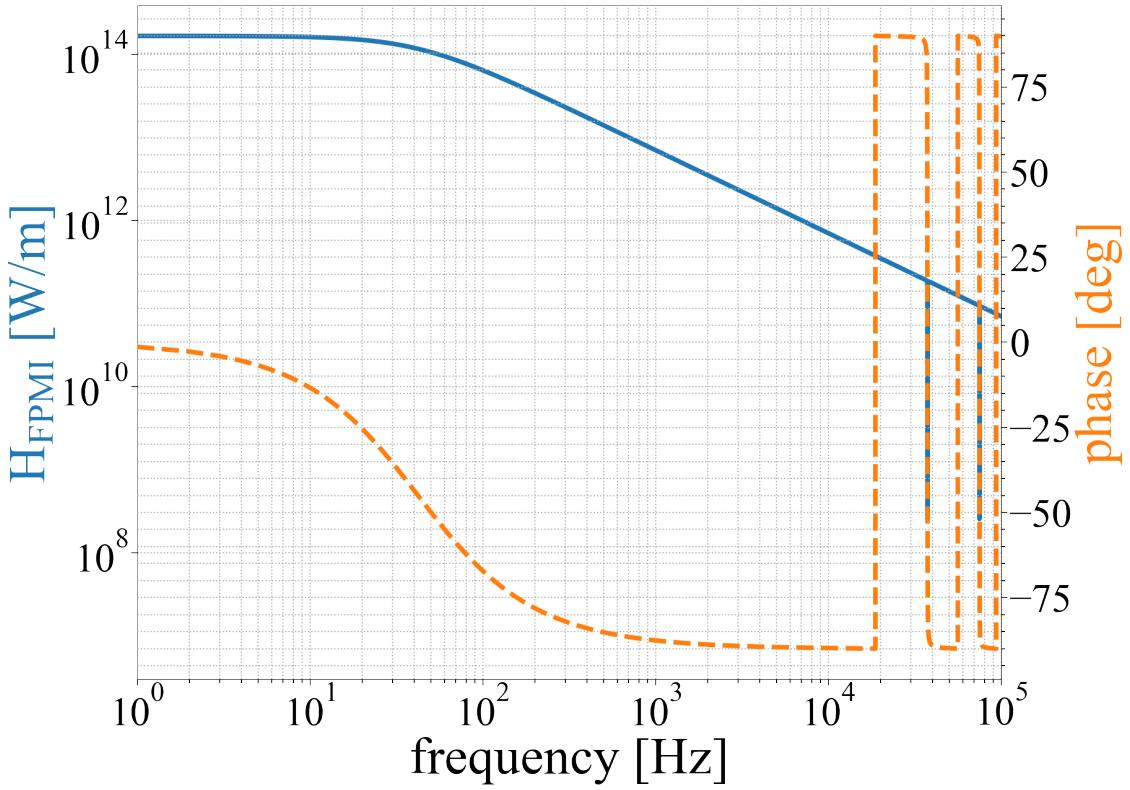
```
Signature: MICH(freq, Length, phi_0, P_in, OMEGA)
Docstring:
MICHELSON FREQEUNCY RESPONSE CALCULATOR
freq : standard (gravitational wave) frequency [Hz]
Length : Michelson ifo arm length [m]
phi_0 : static differential arm length tuning phase [rad]
P_in : input power [W]
File:      ~/Documents/git/SU/dissertation/code/ifo_configs.py
Type:      function
```

[7]:
```
H_MICH = MICH(nu, L, PHI_0, P_IN, OMEG)
```

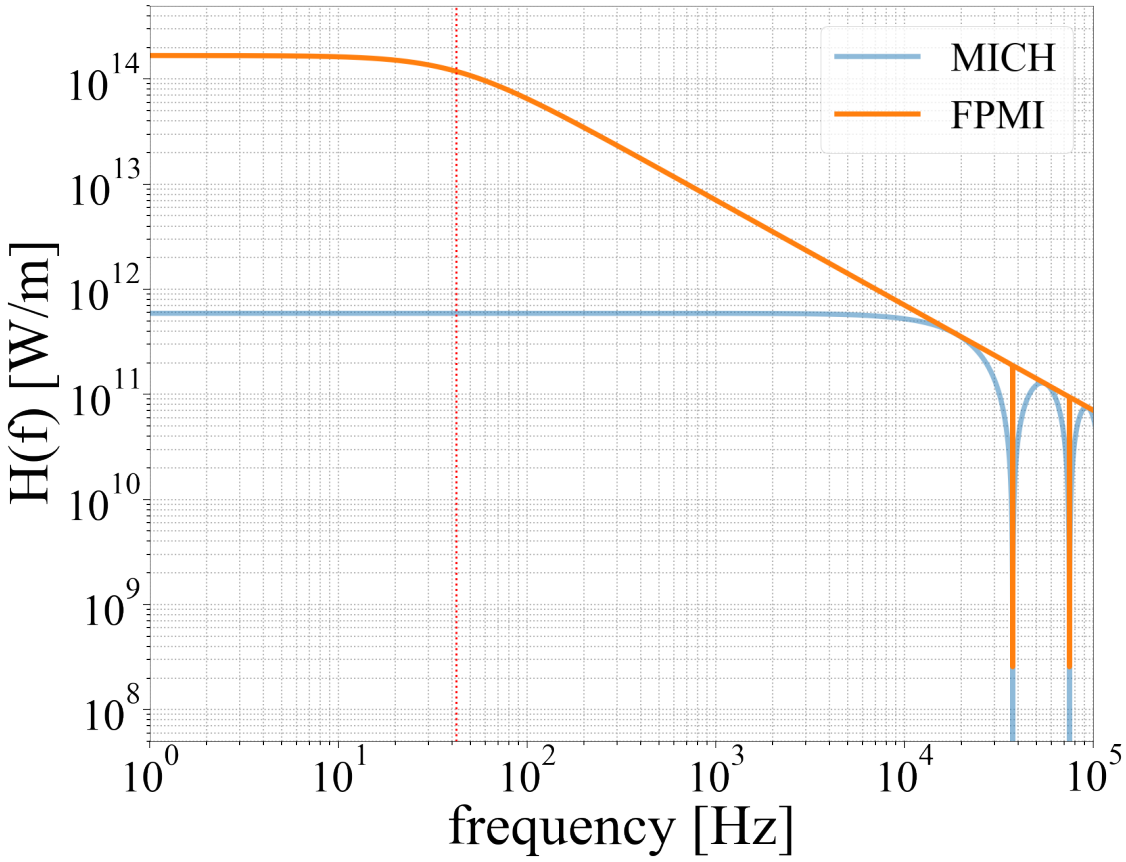
[8]:
```
fig, ax1 = plt.subplots()
ax1.set_xlabel('frequency [Hz]')
ax1.set_ylabel('H$_\mathdefault{FPMI} \; \mathdefault{ [W / m] } $ ', color='C0')
#ax1.plot(w/(FSR), F_w_cc_modsq*100)
ax1.loglog(bode_amp(H_FPMI), label='FPMI', linewidth=line_width,color='C0')
#ax1.loglog(w,H_MI_modsq, label= 'MICH', linewidth= 5)
#ax1.loglog(w,H_FPMI_LP_modsq*H_FPMI_modsq[0], label='FPMI LP', linewidth = 20.
 ↪0, alpha=0.25,color='C2')
#ax1.axvline (x=f_pole,ymin=1e-13, color='red', linestyle='dotted', linewidth=3)
ax2 = ax1.twinx()
ax2.semilogx(nu,bode_ph(H_FPMI),'--', linewidth=line_width, color='C1')
#ax2.semilogx(w,(180/np.pi)*np.arctan(np.imag(H_MI)/np.real(H_MI)), '--')
#ax2.semilogx(w,(180/np.pi)*np.arctan(np.imag(H_FPMI_LP)/np.
 ↪real(H_FPMI_LP)),linestyle='--', linewidth=20.0,dashes=(4,10),alpha=.25,␣
 ↪color='C2')
```

```
plt.xlim([1,1e5])
plt.ylabel('phase [deg]', color='C1')
#fig.savefig('../figs/INTRO/fpmi_fr.pdf', dpi=300, bbox_inches='tight')
```

[8]:
```
Text(0, 0.5, 'phase [deg]')
```

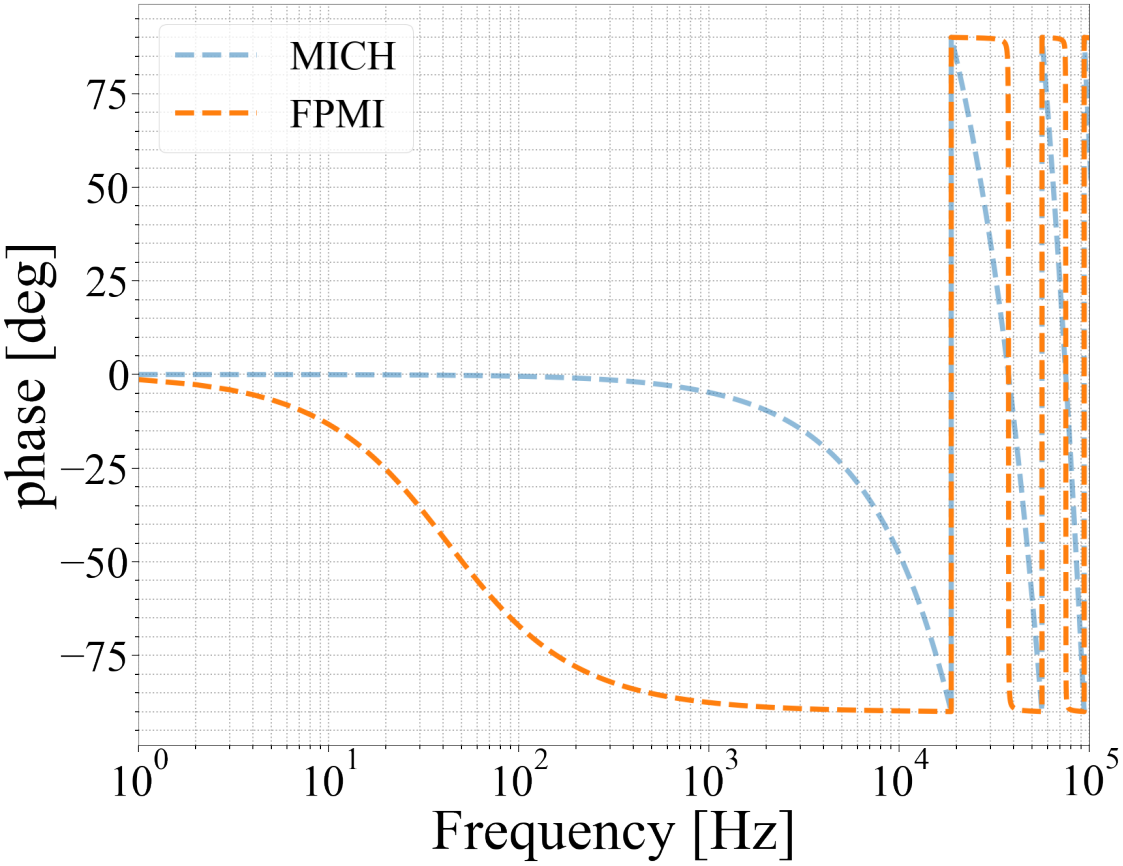


[9]:
```
plt.loglog(nu,bode_amp(H_MICH), label= 'MICH', linewidth= line_width, alpha=.5)
plt.loglog(nu,bode_amp(H_FPMI), label='FPMI', linewidth=line_width)
#plt.loglog(nu,bode_amp(H_FPMI_LP)*bode_amp(H_FPMI)[0], label='FPMI LP',␣
↪linewidth = 20.0, alpha=0.25)
plt.axvline (x=f_pole,ymin=1e-11, color='red', linestyle='dotted', linewidth=3.0)
plt.ylim([5e7, 5e14])
plt.xlim([1e0, 1e5])
#plt.grid(visible=True, which='minor', axis='y')
plt.xlabel('frequency [Hz]')
plt.ylabel('H(f) $\mathdefault{[W/m]}$')
lgd=plt.legend()
plt.savefig('../figs/INTRO/fpmi_fr.pdf', dpi=300, bbox_inches='tight')
```



You can clearly see that there is a clear increase in gain at lower frequencies (below 5000 kHz) Doesn't exactly look like Kiwamu's but close enough?

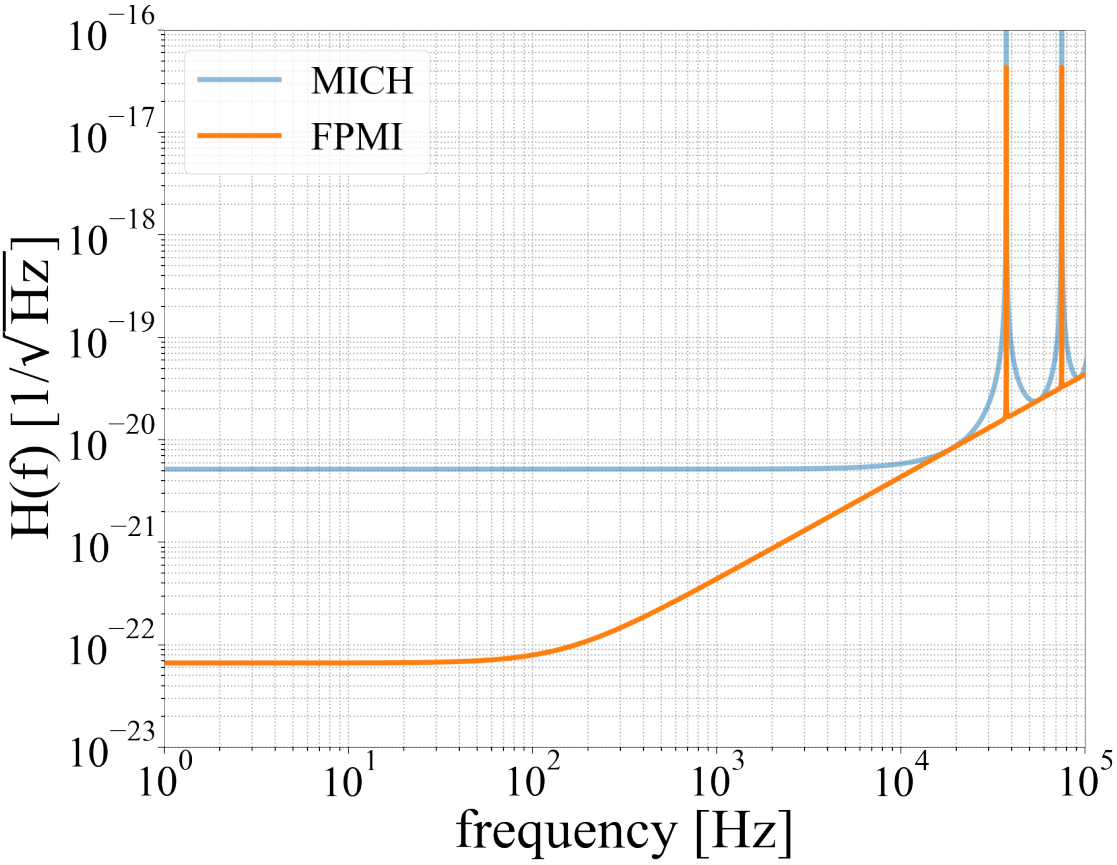
[10]:
```
plt.semilogx(nu,bode_ph(H_MICH), '--', label='MICH', linewidth= line_width,␣
↪alpha=.5)
plt.semilogx(nu,bode_ph(H_FPMI),'--', label='FPMI', linewidth= line_width)
#plt.semilogx(nu,bode_ph(H_FPMI_LP),linestyle='--', linewidth=3.0,dashes=(3,10))
plt.xlim([1,100000])
plt.ylabel('phase [deg]')
plt.xlabel('Frequency [Hz]' )
lgd=plt.legend()
```



[11]:
```
Sh_noise = N_shot(OMEG, P_IN)
```

[12]:
```
plt.loglog(nu,Sh_noise/bode_amp(H_MICH), label= 'MICH', linewidth= line_width,␣
 ↪alpha=.5)
plt.loglog(nu,Sh_noise/bode_amp(H_FPMI), label='FPMI', linewidth=line_width)
#plt.loglog(nu,Sh_noise/(bode_amp(H_FPMI_LP)*bode_amp(H_FPMI)[0]), label='FPMI␣
 ↪LP', linewidth = 20.0, alpha=0.25)
#plt.axvline (x=f_pole,ymin=1e-11, color='red', linestyle='dotted', linewidth=3)
plt.ylim([1e-23, 1e-16])
plt.xlim([1e0, 1e5])
plt.xlabel('frequency [Hz]')
plt.ylabel('H(f) $\mathdefault{[1/\sqrt{\mathdefault{Hz}}]}$')
lgd=plt.legend()
fig.savefig('../figs/INTRO/fpmi_sensi.pdf', dpi=300, bbox_inches='tight')
```

```
/Users/daniel_vander-hyde/anaconda3/envs/jupy/lib/python3.7/site-
packages/IPython/core/pylabtools.py:151: UserWarning: Creating legend with
loc="best" can be slow with large amounts of data.
  fig.canvas.print_figure(bytes_io, **kw)
```



2.0.1 *Heavily HEAVILY inspired by Kiwamu's thesis chapter on this subject (https://gwic.ligo.org/thesisprize/2012/izumi-thesis.pdf)

Next it might be useful to see the frequency dependence of just the FP alone

2.1 Vs. the delay line

Expressing part of the transfer function in terms of cavity storage time $\tau_{\rm stor}$. This helps us create a concept of a "equivalent length" or total elapsed travel time of a given phasefront in the cavity. Revisiting our Michelson response:

$$H_{\rm MI}(f_{\rm gw}) = \tau_{\rm stor}\frac{2\pi c}{\lambda}e^{-i\pi f_{\rm gw}\tau_{\rm stor}}\frac{\sin(f_{\rm gw}\tau_{\rm stor})}{f_{\rm gw}\tau_{\rm stor}}$$

[13]:
```
# Delay line storage time
N_rt = 10
tau_stor_dl = N_rt*(2*L)/cee
```

[24]:
```
# Fabry perot storage tim
#Finn = np.pi*np.sqrt(r_1*r_2)/(1-(r_1*r_2))
Finn = 3
```

```
tau_stor_fp = (L*Finn)/(cee*np.pi)
```

[15]:
```
def mich_freq_resp_(freq, L, t_s, lambd, h0):
    return ((h0*t_s*2.0*np.pi*cee)/lambd)*np.exp((-1j*np.pi*2.0*np.pi*freq)/
↪cee)*np.sin((L*2.0*np.pi*freq)/cee)/(L*2.0*np.pi*freq)
```

[ ]:
```
def mich_freq_resp_(
```

[17]:
```
H_MI_sto = mich_freq_resp_(nu, L, tau_stor_dl, 1064e-9, h_0)
```

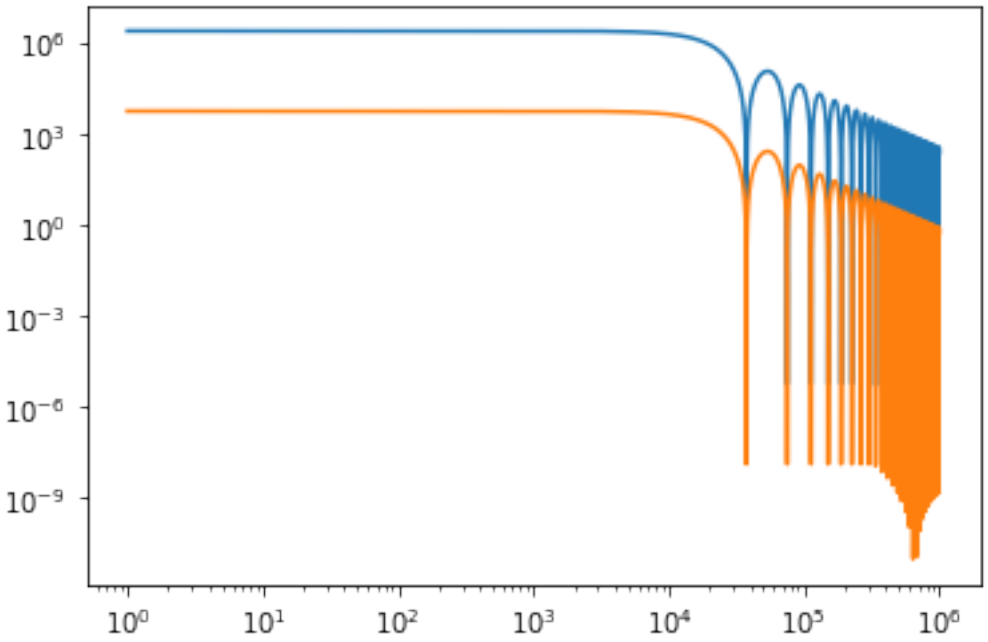
[25]:
```
H_FPMI_sto = mich_freq_resp_(nu, L, tau_stor_fp, 1064e-9, h_0)
```

[26]:
```
H_MI_sto_modsq = np.real(H_MI_sto)**2 + np.imag(H_MI_sto)**2
H_FPMI_sto_modsq = np.real(H_FPMI_sto)**2 + np.imag(H_FPMI_sto)**2
```

[27]:
```
H_MI_sto_ph = (180/np.pi)*np.arctan(np.imag(H_MI_sto)/np.real(H_MI_sto))
H_FPMI_sto_ph = (180/np.pi)*np.arctan(np.imag(H_FPMI_sto)/np.real(H_FPMI_sto))
```

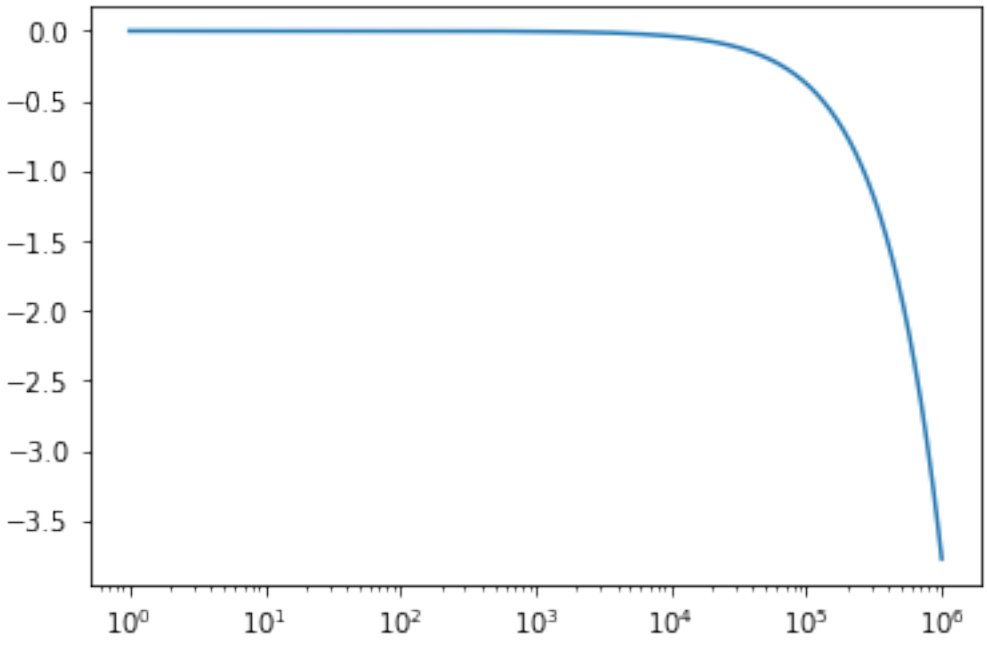
[28]:
```
plt.loglog(nu, H_MI_sto_modsq)
plt.loglog(nu, H_FPMI_sto_modsq)
```

[28]:
```
[<matplotlib.lines.Line2D at 0x177cc23d0>]
```



[20]:
```
plt.semilogx(nu, H_MI_sto_ph)
```

[20]:
```
[<matplotlib.lines.Line2D at 0x2a0c078b0>]
```


0.0
−0.5
−1.0
−1.5
−2.0
−2.5
−3.0
−3.5
10^0
10^1
10^2
10^3
10^4
10^5
10^6

[ ]: