Stability and Control of a Suspended Pendulum Fabry-Perot Cavity

Jonathan W. Perry & Daniel D. Brown

Summer 2020

1 Introduction

Simulating the optomechanics and stability of interferometers is an important step in the development of future gravitational wave detectors and allows for the detection of troublesome noise signals early in the design process, helping to further increase the sensitivity of the instrument. Software currently used to simulate gravitational wave detectors mostly consider only optical interactions with simple mechanical systems and is therefore unaware of potentially destabilizing frequencies. Instabilities can be introduced by the small oscillations from optical springs, the strength and stability of which depend on a multitude of experimental parameters. These optical springs must be considered for the detector control systems to ensure low noise and reliable operations.

Finesse is an interferometer simulation program in the frequency domain which aims to simulate not only the optical signals of an interferometer, but also the mechanical and electrical signals as well. This paper shows how the simple case of controlling an optical spring in a suspended pendulum Fabry-Perot cavity might be implemented and simulated in Finesse. More information about how to install and use Finesse 2 and Pykat [2] can be found on the website [3].

Several steps need to be taken in order to simulate the control of a suspended pendulum Fabry-Perot cavity. The mechanics of the suspension needs to be defined and combined with the radiation pressure inside the cavity. This produces the optomechanical feedback process known as optical springs, the behavior of which depends upon the propagation of the optical field. Once able to reliably determine the stability of the system, a Pound-Drever-Hall error signal then needs to be generated and included in a control loop. Each of these features must be functioning before everything can operate correctly. This paper uses a pre-alpha stage version of Finesse 3 [6] and a few bugs were encountered along the way which have been fixed or will be soon.

2 Simulation Basics

It only takes a few lines of code to simulate a simple laser and mirror setup in FINESSE. The following example uses the FINESSE scripting language `katscript` [7] to create a simple setup with a laser component `l1`, space component `s1`, and mirror component `m1`. Each component has a number of ports available, each with a set of input and/or output nodes of various types. For specifics on components and the ports available to them, see the documentation [12]. In this example, the space component `s1` is connecting the optical port of the laser `l1.p1` to one of the optical ports of the mirror `m1.p1`. A signal `sig` with frequency set by `fsig` is then generated and sent to the longitudinal force node of the mirror's mechanical port `m1.mech.F_z` effectively applying a 1 N force to the mirror every second. The motion of the mirror is then detected using an amplitude detector `ad` reading from the longitudinal position node of the mirror's mechanical port `m1.mech.z` at the signal frequency denoted by `&fsig`. The signal frequency `fsig.f` is then swept from 0.01 Hz to 1000 Hz on a log scale with 1000 steps. Finally, the model is run to determine the output at each of the detectors.

After running the simulation, the returned solution tree contains all of the information provided by the detectors in addition to a convenient `plot()` method to quickly plot the results. In this particular example, however, the detector detects no motion. This is because in order for the applied force to affect the mirror's position, the motion of the mirror needs to be defined by a mechanical force-to-motion transfer function. The next section will discuss how to determine and implement this mechanical transfer function for a suspended pendulum.

```
# create new kat model
kat = finesse.Model()

# parse katscript
kat.parse(
    """
    l l1                            # laser
    s s1 l1.p1 m1.p1                # space
    m m1                            # mirror

    fsig 1                          # signal frequency
    sgen sig m1.mech.F_z            # 1 N @ &fsig to mirror

    ad z m1.mech.z &fsig            # detect motion

    xaxis fsig.f log 1e-2 1e3 1000  # sweep frequency
    """
)

# run simulation
sol = kat.run()
```

Listing 1: Simple FINESSE simulation.

3 Implementation of the Pendulum

We want to know how a system behaves in the frequency domain when an external force $\mathcal{F}_{ext}$ is introduced. By default, the mirrors are static and unable to respond to forces. To simulate the suspension of a mirror, the equations of motion need to be transformed into the frequency domain. This is accomplished using the Laplace transform which transforms a function of time t into a complex function of frequency $s = \sigma + i\omega$.

$$\mathcal{L}(f(t))(s) = \int_0^\infty f(t)e^{-st}dt = \mathcal{F}(s) \tag{1}$$

The Laplace transform has the convenient effect of transforming the differential equation into an algebraic equation. The mechanical transfer function $H(s)$ can then be defined as the ratio of the input force $\mathcal{F}_{ext}(s)$ and the resulting motion $\mathcal{Z}(s)$.

$$H(s) = \frac{\mathcal{Z}(s)}{\mathcal{F}_{ext}(s)} \tag{2}$$

For a damped pendulum with natural frequency ω_0, quality factor Q and assuming a small angle, the mechanical transfer function as a function of complex frequency s is defined as:

$$H_{pend}(s) = \frac{1}{m(s^2 + \frac{\omega_0}{Q}s + \omega_0^2)} \tag{3}$$

The transfer function can then be written as a function of a real valued frequency Ω after substituting $s = i\omega$:

$$H_{pend}(\Omega) = -\frac{1}{m(\Omega^2 - i\frac{\omega_0}{Q}\Omega - \omega_0^2)} \tag{4}$$

The transfer function can then easily be modified for a free mass by assuming $\omega_0 = 0$:

$$H_{free}(\Omega) = -\frac{1}{m\Omega^2} \tag{5}$$

This transfer function is then used to define the force-to-z coupling in a new mechanical component which can be attached to the mirror's mechanical port using the model's `add()` method before running the simulation. The full code for the pendulum can be seen in the additional listing in Appendix A.

```
# add pendulum suspension to mirror
kat.add(
    Pendulum("m1_sus", kat.m1.mech, mass=1, w0=1, q=1000)
)
```

Listing 2: Adding the pendulum to the mirror.

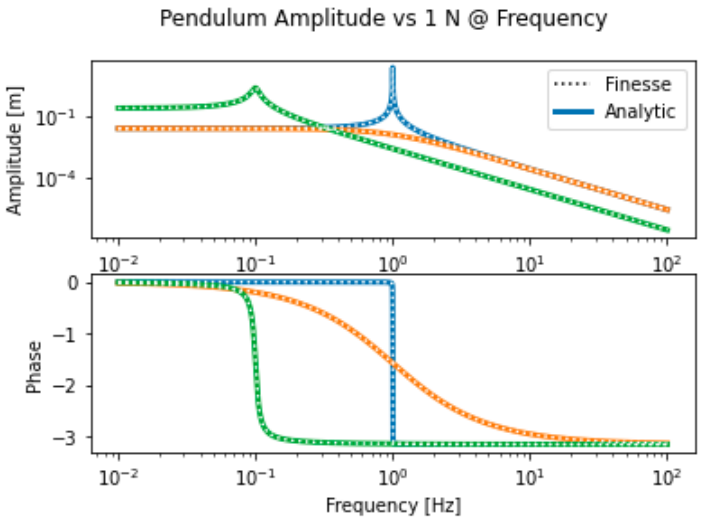

Figure 1: Plot showing the same output for analytic solution and FINESSE output when provided various inputs.

To ensure correct implementation, the analytical solution can be compared to the solution provided by FINESSE as seen in Figure 3 where the transfer functions for various pendulum parameters have been plotted.

Now that the pendulum has been implemented, the complexity of the simulation can be increased by adding a second mirror and creating a Fabry-Perot cavity.

4 Fabry-Perot Cavity

A Fabry-Perot cavity is a simple, linear, two-mirror optical cavity with a circulating power dependent on the cavity's length and/or frequency of light being circulated (Figure 4). Assuming a cavity with input power P_0 from a laser beam with wavenumber $k = 2\phi/\lambda$ incident to parallel input and end mirrors with transmissivities T_1, T_2 and reflectivities R_1, R_2, respectively, and a cavity length of L, the circulating power can be written as [1]:

$$P_{circ} = P_0 \frac{T_1 R_1}{1 + R_1 R_2 - 2\sqrt{R_1 R_2}\cos 2kL} \tag{6}$$

Due to the periodicity of light, it is often useful to break the length of the cavity into two terms; $L = z_0 + \delta z$ with z_0 defined as being "on resonance" or

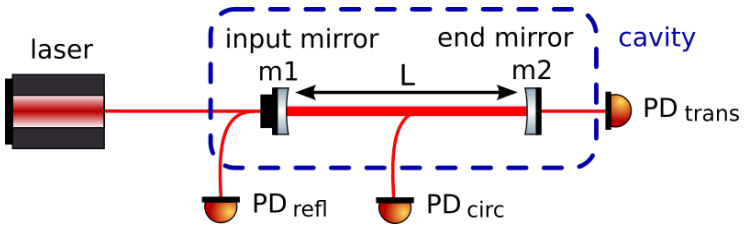

Figure 2: Diagram of a simple Fabry-Perot cavity with detectors for reflected power, circulating power, and transmitted power [8].

an integer number of wavelengths $z_0 = N\lambda = 2\pi N/k$ and δz as a displacement from resonance relative to one wavelength $\delta z = \phi/k$ where ϕ is measured in radians. The circulating power in the cavity can then be written as a function of this detuning from resonance ϕ:

$$P_{circ}(\phi) = P_0 \frac{T_1 R_1}{1 + R_1 R_2 - 2\sqrt{R_1 R_2}\cos 2\phi} \tag{7}$$

The circulating, reflected, and transmitted power can be simulated in FINESSE with the following example. Here, the laser is linked to the cavity using the `link` command which automatically connects the appropriate ports and nodes depending on signal type. Three power detectors have also been added to detect the power of the beam circulating within or leaking out through the cavity as the value of phi is swept into and out of resonance. As seen in both the output of FINESSE (Figure 4) and Equation (7) the circulating power symmetrically reaches a maximum while perfectly on resonance.

It should also be noted the circulating power is significantly higher than the input power of 1 W. If one of the mirrors is suspended, this accumulation of power leads to an optomechanical effect known as an optical spring which couples the radiation pressure force caused by the circulating power to the motion of the mirror.

5 Optomechanics

5.1 Radiation Pressure

Light carries momentum and therefore exerts a radiation pressure force when it reflects off of a surface. The radiation pressure force felt by a perfectly reflective

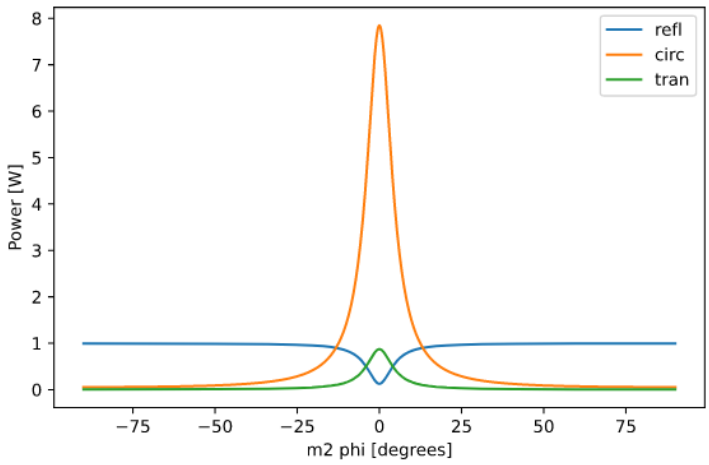

Figure 3: Output for a simple Fabry-Perot cavity with an input power of 1 W detecting the reflected, circulating, and transmitted power as a function of detuning phi.

mirror is proportional to the power of the incident beam [9].

$$F_{rp}(\Omega) = \frac{2P(\Omega)}{c} \tag{8}$$

For the end mirror of a Fabry-Perot cavity, it follows that a sufficiently high circulating power can cause a significant amount of radiation pressure to build inside the cavity. When considering the mechanics of suspended mirrors, this additional force needs to be considered.

5.2 Optical Spring

Due to the circulating power's dependence on the position of the end mirror, an optomechanical feedback process occurs which couples the fluctuations in circulating power to the motion of the mirror. The circulating power contributes to the radiation pressure force which causes displacement in the position of the mirror which in turn feeds back into the circulating power, closing the loop. This means the radiation pressure force on the mirror is position dependent. For the adiabatic situation in which the length of the cavity is sufficiently small compared to the speed of light, the definition of a spring constant, or, in this case, an optical spring constant, is the derivative of this position dependent force.

$$k_{opt} = -\frac{dF(z)}{dz} = -\frac{d}{dz}\frac{2P_{circ}(z)}{c} = \frac{-8P_0\sqrt{R_1R_2}kT_1\sin(2kz)}{c(1+R_1R_2-2\sqrt{R_1R_2}\cos(2kz))^2} \tag{9}$$

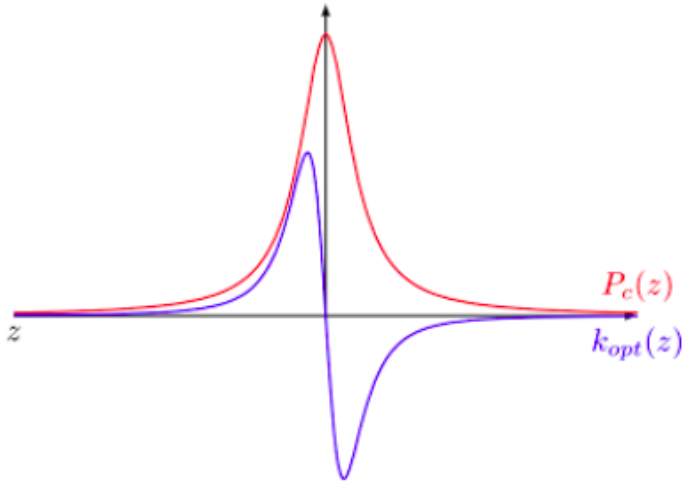

Figure 4: Optical spring constant plotting alongside the circulating power showing stable and unstable regions based upon positive and negative detunings of phi. Not to scale. [1]

Plotting the spring constant about resonance (Figure 5.2), two regions become evident. With positive detunings which lengthen the cavity there exists a restoring force $k_{opt} < 0$ which is statically stable, and with negative detunings there exists an anti-restoring force $k_{opt} > 0$ which is statically unstable. It is also worth noting there is no spring $k_{opt} = 0$ when the cavity is perfectly resonant.

For the full, non-adiabatic optical spring, the time delayed response of the cavity leads to the spring constant taking on a more complicated, complex form $\kappa(\Omega)$ showing a dynamic, oscillatory behavior dependent upon the length of the cavity L:

$$\kappa(\Omega) = -\frac{8kP_c r_1 \sin 2\phi}{c} \frac{e^{-i2\frac{\Omega}{c}L}}{1 + R_1 e^{-i4\frac{\Omega}{c}L} - 2r_1 e^{-i2\frac{\Omega}{c}L} \cos 2\phi} \tag{10}$$

The radiation pressure force $F_{rp}(\Omega)$ can then be written using this complex-valued optical spring coefficient:

$$F_{rp}(\Omega) = \kappa(\Omega)\delta z(\Omega) \tag{11}$$

A new mechanical transfer function can be obtained with the same process as Equation (4) but with the addition of the radiation pressure force:

$$H(\Omega) = -\frac{1}{m(\Omega^2 - i\frac{\omega_0}{Q}\Omega - \omega_0^2) - \kappa(\Omega)} \tag{12}$$

6 Stability

As previously mentioned, the stability of the system is dependent upon the detuning of the end mirror. If the mirror is detuned below resonance, shortening the cavity, the system is statically unstable and if it is detuned above resonance, lengthening the cavity, the system is statically stable. The time delayed response of the circulating power caused by the motion of the mirror leads to the statically stable situation being dynamically unstable and the statically unstable situation being dynamically stable. This can be understood by recognizing that the delayed response of the circulating power caused by the external force will have a tendency to be stronger when tuned above resonance than when tuned below.

6.1 Poles & Zeros

The mechanical transfer function can be further analyzed to quantitatively determine the stability of the system. The numerator and denominator of the transfer function can be factored into its roots to extract its zeros z, poles p, and gain k:

$$H(s) = k\frac{\prod_{i=1}^{N_z}(s - z_i)}{\prod_{j=1}^{N_p}(s - p_j)} \tag{13}$$

The poles will then provide information on the stability of the system. Any poles which exist on the right-hand side of the complex plane will result in the system being unstable. This is because of the positive exponent in the inverse Laplace transform (Equation 14) which causes any value of s with a positive real part to be unbounded when transforming back into the time domain:

$$\mathcal{L}^{-1}\{\mathcal{F}(s)\}(t) = \frac{1}{2\pi i}\lim_{T\to\infty}\int_{\gamma-iT}^{\gamma+iT} e^{st}\mathcal{F}(s)ds \tag{14}$$

6.2 Fitting the Transfer Function

To determine the poles and zeros of the output produced by FINESSE, a fitting algorithm must be used. The algorithm typically used to fit transfer functions is Vector Fitting, also known as `vectfit` [11], which distributes likely poles and iterates linear problems to maneuver them into place. When testing for stability with known random poles we found `vectfit` to work well but would become increasingly divergent with additional poles. Another algorithm known as `fittf` [10] was found to be more consistent for our purposes when tested against higher numbers of known poles.

$+\phi$	$16.11668693 + 0.j$	$-16.12358358 + 0.j$
$-\phi$	$0.00344833 + 16.12013231j$	$0.00344833 - 16.12013231j$

Table 1: Poles produced by fittf for positive and negative detunings of phi.

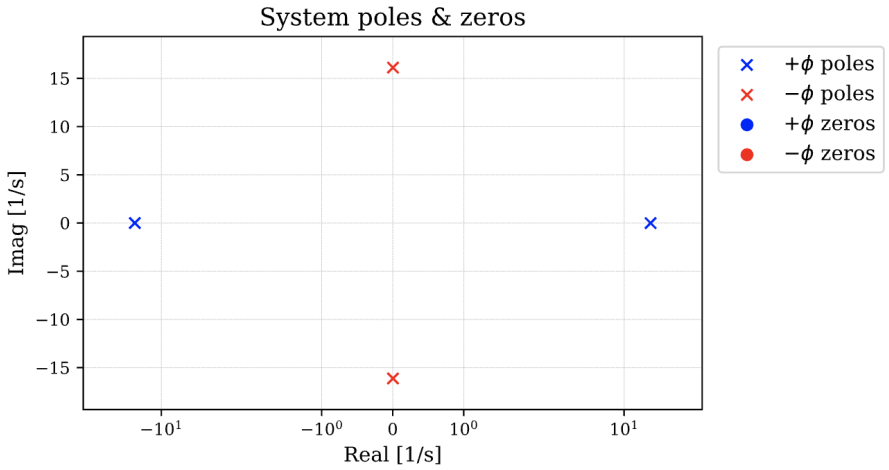

Figure 5: Poles and zeros on the complex plane for positive and negative values of phi. Poles on the right-hand side imply an unstable system.

With the algorithm returning fitted poles and zeros, the poles and zeros could then be plotted on the complex plane for positive and negative detunings of phi to see if the system is stable in that region. As seen in Figure 5, for positive values of phi the transfer function has two real-valued poles and for negative values the transfer function has one complex-valued pole. Table 1 shows the pole values which describes both regions as unstable since both contain poles on the right-hand side of the complex plane. The positive real-valued pole corresponds to the statically unstable situation while the positive real part of the complex-valued pole corresponds to the dynamically unstable situation.

The goal is now to design a controller which feeds a signal back into the system and shifts the poles to the left-hand side of the complex plane, thus stabilizing the system.

7 Simple Feedback Example

This section will go through an example of a simple feedback system, briefly explaining how one might develop a model mathematically. The following example uses a photodiode to read the power output of a laser and then feed the

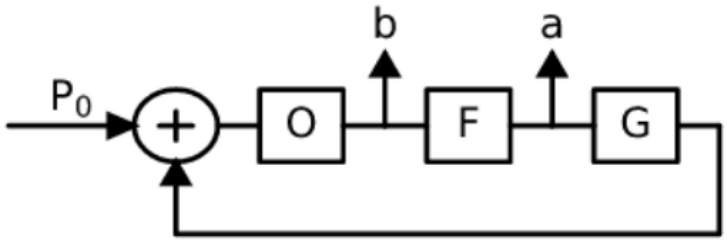

Figure 6: Block diagram representing a simple feedback system.

signal through a low-pass Butterworth filter F and back into the amplitude of the laser. The feedback loop can be toggled open or closed with an amplifier G. A generated signal is then modulated into the amplitude of the laser and detected both after the photodiode and after the filter. The Butterworth filter gain can then be adjusted to control

```
# create new kat model
kat = finesse.Model()

# parse katscript
kat.parse(
    """
    l l1                            # laser
    m m1                            # mirror
    photodiode pd1                  # photodiode
    butter F 4 lowpass 100          # lowpass Butterworth filter
    amp G gain=1                    # loop toggle

    link l1 pd1 F G l1.amp          # feedback into laser

    fsig 1                          # 1 Hz signal
    sgen sig l1.amp.i 1 0           # generate signal on amplitude

    ad b pd1.p2.o f=&fsig           # detect signal after photodiode
    ad a G.p1.i f=&fsig             # detect signal after filter

    xaxis fsig.f log 1 1e3 1000     # sweep signal frequency (Hz)
    """
)
```

Listing 3: Simple feedback example with Butterworth filter

The block diagram for this example can be seen in Figure 7. Each block represents a component and each connection represents input and output signals. Each component can be represented mathematically as a transfer function and each output can be written as the composite of the inputs:

$$b = O(P_0 + G(F(b))) \tag{15}$$

$$a = F(b) \tag{16}$$

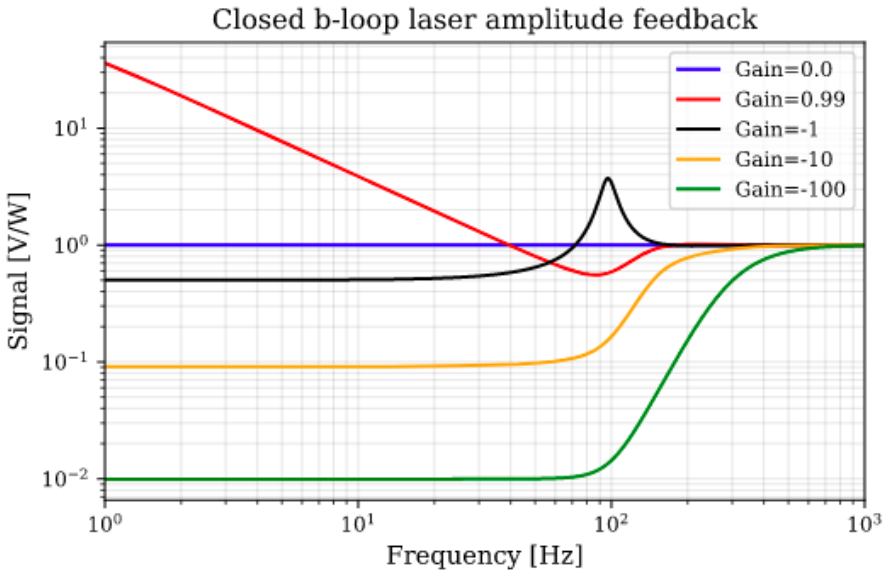

Figure 7: Plots produced by a simple feedback system for various gains which match the developed mathematical model.

By assuming all functions are linear, the output can be solved algebraically to determine the transfer function between the input P_0 and output signal:

$$b = \frac{O}{1 - OGF} P_0 \tag{17}$$

$$a = \frac{OF}{1 - OGF} P_0 \tag{18}$$

After defining transfer functions for the optical readout O, Butterworth filter F, and closing the feedback loop with $G = 1$ (Appendix B), the output can be plotted versus frequency. Figure 7 shows the resulting plots for the closed-loop output signal b provided various filter gains.

It is important to recognize that by changing the filter gain, the impact the feedback has on the stability of the system changes as well. Compared to the open-loop solution ($Gain = 0$) in which all frequencies produce the same output, certain gains ($Gain = 0.99, -1$) produce stable systems where greater negative gains produce unstable systems. It is the role of the controller to determine this gain depending on the desired output.

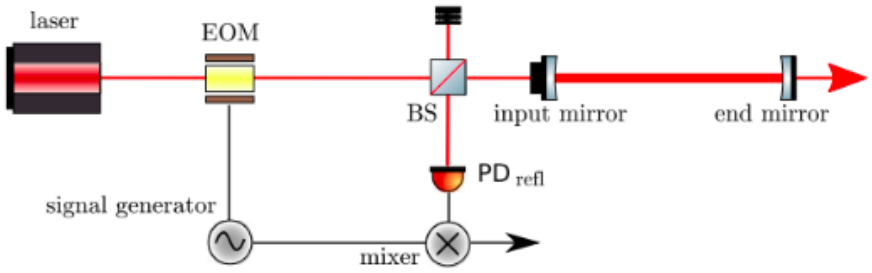

Figure 8: Diagram of the Pound-Drever-Hall technique.

8 Feedback & Control

In order to control the mirror, a error signal needs to be produced by the cavity which is capable of determining which direction a force must be applied to correct for any change in length. Simply reading the power output inside or outside of the cavity is not enough due to the symmetry about resonance as seen in Figure 4. One way of producing this error signal is with the Pound-Drever-Hall technique.

8.1 Pound-Drever-Hall Signal

The Pound-Drever-Hall technique was originally developed to lock the frequency of a laser to the length of a Fabry-Perot cavity [5], but can similarly be used to lock a cavity length to the frequency of a laser.

To produce the signal, the laser is first phase modulated prior to entering the cavity and then demodulated with a mixer using the reflected beam escaping the cavity (Figure 8). The reflected beam will be proportional to the incident beam but will experience some amount of phase shift when the length of the cavity changes. The resulting signal (Figure 9) is anti-symmetric about resonance with it being positive for negative detunings and negative for positive detunings, therefore providing an error signal appropriate for use in determining the direction in which the cavity has changed.

```
kat.parse(
    """
    l l1
    s s0 l1.p1 phasemod.p1 L=0.1
    mod phasemod 80M 0.3 1 pm      # phase modulation
    s s1 phasemod.p2 m1.p1 L=0.1

    m m1 T=0.015 L=0
    s scav m1.p2 m2.p1 L=0.2
```

```
    m m2 T=0.015 L=0

    pd1 demod m1.p1.o 80M 0         # demodulation

    xaxis m2.phi lin -50 50 500
    """
)
```

Listing 4: Example FINESSE code to produce Pound-Drever-Hall signal

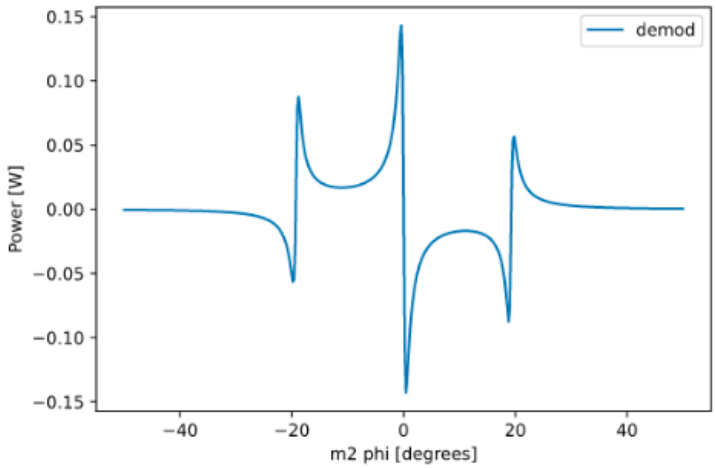

Figure 9: Example signal produced by the Pound-Drever-Hall technique.

8.2 Controlling the Cavity

Now that we have a signal capable of acting against the change in the cavity, the last step is to feed the signal through a filter to scale it before sending it to an actuator attached to the mirror which will convert the electrical signal into a force (Figure 10).

```
kat.parse(
    """
    l l1                                # laser
    mod modulator 80M 0.3               # modulator
    beam_splitter bs                    # beam splitter
    m m1 R=0.99 L=0                     # front mirror
    s scav m1.p2 m2.p1 L=0.2
    m m2 R=1 L=0                        # end mirror
    photodiode1 demod f=80M             # demodulating photodiode
    filter_zpk G [] [0] 1000 gain=1     # filter
    actuator act m2.mech.F_z            # actuator on end mirror

    # connections
```

```
    link l1 modulator bs m1            # laser->mod->bs->cavity
    link bs.p4 demod G act             # bs->demod->filter->actuator

    fsig 1                             # 1 Hz signal frequency
    sgen sig2 m2.mech.F_z amplitude=1 # apply a signal to m2

    ad z2 m2.mech.z &fsig              # detect motion of m2

    xaxis fsig.f log 1e-2 1e3 1000
    """
)
kat.add(pendulum.Pendulum("m2_sus", kat.m2.mech, w0=0))

out = kat.run()
```

Listing 5: Approximate FINESSE 3 code to stabilize a suspended pendulum Fabry-Perot cavity

NOTE: The above code does not actually stabilize the cavity and instead represents the status of the simulation before the project was able to be completed. Future work will involve determining the correct zpk filter configuration to stabilize the cavity and the paper will be updated accordingly.

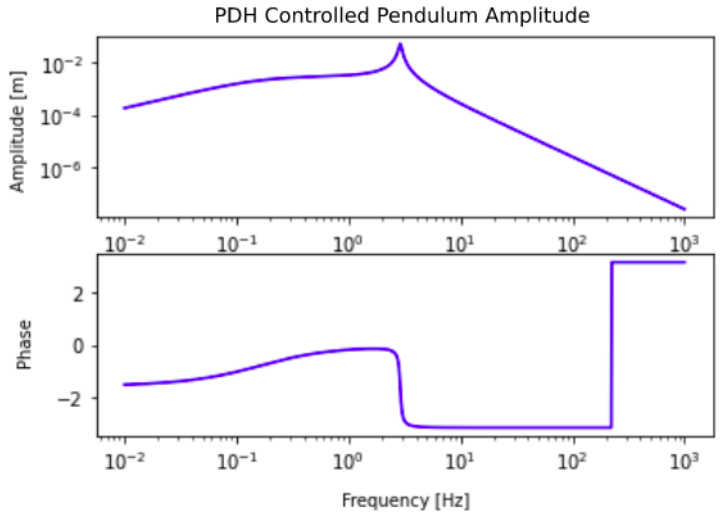

Figure 10: Pendulum amplitude after applying filtered Pound-Drever-Hall control signal.

9 Conclusion

The features added to FINESSE 3 allow for the simulation of multiple signal types and feedback loops. This includes the motion of a pendulum by defining a transfer function in the frequency domain via the Laplace transform. By suspending the end mirror of a Fabry-Perot cavity with the implementation of the pendulum, interesting and challenging optomechanical effects involving the radiation pressure from the circulation power (namely optical springs) arise. These optical springs can cause the system to become unstable and must be controlled by providing a corrective feedback signal. The Pound-Drever-Hall technique uses the reflected power leaking out of the mirror to determine the sign of this corrective signal which is then used to control the length of the cavity thus stabilizing the system.

In addition to combining the necessary steps to control the cavity, it was determined that `vectfit` was insufficient for what was needed on a larger scale and that FINESSE 3 will require something more suitable to be developed, most likely a custom fitting routine.

The greater vision of FINESSE is to provide a platform for the simulation of complicated feedback systems involving multiple types of signals which far exceed the scope of this paper. The reader is invited to learn more about FINESSE and its recent developments at the FINESSE homepage [3, 6].

10 Acknowledgements

Thank you to Daniel Brown at the University of Adelaide for his mentoring and patience while working on this project. Also, thank you to the entire FINESSE development team for their acceptance into their weekly developer calls, specifically Andreas Freise at Vrije Universiteit Amsterdam for his accessible expertise and Philip Jones for willfully helping to find errors. Additional thanks to Paul Fulda and Peter Wass at the University of Florida for their mentoring (and patience) and organizing the remote program under the added strains of COVID-19, this was surely not a simple task. And thank you to the National Science Foundation for funding this opportunity to perform international research through grants #1460803 and #1950830.

References

[1] Charlotte Bond et al. "Interferometer techniques for gravitational-wave detection". In: *Living reviews in relativity* 19.1 (2016), p. 3.

[2] Daniel D. Brown et al. *Pykat: Python package for modelling precision optical interferometers*. 2020. arXiv: 2004.06270 [astro-ph.IM].

[3] Daniel David Brown and Andreas Freise. *Finesse*. The software and source code is available at http://www.gwoptics.org/finesse. May 2014. DOI: 10.5281/zenodo.821363. URL: http://www.gwoptics.org/finesse.

[4] Stephen Butterworth et al. "On the theory of filter amplifiers". In: *Wireless Engineer* 7.6 (1930), pp. 536–541.

[5] RWP Drever et al. "Laser phase and frequency stabilization using an optical resonator". In: *Applied Physics B* 31.2 (1983), pp. 97–105.

[6] FINESSE *3 Docs*. URL: https://finesse.docs.ligo.org/finesse3/.

[7] FINESSE *Syntax Reference*. URL: http://www.gwoptics.org/finesse/reference/.

[8] *Learn Laser Interferometry with* FINESSE*: Resonance*. URL: http://www.gwoptics.org/learn/02_Plane_waves/01_Fabry_Perot_cavity/01_Resonance.html.

[9] Pierre Meystre et al. "Theory of radiation-pressure-driven interferometers". In: *JOSA B* 2.11 (1985), pp. 1830–1840.

[10] Ahmet Arda Ozdemir and Suat Gumussoy. "Transfer function estimation in system identification toolbox via vector fitting". In: *IFAC-PapersOnLine* 50.1 (2017), pp. 6232–6237.

[11] PhilReinhold. *vectfit.py*. URL: https://github.com/PhilReinhold/vectfit_python.

[12] *Using* FINESSE*: The Port and Node System*. URL: https://finesse.docs.ligo.org/finesse3/usage/nodesystem/.

A Additional Listings

A.1 Pendulum.py

```
1  class Pendulum(Connector):
2      """Suspended pendulum.
3
4      The object being suspended must have a mechanical port with
5      nodes z, pitch, and yaw and forces F_z, F_pitch, and F_yaw.
6      """
7      def __init__(self, name, mech_port, mass=1, w0=1, q=1):
8          super().__init__(name)
9          self.mass = mass
10         self.w0 = w0
11         self.q = q
12         ...
13
14         # Add motion and force nodes to mech port.
15         # Here we duplicate the already created mechanical
16         # nodes in some other connector element
17         self._add_port("mech", NodeType.MECHANICAL)
18         self.mech._add_node("z", None, mech_port.z)
19         self.mech._add_node("F_z", None, mech_port.F_z)
20         self._register_node_coupling("F_to_Z", self.mech.F_z, self.
       mech.z)
21         ...
22
23     ...
24
25     def fill(self, ws):
26         f = ws.sim.model_data.fsig
27         w = 2*np.pi*f
28         w0 = 2*np.pi*self.w0
29         with ws.sim.component_edge_fill3(
30             ws.owner_id, ws.connections.F_to_Z_idx, 0, 0, False,
31         ) as mat:
32             mat[:] = -1/(self.mass*(w**2-w0**2-w*w0/self.q*1j))
```

Listing 6: Partial code for Pendulum object with the mechanical transfer function located on line 32. Removed but necessary code replaced with '...' see source for full implementation.

B Transfer Functions for Simple Feedback

This appendix is dedicated to explaining the transfer functions for the optical readout O, Butterworth filter F, and closed-loop toggle G.

B.1 Optical Readout O

For amplitude modulation, the output field of a laser beam with modulation index m can be described by the following:

$$E = E_0 e^{i\omega_0 t}(1 - \frac{m}{2}(1 - cos(\Omega t))) \tag{19}$$

The light power P as seen by a photo detector is:

$$P = EE^* = P_0(1 - \frac{m}{2}(1 - cos(\Omega t)))^2 \tag{20}$$

Expanding this out we get:

$$P(t) = P_0(1 - m(1 - cos(\Omega t)) + \frac{m^2}{4}(1 - 2cos(\Omega t) + cos^2(\Omega t))) \tag{21}$$

By assuming a small modulation index $m \ll 1$ we get:

$$P(t) = P_0 - P_0 m - P_0 m \cos \Omega t \tag{22}$$

We're interested in the oscillating term as this tells us how a small signal propagates from the applied laser amplitude modulation to what the photodiode measures. The power fluctuations at the frequency of the injected signal Ω is then:

$$P(\Omega) = P_0 m \tag{23}$$

B.2 Butterworth Filter F

A Butterworth filter is a low-pass filter which aims to provide maximally flat frequency response in the passband below the cutoff frequency [4].

The transfer function is defined with DC gain G_0, cutoff frequency ω_c and poles s_k. How steep the signal drops off is defined by the number of poles n.

$$H(s) = \frac{G_0}{\prod_{k=1}^{n}(s - s_k)/\omega_c} \qquad k = 1, 2, 3, ..., n \tag{24}$$

Where the poles s_k are defined by:

$$s_k = \omega_c \exp \frac{i(2k + n - 1)\pi}{2n} \tag{25}$$

Here is an example implementation of the Butterworth filter transfer function using Python.

```python
## Low-pass Butterworth Filter
## fs: frequency domain
## fc: frequency cutoff
## n: order
## g: gain
def butterworth(fs, fc, n, g):
    ws, wc = 2*np.pi*np.array([fs, fc]) # angular frequencies
    s = 1.0j*ws                         # complex phase
    H = g*wc**n                         # apply gain
    for k in np.arange(1,n+1):
        H *= (s-wc*np.exp((2*k+n-1)/(2*n)*np.pi*1j))**-1
    return H
```

Listing 7: Butterworth filter implementation in Python

B.3 Closed-loop Toggle Gain G

This transfer function is either 1 to close the loop and produce a closed-loop system or 0 for an open-loop system in which there is no feedback. Figure 11 aims to illustrate this idea more clearly.

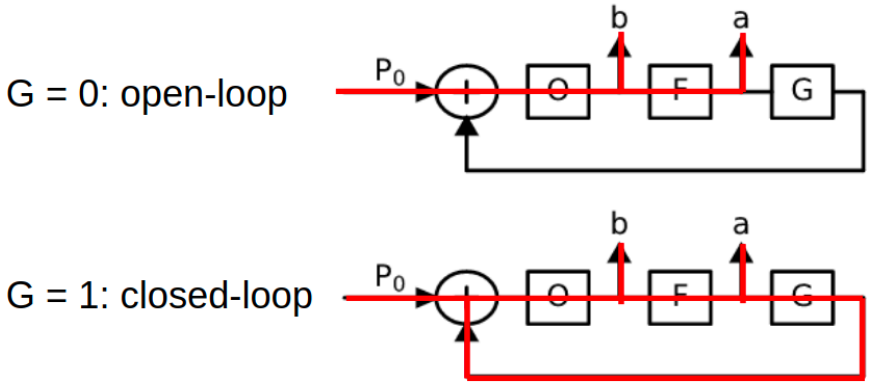

Figure 11: Illustration of open-loop versus closed-loop.