Suspension Commissioning Handbook*†

TSANG, Terrence Tak Lun‡

Last update: June 4, 2021

Contents

*Upstream url: https://github.com/gw-vis/vis-commissioning-tex

†JGWDoc: T2112780

‡ttltsang@link.cuhk.edu.hk, The Chinese University of Hong Kong

1 Introduction

1.1 About this document

This is a document with suspension commissioning tasks described in detail, as in mathematical/theoretical/code/ procedure details. We sincerely hope that this document can serve as a handbook/guideline for suspension commissioning, and as an educational document for people, who want/need to participate in suspension commission activities, to know more about the activities.

This document is dynamic, and it should be, as we get input and suggestions from people, and as technology evolves. With that said, if you have an opinion on particular tasks or methods described in this document or suspension commissioning in general, feel free to submit an issue here. Alternatively, you can contact the maintainer (Terrence for now) via any communication channels (emails, slack, whatever). We will update the document accordingly.

As we all know, people have taken different approaches for suspensions commissioning in the past. And, there were not enough communications between subgroups/people on how suspensions should be configured. With this repository, we hope to create an open environment for people to exchange ideas regarding suspension commissioning. And, we wish the document to be filled with ideas and methods that we all agree on, so all of us will commit to follow this document when participating commissioning activities.

In this document, we will mainly review some of the techniques and methods involved in suspension commissioning tasks. This document is organized as follows, the follow two subsections will be discussing the role of suspension commissioning. In particular, we will be talking about what are the expectations of the hardware aspect of the suspensions as we are given the hardware to work with. We will also be discussing what should interferometer commissioners expect from suspension commissioning. In section 2, we will be discussing the goals that we should work towards to achieve. We will be briefly discussing the background behind these goals and requirements. These are mainly taken from [1] and we strongly recommend readers to read it for detailed explanation. In section 3, we will be discussing some of the tasks involved in suspension commissioning. These tasks are foreseen to be recurring in the future so it's important to document the methods and techniques used to tackle them So, section 4 and 5 will be dedicated to explaining the methods and the ideas behind them. The former will include baseline/fallback methods, such as noise modeling, classical loopshaping, and stability criteron, that are decent enough or easy to understand, may be suboptimal but sufficient. The latter will include advanced/modern methods, such as optimization methods, $\mathcal{H}_\infty$ methods in control, and PReQua [2], for readers who seek to optimize suspension performance. We will refer to external references, give examples, and even code wherever possible.

1.2 Suspension commissioning as a stepping stone

Suspension commissioning is about a series of tasks that get us from hardware installation/upgrade to interferometer commissioning. In other words, we do whatever is necessary to the suspensions into a state where interferometer commissioners can "use" the suspensions to do interferometer alignment and locking, without tweaking the internals of the suspension systems. With that said, we should have certain expectation on the initial state of the suspension (as hardware teams handover the suspensions to us), and interferometer commissioning team should expect something from us. So, this document is about defining those expectations and how do we achieve them.

1.2.1 Hardware expectations

Before we start working on anything, the suspensions have to be ready for commissioning. That is, we shouldn't be tweaking the hardware of the suspensions All tasks that needed to be done at the hardware level, is by definition, not our tasks. We should expect the suspensions to be "healthy", as in, the dynamics of the suspensions are exactly what one would expect, and the sensors and actuators should be fully functional and calibrated. At the end of the hardware installation process, there's an acceptance test, which every suspension should pass. This should, in principle, make sure that the suspensions are "ready". But, if we observe anything out of the normal, we should check the system's vitals and explain what's wrong to hardware experts.

There are many ways that hardware fault can leave you head-scratching for a long period of time. And, sometimes they can be hard to catch, especially when there's no testing pipeline in KAGRA. I learnt this from painful experience. I was asked to work on the optical lever on the signal recycling mirror (SR) suspensions. In particular, I was working on something we called "diagonalization" where we align the sensor signals to some desirable basis. I worked out the math but I can't seem to get the signals decoupled, regardless of the fact that the same method worked for other optical lever. At the end, we discovered that one sensor (QPD segment) has a higher gain than the others and that was simple a hardware fault that has haunted me and the Type-B team for months [3, 4]. Also, there's nothing that prevents the suspensions to go "unhealthy" even if it was perfectly fine. Systems can degrade and degradation can happen. For example, one of the PRM magnet came off during commissioning [5, 6]. Fortunately, commissioners were able to pin point the source of error and asked the vibration isolation system (VIS) team to check it. The lesson learnt here, is that, we shouldn't trust what was given to us, even if the suspensions were given to us by Albert Einstein himself. People make mistakes, and there's no exception. We are scientist, and we need to be skeptical[1]. So, think carefully if any abnormality is due to hardware fault. And, do consult hardware experts and demand a hardware fix when you got stuck.

Here is a non-exhaustive list of problems that I can think of[2], that can only be fixed at a hardware level, but can limit the outcome of suspension commissioning, and demand fixes:

- Suspension in contact with components near it (e.g. Touching a cable). This will usually result in a very low Q-factor in certain vibration modes.
- Sensor not within good operation range. This will pose a limit in the amount of actuation during feedback control. This can also result in wrong sensors/actuator diagonalization. And, this can lead to higher sensor noise.
- Stepper motors stuck. Static load on actuators cannot be offloaded to stepper motors and hence limiting the actuation force during feedback control.
- Suspension cannot be moved in full range (e.g. the BF stage of the SRM [7]).
- Abnormal coupling (e.g. Same signal but with phase difference as reported in [2]).
- Very different coil actuation efficiency, indicating a non-functional actuator.

[1] If there's anything I (Terrence) learnt from Rana, it's this.
[2] Please submit an issue if you can think of more that are worth mentioning.

Fixing these problems are not within the scope of suspension commissioning. We are suppose to work with a healthy system so we get maximum potential out of the hardware. We should ask for a fix, before proceeding. If it can't be fixed, we should discuss possible limitations and possible workarounds together.

1.2.2 What should interferometer commissioners expect?

We cannot guarantee that the outcome of suspension commission are exactly needed for interferometer commissioning. There are few reasons: 1) There are too many uncertainties. 2) Various topics regarding the suspensions are not well-studied, and are still being researched. 3) There are some performances that cannot be confirmed until we have the interferometers. And most importantly, 4) We don't exactly know what realistically interferometer commissioning needs, except we can only refer to theoretical studies like [1], which can be outdated and not really aligned with the real deal here in KAGRA. Therefore, regarding the last point, we strongly encourage interferometer commissioners to cooperate by sharing information and suggestions.

In particular, these information will be very useful:

- What is the realistic displacement level requirement, in RMS and peak, for the optics?
- At what optics displacement level did you observe lock-loss?
- At what seismic noise level did lock-loss occur?
- Is lock-loss correlated with the seismic noise level?
- How would you like to control the suspensions?
- What are the entry points? Do you want to control the suspensions directly from the setpoints? Or do you want to control the suspension using directly from the actuation signals?
- Why did you run into actuation saturation? Do you want the signals to be offloaded to higher stages, like the preisolator?
- What are some problematic cross-couplings (e.g. length-to-pitch), and in what condition you would like them to be decoupled?

While we cannot provide any guarantee, we can and should provide warranty. If the suspensions are failing, it is suspension commissioners' job to fix or improve it. Interferometer commissioners should **not** be tackling suspension tasks like, internal damping of the resonances, sensors decoupling, or seismic noise attenuation. And, interferometer commissioners should expect the suspensions to behave like a blackbox (as you don't need to know what's happening internally), and as some sort of servo (as it steers the optics). If the suspensions become problematic and is hindering interferometer commissioning, interferometer commissioners should work with suspension commissioners to tackle the problem. Previously, such kind of interface was clearly lacking[3] and we hope this can be improved in the future. We need good communication for KAGRA to work. So, here we should say that **suspension commissioning is not a "once done and for all" process**. Instead, it is something that should be recursively done, as we get "complaints" from interferometer commissioners during interferometer commissioning.

2 Goal

Qualitatively speaking, the goal of suspension commissioning is simply to bring the suspensions into a state where interferometer commissioners can use the suspensions, via defined entry points, to steer the mirrors and make an interferometer, which is sensitive enough to be a gravitational wave detectors. Althrough, as mentioned, there's no realistic requirements, which are derived from previous experience, we adopt theoretical requirements derived from [1]. In particular, in order for KAGRA to function as an gravitational wave detector, the suspensions have

[3] As Nakano-san is a superman who tackles every problem on his own.

to satisfied 2 types of requirements, displacement noise requirement, residual motion requirement. In this section, we will briefly discuss the requirements, but will not go into the details, as this is not the scope of this document. Readers are recommended to refer to the external references cited.

2.1 Displacement noise requirement

Ground-based gravitational wave detectors have a detection band starting from 10 Hz. The detectable effect of gravitational waves comes in the form of strain, which loosely speaking, is the fractional change in length. To detect gravitational waves, we simply use an L-shape interferometer and measure the change in the arm lengths in two directions. We expect targeted gravitational waves to have a peak amplitude of 10^{-21}. While the arm lengths of the interferometer are in the order of 10^3 m, the change in arm lengths caused by targeted gravitational wave is expected to be in the order of $10^{-21} \times 10^3$ m $= 10^{-18}$ m, which is a very tiny amount of displacement. Effectively, this means that the test masses (TM), i.e. the end mirrors, must have a displacement level lower than that. Giving a safety factor of 10, i.e. signal-to-noise ratio (SNR) of around 100, this converts a displacement noise requirement of 10^{-19} m at 10 Hz, roughly speaking. Figure. 1 shows the displacement noise requirement of the optics including the test mass (TM), beamsplitter (BS), signal recycling mirror (SRM), and power recycling mirror (PRM) above 10 Hz. As can be seen, the requirements of optics other than the TMs are much lower. The displacement noise

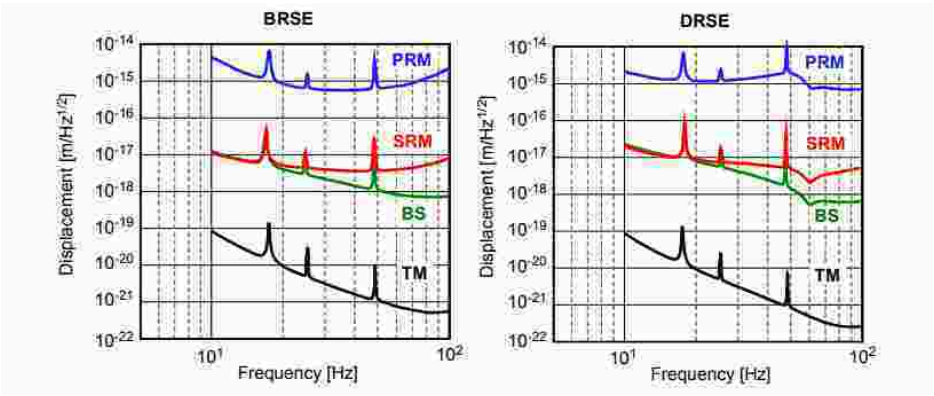

Figure 1: Displacement noise requirements of the test mass (TM) (black), beamsplitter (BS) (green), signal recycling mirror (SRM) (red), and power recycling mirror (PRM) (blue), under boardband resonant sideband extraction configuration (BRSE) (Left) and detuned resonant sideband extraction scheme (DRSE) (Right). Retrieved from [1].

level of the BS and the SRM is at 10^{-17} m, while that of the PRM is at 10^{-15} m. Table .1 summarizes the displacement noise of various optics at 10 Hz. Therefore, the partial goal of suspension commissioning is to make sure these displacement noise levels requirement are met. The suspensions of the optics are designed specifically to

	Displacement level ($\mathrm{m}/\sqrt{\mathrm{Hz}}$)
TM	1×10^{-19}
BS	1×10^{-17}
PRM	2×10^{-15}
PR2, PR3	1×10^{-15}
SRM	1×10^{-17}
SR2, SR3	5×10^{-18}

Table 1: Summary of longitudinal displacement noise level (at 10 Hz) of the optics including the folder mirrors of the recycling cavity (PR2, PR3, SR2, SR3). Retrieved from [1].

attenuate the seismic noise at 10 Hz to the displacement level specified in table. 1 via passive isolation. Therefore, the suspensions intrinsically satisfied these requirements already without any tweaking. However, the main concern here is not about passive isolation. Instead, it's about active isolation/feedback control, as we need to keep the control noise below these displacement level, while maintaining certain disturbance rejection capability. This brings us to the other requirement: residual motion.

2.2 Residual motion requirement

In previous section, we discussed what are the requirements for KAGRA to be sensitive enough to become a gravitational wave detector. Here, we will discuss the requirements for KAGRA to be an interferometric gravitational wave detector. In particular, we will discuss velocity, angular fluctuation, and displacement level requirement. Again, here we only briefly mention the rationale behind such requirements and readers are strongly recommended to read [1] for detailed explanations.

2.2.1 Residual velocity requirement

In order for KAGRA to become an interferometric gravitational wave detector, the main optics must be manipulated to form various optical cavities, where feedback control is used to "lock" two mirrors at relatively stable separations. The technique used is called Pound-Drever-Hall (PDH) technique [8]. To use this technique, the separation between two optics stay within the operating range where the control signal for using PDH technique becomes linear. In [1], a velocity requirement of the is derived by considering the maximum actuation power of the TM coils. Consider a case where the optics are moving towards the operating range. As the optics enter the range, the actuators applies maximum force on the optics, causing the optics to decelerate maximally. If the optics are put to halt before they exit the operating range, then lock-acquisition is achieved. This only happens when the velocity of the optics is lower than a certain threshold, which we define as a requirement. The velocity requirements the main optics are derived in [1] and is summarized in table. 2.

	Velocity requirement (RMS) (μm/s)
TM, BS, SR	0.5
PR	2

Table 2: Summary of main optics' velocity requirement. Retrieved from [1].

2.2.2 Residual angular displacement requirement

Now, I don't really understand the rationale behind these angular fluctuation requirements[4] and it seems that different sources suggests different things. I will simply report the requirements from some sources.

In [1], it was reported that the RMS angles of the mirrors should not produce beam spot fluctuation on the optics by larger than an RMS value of 1 mm. Using the beam paths as the lever arm, the angle fluctuation requirements are calculated and shown in table. 3. However, in another source regarding beam jittering at LIGO [11], it's mentioned

	Angular fluctuation RMS (μrad)
TM	0.2[1, 9] / 0.01[10, 11]
BS	4
PRM	45
PR2	20
PR3	2
SRM	25
SR2	10
SR3	1

Table 3: Angular fluctuation requirements of the optics. Retrieved from [1, 9] unless otherwise specified.

that the angular fluctuation requirement of the test masses at LIGO is 10^{-8} rad. The same value is also used to derive the wavefront sensor sensitivity requirement at KAGRA [10]. It's worth noting that both of these sources

[4]Please help me to write this section if you know the correction explanation. As far as I know, the angular fluctuation requirement comes from the fact that the interferometer needs to be aligned and that the beam spots are within good range around the center of the optics. But, I don't really know how people got those numbers.

were also cited in [1]. So unless there's extra comment regarding this, let's just set the requirement for the TM to the lower one (10^{-8} rad) as a worst case scenario[5].

2.2.3 Longitudinal displacement level requirement

At last, the longitudinal displacement requirement was derived also from the maximum actuation power of the force coils at the TM stage. Doing so, we assume that only these actuators are used during lock acquisition, but this may not be the case, so here again we are assuming a worst case scenario. Table. 4 summarizes the displacement RMS value requirement of the main optics in KAGRA. This concludes the residual motion requirements for the

	Displacement RMS requirement (μm)
TM	0.01
BS	3.3
SRM	3.3
SR2, SR3	1.6
PRM	560
PR2, PR3	280

Table 4: Longitudinal displacement RMS requirement of the optics. Retrieved from [1].

core optics. So, the other half of the goal is to actively control the suspensions so requirements as shown in Table. 2, 3, and 4 are met, while not violating noise specification as shown in Table. 1. In the following sections, we shall discuss/review some of the methods and techniques that can be used to achieve these goals. We will also discuss some ways that we used to verify the satisfaction these requirements.

2.3 Miscellaneous

This section mainly refers to artificial requirements that are set in [12].

2.3.1 Seismic noise

As a vibration isolation system, the main goal of the suspensions is to isolate the sole external disturbance, that is, the seismic disturbance. Therefore, the optics must satisfy the displacement noise and residual motion requirements under the influence of a background seismic noise. However, the residual motion of the optics vary with seismic noise level, which means that the performance of the suspension evaluated at a certain moment may not be the same at the other, as the seismic noise is dynamic. So, we purpose an additional constraint here. We require that the displacement noise and residual motion requirements to be satisfied under the influence of a disturbance equivalent to a background seismic noise at the 90^{th} percentile level in the winter season. The choice of this seismic noise meant to be conservative as it's considered a very violent disturbance. To give some perspective of how the seismic noise level is, the seismic noise at the 90^{th} percentile in KAGRA is shown in Fig. 2 [13]. As is mentioned, this is a conservative requirement, and it can be rare to observe such huge disturbance. This means that it can be hard to test the suspension performance naturally. To this end, as we shall see in later sections [ref later sections], we purpose to inject, using the actuators, artificial time series that has the same amplitude spectral density (ASD) of the measured seismic noise.

2.3.2 Guardian

Guardian is an automation system used in advanced LIGO (aLIGO) [14]. It's a platform where operators can program, using Python, certain operations that takes the suspensions from one defined states to another via certain

[5]There's a chance that we cannot confirm that we actually satisfied this requirement at the end of suspension commissioning as there's no sensors sensitive enough to measure such small level of fluctuation.Let's hope that our optical levers are sensitive enough

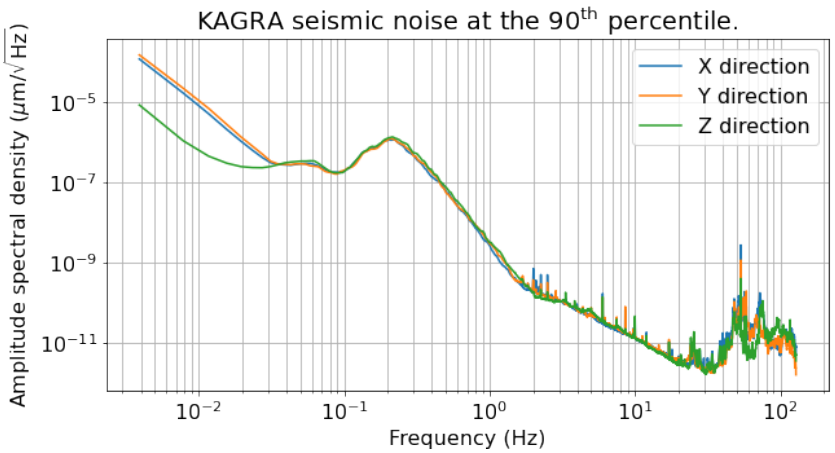

Figure 2: The amplitude spectral density of seismic noise in KAGRA at 90$^{\text{th}}$ percentile in x (blue), y (yellow), and z (green) direction. Data retrieved from [13].

defined paths. The details of Guardian will not be discussed here, readers are strongly recommended to read the "living" document from aLIGO [14]. We will only briefly introduce what we're trying to achieve with Guardian and what people should expect. Here, we will relay information given in [12, 15]. Fig. 3 shows the (proposed?) state-

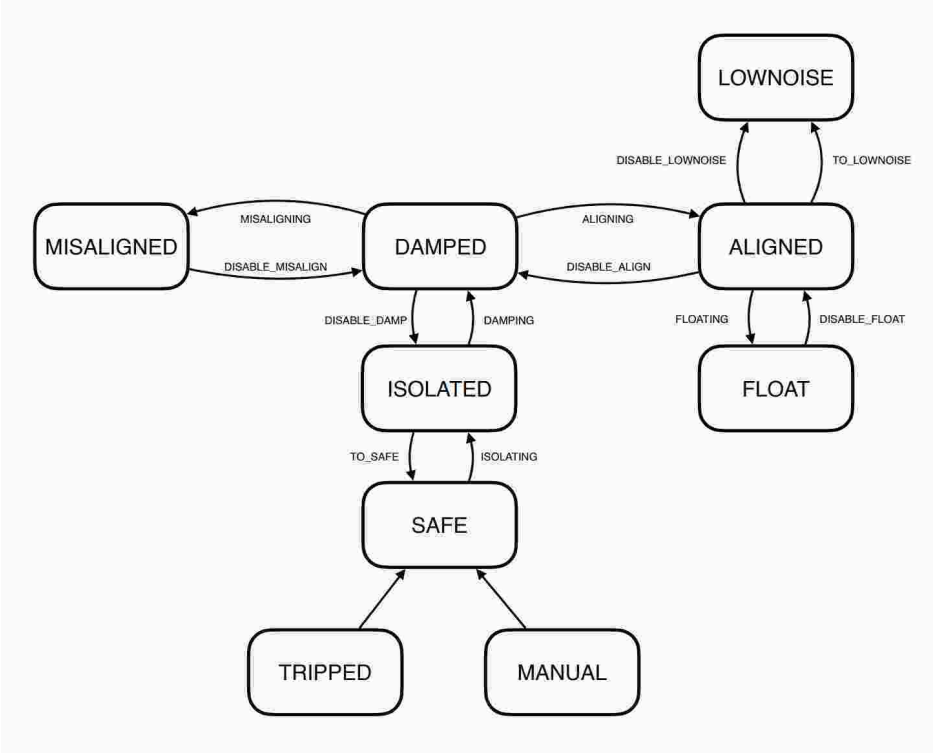

Figure 3: State-transition diagram of the (proposed?) VIS Guardian. Retrieved from [15]

transition diagram of the VIS Guardian. As can be seen, there are 9 Guardian states, namely TRIPPED, MANUAL, SAFE, ISOLATED, DAMPED, MISALIGNED, ALIGNED, FLOATED, and LOWNOISE. Here, description in [15] is lacking so we will be further elaborating the behavior of these states.

In the TRIPPED state, as is indicated by the name of the state, the suspension is tripped. This is triggered by a builtin watchdog system in the real-time model that monitors for abnormal behaviors. The suspension can "jump" (In Guardian language) into this state from every other state. This happens when the watchdog's tripped.

As for the MANUAL state, not sure what that means. But my guess is that this is a state with the master switch turned on, so actuation signals can go through for measurement and diagnostic purposes.

In the SAFE state, actuation signals cannot go to the actuators, essentially immobilizing the suspension.

In the ISOLATED state, active isolation, usually at the stage closest to the ground (likely preisolator/inverted pendulum), is engaged. The controls systems are engaged such that two things are achieved: 1) Coarsest alignment with integration action, and 2) Seismic noise suppression. The controls at the higher stage will bring lower stages into operating point (e.g. centering the optical levers), but will perturb lower stage resonances via control noise injection.

In the DAMPED state, controls at the lower stages will be engage to suppress the these resonances. At this point, the suspension should be able to satisfy the residual motion requirements as stated in Sec. 2.2.

In the ALIGNED state, the optics is steered to be coarsely aligned and is ready to be handed over to inferometer control. Distinct from the ALIGN state, In the MISALIGN state, the optics is steered away from the aligned position.

At last, at the LOWNOISE state, controls that may compromise the detector sensitivity will be shut off or replaced by a lower noise version. At this point, the suspension should satisfy the displacement noise requirements as stated in Sec. 2.1.

3 Suspension Commissioning Tasks

In this section, we will describe what are some specific tasks needed to be done in order to achieve the aforementioned goals in Sec. 2. In next section, Sec. 4, we will describe some of the mathematical details on how to complete these tasks, and how to evaluate the performance of the suspensions. In the section after the next, Sec. 5, we will review some methods that has been proposed but are not considered baseline. Here, we encourage readers to choose and implement the methods themselves. There's no best method. And, we should emphasize that it's not important to use the same methods for all suspensions, but rather, to evaluate all suspensions using the same evaluation methods. As long as we have a consistent evaluation scheme that ensures the requirements are met, that's it needs for KAGRA to work.

3.1 List of tasks

We will provide a list of tasks here in this section. Further elaboration will be provided in the next. These tasks are defined with the assumption that we have healthy hardware and we have sensors (and actuators) calibrated, and they are listed in order.

1. (If not done already) Sensor noise measurement (and modeling).
2. (If not done already) Seismic noise measurement (and modeling). Use a worst-case spectrum, e.g. 90^{th} percentile seismic noise in the winter season.
3. Control Matrices
   (a) Install initial sensing matrices and actuation matrices from first principles.
   (b) Modify sensing matrices or apply "diagonalization"/"decoupling" matrices so the readout is mapped to the desired basis (Cartesian coordinate + Euler angles).
   (c) (Optional) Modify actuation matrices or apply "diagonalization"/"decouping" matrices so the actuations are also in the desired basis.
4. Inter-calibration of sensors.

5. Further sensor noise reduction tasks, e.g. sensor fusion and sensor correction. Redo step 1.
6. Transfer function (TF) measurements and modeling. Do for
   - Diagonal actuation TFs in a stage-by-stage basis,
   - (Optional) Cross actuation TFs in a stage-by-stage basis, and
   - TFs from each displacement to optics/test mass (TM) degrees of freedom (DoF).
7. From the highest stage (closest to the ground) to lowest stage, design the control filter and do the following
   (a) Predict the closed-loop displacement level and
   (b) estimate the residual motion and displacement noise contribution to the optics using
      - the seismic noise measurement/open-loop displacement levels from step 2 or step 7f,
      - sensor noise measurement from step 1, and
      - the displacement-to-optics displacements transfer functions from step 6.
   (c) Check stability using stability critera (Nyquist plot and stability margins.) and transfer functions from step 6.
   (d) If any of the above failed, tune the control filter.
   (e) Install the control filters and close the loop.
   (f) Measure open-loop displacement levels of the next stage (Keep the controls at upper stage engaged.) and move on the next stage.
   (g) Repeat step 7 until all local control-loops at all stages are closed.
8. Measure the residual motion using the sensors at the optics stage as an out-of-loop sensor (Do not engage the control-loops that use the optics' sensors). If this fails, find the problematic stage and redo all controllers starting from there.
   - We can measure the spectrum directly, given that the sensors at the optics are not coupled to the ground motion and that the ground motion is close to what we are targeting.
   - Alternatively, we can inject a simulated time series equivalent to the target seismic noise at the highest stage to simulate the worst-case scenario.
9. (If there exists an interferometer) Measure actuation to differential arm length control signal (DARM) transfer function and measure the displacement noise. If this fails, find the problematic stage and redo all controllers starting from there.

3.2 Further elaboration

In this section we will discuss why do we need to complete the tasks as stated in Sec. 3.1 and why are they listed in that order. We will be discussing in the language of control theory. If you're not familiar with the topic, please refer to introductory textbooks such as [16] and [17]. Please also refer to appendix A for definitions and properties related to spectral densities.

Consider the control diagram as shown in Fig. 4. $R(s)$, $D(s)$, and $N(s)$ are the external inputs, and they are the setpoint, disturbance, and noise respectively. $R(s)$ is usually a DC setpoint purposed for coarse alignment. It has no non-zero frequency content so we'll just mention its existence here and will drop it in further discussions. $D(s)$ can be any arbitrary disturbance to that particular degree of freedom, that directly goes to the output $X(s)$ It can be thought as the output $X(s)$ when there's no actuation, and can be defined as

$$D(s) \equiv \lim_{K(s)\to 0} X(s; K(s))\,. \tag{3.1}$$

$X(s)$ is the output and can be thought as the actual displacement, not measured, of the DoF we aimed to control. This can be any of the arbitrary displacement at any stage of the suspension. $N(s)$ is the sensing noise, not exactly the sensor's intrinsic noise, but is defined as

$$N(s) \equiv \lim_{K(s)\to \infty} -X(s; K(s))\,. \tag{3.2}$$

As can be seen from Eqn. (3.1) and (3.2), the external disturbance $D(s)$ and sensing noise $N(s)$ can be thought as the limitation of the control performance. While $D(s)$ is some random process that we cannot minimize, $N(s)$ can be reduced by methods such as sensor fusion and sensor correction. Therefore, it's desirable to complete these additional tasks before proceeding to designing the controllers.

Back to Fig. 4, $U(s)$ is the actuation signal. $P(s)$ is the actuation transfer function of this particular degree of freedom, and is defined as

$$P(s) \equiv \frac{X(s)}{U(s)}\,. \tag{3.3}$$

At this point it because tedious to specify everything as a function of s, so will simply omit to write the s dependencies and assume all variables with capital letters are all functions of s unless otherwise specified.

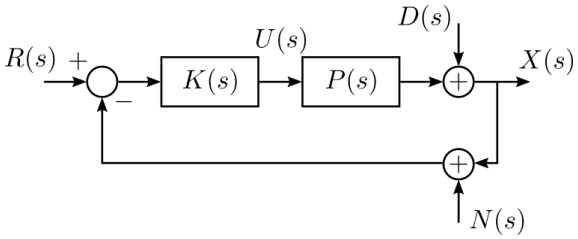

Figure 4: Typical control block diagram of a single degree of freedom. $R(s)$: reference/setpoint, $D(s)$: external disturbance, $X(s)$: displacement, $N(s)$ sensing noise, $K(s)$: controller, $P(s)$: actuation transfer function.

Here, it's important to note is that the suspension is not automatically configured in such a way Fig. 4 is valid. This is because the sensors on a stage are not necessarily aligned to the DoFs that we are interested in. The location of the sensors are usually well known. With a geometric matric transformation, we will be able to obtain an initial "map" or sensing matrix that maps the sensing readouts to the DoFs that we are interested in. See [18, 19, 15, 20] for references. However, this initial sensing matrix gives good calibration of the DoFs but might not be perfect so residual cross-couplings might still exist. This is done by modifying the sensing matrix / applying a second matrix along the signal path, i.e. "diagonalization". Similarly, the actuators might not be aligned to the desired DoFs and an actuation matrix is needed to align that. With the sensors and actuators aligned, Fig. 4 becomes valid for individual DoFs on a stage.

Now, in Fig. 4, X represents the displacement of in an arbitrary DoF of an arbitrary stage. As shown in Fig. 5, there exists paths $P_{\mathrm{X}\to\mathrm{X_{TM}}}$, such that the displacement at any stage is transferred to the optics displacement X_{TM}. In reality, since there are many DoFs, there exists many paths like Fig. 5, as shown in Fig. 6. Hence, without the

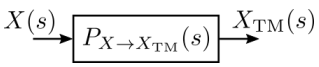

Figure 5: Displacement to optics' displacement path

optics local control, the amplitude spectral density of the optics displacement $\hat{X}_{\mathrm{TM}}(f)$ is a quadrature sum of all these contribution, i.e.

$$\hat{X}_{\mathrm{TM}}(f) = \left[\sum_i \left|P_{X_i\to X_{\mathrm{TM}}}\right|^2 \hat{X}_i(f)^2\right]^{\frac{1}{2}}\,, \tag{3.4}$$

where X_i is the displacement of the i^{th} DoF, $\hat{X}_i(f)$ is the amplitude spectral density of X_i and $P_{X_i\to X_{\mathrm{TM}}}$ the transfer function from X_i the optics displacement X_{TM}.

Now, individual contributions from different displacements in Eqn. (3.4) can vary by orders of magnitude, and so they are often viewed in logarithmic scale. And because of this, the Eqn. (3.4) can be approximated as

$$\hat{X}_{\mathrm{TM}}(f) \approx \max_i \left(\left|P_{X_i\to X_{\mathrm{TM}}}\right| \hat{X}_i(f)\right)\,. \tag{3.5}$$

With this in mind, the strategy is simple. We need to make sure that the displacement of the optics can satisfy the displacement requirements and residual motion requirements stated in Sec. 2.1 and Sec. 2.2. So, for each stage and

each DoF, we estimate individual $|P_{X_i \to X_{\rm TM}}|\,\hat{X}_i(f)$ and make sure that they satisfy the requirements instead. To estimate this, let's refer back to Fig. 4. The displacement X in the closed-loop configuration reads

$$X = \frac{1}{1+KP}\,D - \frac{KP}{1+KP}\,N\,, \tag{3.6}$$

and the amplitude spectral density reads

$$\hat{X}(f) = \left[\left|\frac{1}{1+KP}\right|^2 \hat{D}(f)^2 + \left|\frac{KP}{1+KP}\right|^2 \hat{N}(f)^2\right]^{\frac{1}{2}}\,. \tag{3.7}$$

If we know the the amplitude spectral densities $\hat{D}(f)$ and $\hat{N}(f)$, and the plant P, we can estimate the amplitude spectral density $\hat{X}(f)$ when designing the controller K. If we know $\hat{X}(f)$ and $P_{X\to X_{\rm TM}}$, then we can estimate $|P_{X\to X_{\rm TM}}|\,\hat{X}(f)$, i.e. the displacement level of the optics due to displacement X. Then, we can tune K such that $|P_{X\to X_{\rm TM}}|\,\hat{X}(f)$ satisfies the requirements. This is how we can design the controllers such such that the requirements are met.

But realistically, we cannot design the controllers in arbitrary order. This strategy requires the a top-down hierarchical control tuning scheme, i.e. tuning the controllers stage-by-stage from the stage closest to the ground. This is because the amplitude spectral density of the disturbance $\hat{D}(f)$ is not easy to estimate. In fact, the disturbance at a lower stage depends on the displacement level at a higher stage, which depends on the controller of that higher stage. This is demonstrated in Fig. 6. The only disturbance we can estimate is the disturbance at the highest stage D_1, i.e. the preisolator (IP), whose disturbance is closely related to the seismic noise X_g.

Now, let's re-elaborate the strategy using Fig. 6, which shows a general hierarchical control architecture of KAGRA's suspensions. The strategy goes as follows. We specify a conservative seismic noise lever for X_g, say the seismic noise level at the 90$^{\rm th}$ percentile in winter season as shown in Fig. 2, so we know $\hat{X}_g(f)$. The ASD of disturbance is given by

$$\hat{D}_1(f) = \left|P_{X_g\to X_1}\right|\hat{X}_g(f)\,, \tag{3.8}$$

and $P_{X_g\to X_1}$ can be measured or estimated directly by P_1, but with a unity gain at DC. Then, we can estimate $\hat{N}_1$, the sensing noise at the preisolator stage. After that, we measure P_1, the actuation transfer function of the preisolator stage. From here, we have all the ingredients to estimate the displacement level $\hat{X}_1$ using Eqn. (3.7), when designing the controller K_1. Then, we can also measure the transfer function $P_{X_1\to X_{\rm TM}}$[6]. With this, we can calculate the optics' displacement due to X_1 by

$$\hat{X}_{\rm TM}(f) \approx |P_{X_1\to X_{\rm TM}}|\,\hat{X}_1(f)\,. \tag{3.9}$$

We then tune K_1 such that Eqn. (3.9) satisfies requirements in Sec. 2.1 and Sec. 2.2. After tuning, K_1 is fixed and so is X_1. Hence, we can estimate the disturbance on the next stage via

$$\hat{D}_2(f) = |P_{X_1\to X_2}|\,\hat{X}_1(f)\,. \tag{3.10}$$

Alternatively, we can simply estimate D_2 by measuring the open-loop spectrum of X_2 when the control of X_1 is engaged. But there're two assumptions here, 1) $D_2 \gg N_2$, so the readout is mostly D_2 and that 2) D_1 is suppressed, so X_1 doesn't change a lot when X_g changes. Then, we repeat the process for stage 2, stage 3 after that, and so on until all control loops are closed.

To verify the control performance, we need to actually measure $\hat{X}_{\rm TM}(f)$ and verify that it satisfies the requirements in Sec. 2.1 and Sec. 2.2. However, with all controls engaged, measuring the optics' displacement using the local sensors could yield a wrong result. The optics' readout $X_{\rm TM,readout}$, i.e. the bottom-left path in Fig. 6, can be written as

$$X_{\rm TM,readout} = \frac{1}{1+K_{\rm TM}P_{\rm TM}}\,D_{\rm TM}\,, \tag{3.11}$$

which can be theoretically zero when $K_{\rm TM}$ is high. From Eqn. (3.7), we know that this is not true. And in fact, the lower bound of the displacement is

$$\underline{\hat{X}}_{\rm TM}(f) \approx \min\left(\hat{D}_{\rm TM}(f),\,\hat{N}_{\rm TM}(f)\right)\,. \tag{3.12}$$

[6] It's another story on how to measure this.

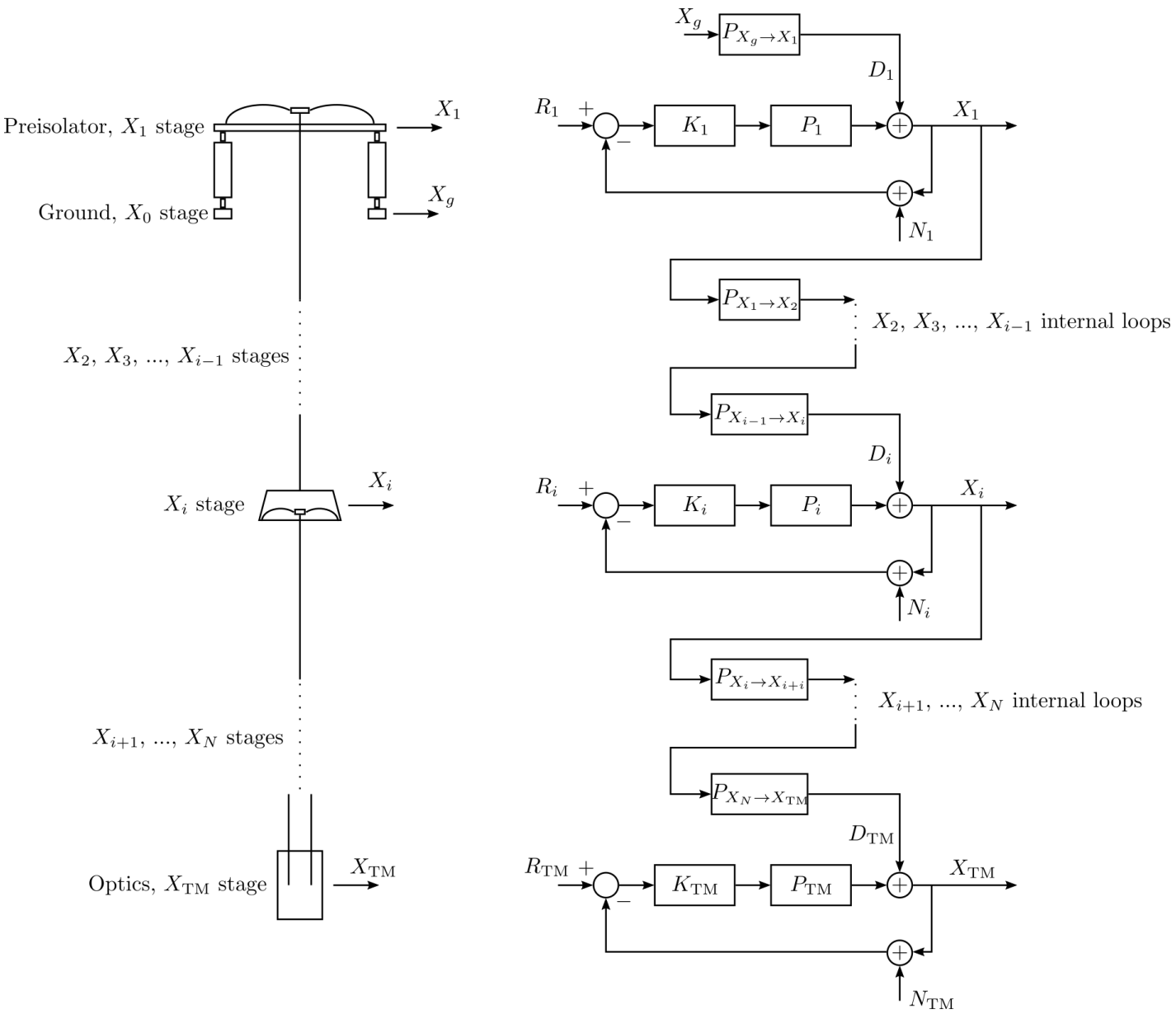

Figure 6: Control block diagram of a general multiple stage suspension with N number of stages. The disturbance of a lower stage is connected to the displacement at a higher stage. Here we use numbered subscripts to denote stages further from the ground for simplicity. In reality, the stages are named something like F1, F2, BF, IM, etc... The optics' displacement X_{TM} is an example.

This is a well known measurement error when using "in-loop" sensors for measurement. To measure the optics displacement properly, we must use sensors that is not part of the control loops, i.e. using an "out-of-loop" sensor. The candidate for this is a second optical lever, which doesn't exist, or the interferometer, which also doesn't exist at the moment. Therefore, we must disengage the controls at the optics stage before we can measure the residual motion of the optics. We assume that the displacement level of the optics will only be lower when the optics controls are engaged. Then, if the measurement above satisfies the residual motion requirements, then having the optics controls engaged will guarantee that the requirements are met as well.

4 Suspension Commissioning Baseline Methods

In this section, we will review some methods that can be used to tackle tasks as listed in Sec. 3.1. This section will include "baseline" methods, which are some techniques that are considered to be the standard, or the fallback, methods that have been implemented previously or are simple enough that they must work. They are sufficiently decent methods that should theoretically get the suspensions to satisfies the requirements. These are also methods that has been used previously by various experts at KAGRA, so these shouldn't new to many of us. However, these methods can be suboptimal. For advanced methods, please refer to the next section, Sec. 5.

4.1 Sensor noise measurement and estimation

In this section we will describe the measurement and estimation of the intrinsic noise of displacement sensors and inertial sensors, and how to model them.

4.1.1 Displacement sensors

Displacement sensors are usually relative displacement sensors that measure the differential displacement between the mounting point and an object. Example sensors are Linear Variable Differential Transformer (LVDT) [21], optical sensors and electromagnetic actuator (OSEM) [22, 23], photo-reflective displacement sensors/photo sensors (PS) [24], and optical levers (OpLev) [20, 25, 26].

In KAGRA's suspension, displacements sensors are non-contact sensors, i.e. the sensor doesn't affect the motion the sensing object, so the suspensions can swing freely to achieve passive seismic isolation. However, this would mean that the sensing readout Y contains the motion of the suspension, if the sensors are already installed, i.e.

$$Y = X + N\,, \tag{4.1}$$

where X is the displacement readout and N is the sensing noise. Therefore, the only way to measure the sensor noise of the displacement sensors is to fix the suspension using the security structure, such that $X = 0$.

Alternatively, we can measure the sensor noise before installation and apply proper calibration factors afterwards to convert the measurement units to displacement. However, there might be error using this method, as there's no guarantee that the sensor noise retains the same before and after installation (or outside/inside vacuum/chamber).

Now, the aforementioned two methods can only be done before/during the installation stage. If sensors are installed and in operation, there's no way to use these methods to measure the sensor noise of the displacement sensors[7]. But, approximation can still be done if have a model of the sensor noise. This will be discuss in Sec. 4.2.1.

4.1.2 Inertial sensors

In this section, we will describe techniques using correlation methods to measure sensor noise of inertial sensors [27, 28]. Please also refer to appendix A for definitions and properties related to spectral densities.

Unlike displacement sensors, inertial sensors are self-contained sensors that are mounted to an object, whose motion are to be measured. Some example sensors are geophone [1], accelerometers [29], and seismometers [30]. Calibrated inertial sensors output the velocity or the acceleration of the object which is relative to the object's inertial frame. Because of this, sensor noise of inertial sensors cannot be measured by measuring the readouts when fixing suspension, as the readouts become the motion of the ground (effectively making the inertial sensor a seismometer.). In any case, The readout of a single inertial sensor will be no different from that of Eqn. (4.1). Therefore, we cannot use single readout to estimate the sensor noise.

[7] Can we use a three-channel correlation method?

Instead, we rely on using multiple sensors and use correlation methods. Now, let's assume that we have multiple sensors that reads Y_i, where $i = 1, 2, \ldots, K$, and K is the total number of sensors. If we place the sensors at the same location, they measure the same signal X. Then, the sensor readouts becomes

$$Y_i = XH_i + N_i \,, \tag{4.2}$$

where H_i is the transfer function from the signal to the sensing readout and N_i is the sensor noise of the i^{th} sensor.

Two-channel method If we have sensors that have the same response and same power spectral density, then we can determine the sensor noises using only two sensors [27]. Let's say the sensors are well calibrated such that $H_i = 1$, and assuming that the signal X_i is uncorrelated with the noise N_i, then the power spectral density of the sensor readouts is simply

$$P_{y_i y_i}(f) = P_{xx}(f) + P_{n_i n_i}(f) \,, \tag{4.3}$$

and the cross power spectral density (CPSD) between Y_i and Y_j is

$$\begin{aligned} P_{y_i y_j}(f) &= P_{xx}(f) + P_{x n_i}(f) + P_{n_i x}(f) + P_{n_i n_j}(f) \\ &= P_{xx}(f) \,, \end{aligned} \tag{4.4}$$

where we assume X, N_i, and N_j are uncorrelated for $i \neq j$ so their CPSDs are identically zero. Following that, the coherence between sensor readout Y_i and Y_j is

$$\begin{aligned} C_{y_i y_j}(f) &= \frac{\left|P_{y_i y_j(f)}\right|^2}{P_{y_i y_i}(f) P_{y_j y_j}(f)} \\ &= \frac{P_{xx}(f)^2}{\left[P_{xx}(f) + P_{n_i n_i}(f)\right]\left[P_{xx}(f) + P_{n_j n_j}(f)\right]} \,. \end{aligned} \tag{4.5}$$

Here, if the power spectral densities of N_i and N_j is the same, i.e. $P_{n_i n_i}(f) = P_{n_j n_j}(f)$, Eqn. (4.5) further simplifies to

$$C_{y_i y_j}(f) = \frac{P_{xx}(f)^2}{\left[P_{xx}(f) + P_{n_i n_i}(f)\right]^2} \,. \tag{4.6}$$

Substituting $P_{y_i y_i}(f) = P_{xx}(f) + P_{n_i n_i}(f)$ and rearranging, we get

$$\begin{aligned} C_{y_i y_j}(f)^{\frac{1}{2}} &= \frac{P_{xx}(f)}{P_{y_i y_i}(f)} \\ P_{y_i y_i}(f) C_{y_i y_j}(f)^{\frac{1}{2}} &= P_{xx}(f) \\ P_{y_i y_i}(f)\left[1 - C_{y_i y_j}(f)^{\frac{1}{2}}\right] &= P_{y_i y_i}(f) - P_{xx}(f) \,. \end{aligned} \tag{4.7}$$

Substituting $P_{n_i n_i}(f) = P_{y_i y_i}(f) - P_{xx}(f)$, finally we obtain

$$\boxed{P_{n_i n_i}(f) = P_{y_i y_i}(f)\left[1 - C_{y_i y_j}(f)^{\frac{1}{2}}\right]} \,. \tag{4.8}$$

And again, note that Eqn. (4.8) only works if the two sensors are identical, i.e. having the same noise spectral density and are inter-calibrated such that the read a same coherent signal.

Three-channel method Now, if we don't have identical sensors, which is generally true, then we have to rely on a three-channel method that uses 3 sensors [28]. The advantage of this method is that we can estimate the sensor noise of each individual sensor, even if they have completely different calibration, dynamics, and noise spectrum. Recall the sensor readout Eqn. (4.2), the cross power spectral density between the i^{th} and j^{th} sensors is

$$P_{y_i y_j}(f) = P_{xx}(f) H_i H_j^* \,, \tag{4.9}$$

where H_j^* denotes the complex conjugate of the transfer function H_j and $i \neq j$. Again, we have assumed that the coherent signal X, the noises N_i and N_j are uncorrelated such that their CPSDs are zero. If we have three sensors

then we can have two cross power spectral density, $P_{y_i y_k}(f)$ and $P_{y_j y_k}(f)$, and $i, j, k = 1, 2, 3$ and $i \neq j \neq k$. Then, taking the ratio gives

$$\frac{P_{y_i y_k}(f)}{P_{y_j y_k}(f)} = \frac{H_i}{H_j} \,. \tag{4.10}$$

The ratio between the PSD $P_{y_i y_i}(f)$ and CPSD $P_{y_j y_i}$ reads

$$\begin{aligned} \frac{P_{y_i y_i}(f)}{P_{y_j y_i}(f)} &= \frac{P_{xx}(f) H_i H_i^* \; + P_{n_i n_i}(f)}{P_{xx}(f) H_j H_i^*} \\ &= \frac{H_i}{H_j} + \frac{P_{n_i n_i}(f)}{P_{y_j y_i}(f)} \,. \end{aligned} \tag{4.11}$$

Now, substituting Eqn. (4.10) into Eqn. (4.11) and rearranging gives

$$\boxed{P_{n_i n_i}(f) = P_{y_i y_i}(f) - \frac{P_{y_i y_k}(f)}{P_{y_j y_k}(f)} P_{y_j y_i}(f)} \,, \tag{4.12}$$

which expresses the PSD of the noise as the PSDs and the CPSDs of the measurements.

Warning. The following paragraph is not from [28], but is seemingly important.

Modified three-channel method Let's inspect Eqn. (4.12). Eqn. (4.12) is an equation as written in [28] but is seemingly not directly implementable. The second term in Eqn. (4.12) are products of cross power spectral densities. If we look at Eqn. (4.10), the CPSDs are expressed in products of transfer functions, are complex-valued series. These features should note exist in a power spectral density as it's expected to be a real-valued frequency series. To fix this problem, we propose to modify Eqn. 4.12 by taking the absolute value of the second term, i.e.,

$$\begin{aligned} P_{n_i n_i}(f) &\approx P_{y_i y_i}(f) - \left| \frac{P_{y_i y_k}(f)}{P_{y_j y_k}(f)} P_{y_j y_i}(f) \right| \\ &= P_{y_i y_i}(f) - \frac{|P_{y_i y_k}(f)|}{|P_{y_j y_k}(f)|} \left| P_{y_j y_i}(f) \right| \,. \end{aligned} \tag{4.13}$$

Recall that the definition of coherence function, first line of Eqn. (4.5), we can write express the absolute value of a cross power spectral density as

$$\left| P_{y_i y_j}(f) \right| = C_{y_i y_j}(f)^{\frac{1}{2}} P_{y_i y_i}(f) P_{y_j y_j}(f) \tag{4.14}$$

where $C_{y_i y_j}(f)$ is the coherence function between readouts Y_i and Y_j. Substituting Eqn. (4.14) into Eqn. (4.13) gives

$$\boxed{P_{n_i n_i}(f) \approx P_{y_i y_i}(f) \left[1 - \left(\frac{C_{y_i y_k}(f)}{C_{y_j y_k}(f)} C_{y_j y_i}(f) \right)^{\frac{1}{2}} \right]} \,, \tag{4.15}$$

which is an expression very similar to Eqn. 4.8 and has convenient properties, i.e. coherence functions are symmetric $C_{ij}(f) = C_{ji}(f)$.

4.1.3 Examples

We will demonstrate the use of these correlation methods in the form of a Jupyter notebook. The notebook is available in the Kontrol library [31], where Eqn. (4.8), (4.13), and (4.15) are coded. We will copy important results from the notebook.

Fig. 7 shows a comparison between the two correlation methods from the notebook. Sensor 1 and 2 has the same dynamics and noise PSDs, while sensor 3 has a completely different dynamics and noise PSD from the other two. We used two-channel method to predict sensor noise 1 and 2, and used three-channel method for all of them. We also used the two-channel method to predict sensor noise 3 using sensor 1 as a correlated sensor. As shown in the figure, the predicted PSDs (shown in green dashed and dotted black lines) are very close to the actual sensor noises

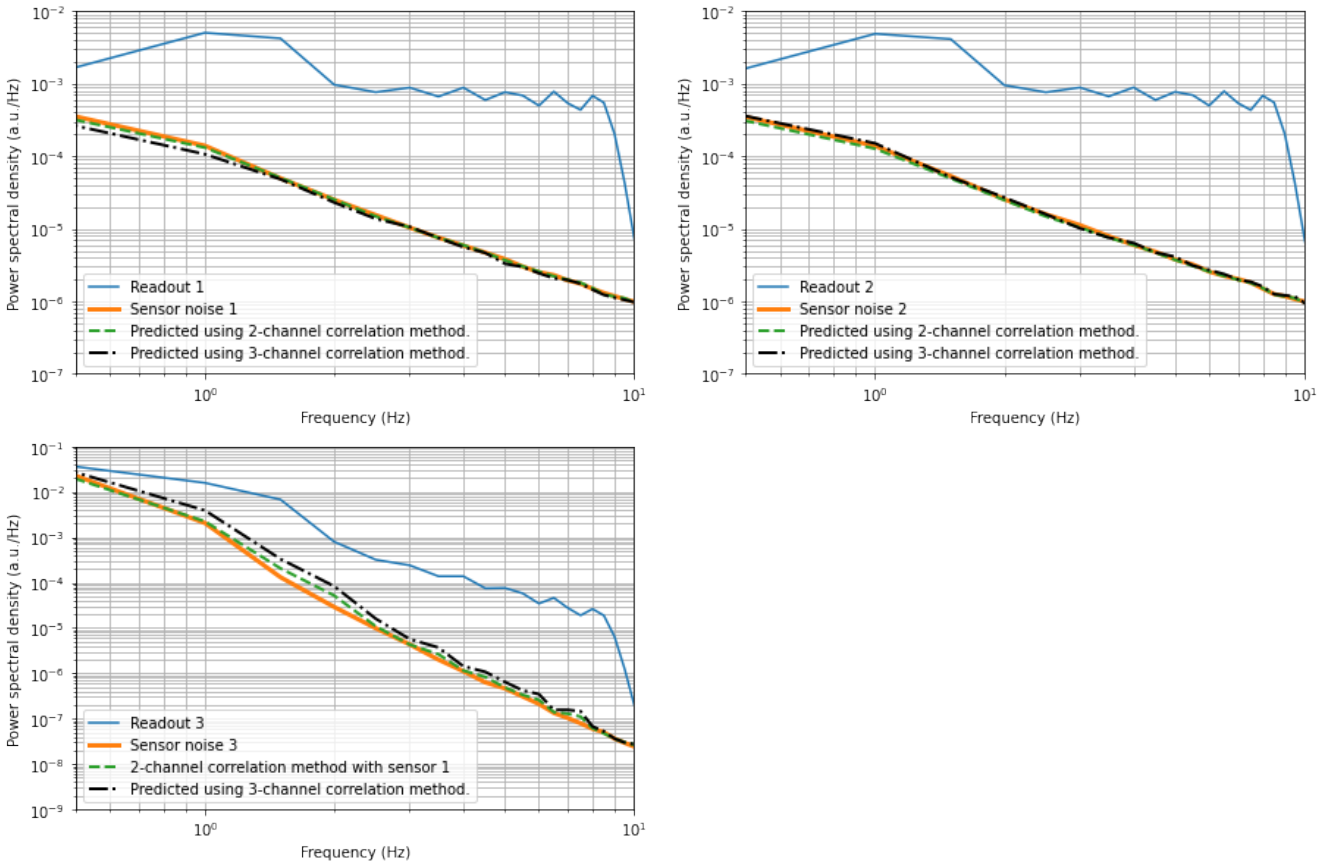

Figure 7: A comparison between the two-channel method and three-channel method.

(shown in orange). Here, we emphasize that using the two-channel method to predict self noise of sensor 3 is not a valid attempt as the other sensors don't have the same sensor dynamics and noise spectral density. Hence, the green dashed line on the bottom left in Fig. 7 might be a fluke. We used the same method to predict sensor noise 1 using sensor 3 as a correlated sensor and the prediction was clearly off (See the notebook for a figure).

4.2 Frequency series fitting using mathematical optimization

In this section, we will be discussing how to model frequency series using optimization methods. In particular, we will discuss the fitting of noise spectum measurments in Sec. 4.2.1, as a follow up of Sec. 4.1. And, the fitting of transfer function measurements is discussed in Sec. 4.2.2. Some general tips on fitting frequency series using optimization is given in Sec. 4.2.3. In Sec. 4.2.4, we will exemplify the methods discussed. We will be using mathematical optimization notations as given in appendix B. A good `SciPy` reference on mathematical optimization can be found in [32]. In this section, we will refer the word "optimization" to "mathematical optimization", which is to minimize an objective function/cost function.

4.2.1 Noise spectrum modeling

In this section, we will discuss how to obtain a noise model from some sensor readout. This can be useful if we need to have a analytic model or if we want to estimate the sensor noise from a signal sensor, whose readout is not purely noise-dominated.

If conditions in the Sec. 4.1.1 and Sec. 4.1.2, such as fixing the suspensions or measuring the readout before installation, are not available, we cannot use the aforementioned noise measurement methods. But, we can still

estimate the sensor noise via a curve fitting procedure, if we have a noise model and if the signal sensor readout is not dominated at all frequencies, i.e. signal only dominates spectral partially. An example of this situation would be a sensor that measures a damped sinusoidal, i.e.

$$y(t) = x(t) + n(t) = C\Re\left(e^{\sigma+i\omega_n t}\right) + n(t)\,, \tag{4.16}$$

where $x(t) = C\Re\left(e^{\sigma+i\omega_n t}\right)$ is the signal, C is a real number, σ is a negative real number, ω_n is the oscillation frequency, and $n(t)$ is the sensor noise. In this case, the frequency spectrum of signal $x(t)$ is localized around $f = 2\pi\omega_n$, while the frequency content of the sensor noise $n(t)$ spreads all frequencies, which is very typical for motion sensors in a suspension.

Say, we have measurements

$$\hat{Y}(f) = \left[\hat{X}(f)^2 + \hat{N}(f)^2\right]^{\frac{1}{2}}\,, \tag{4.17}$$

where $\hat{Y}(f)$ is the ASD of the sensor readout $y(t)$, $\hat{X}(f)$ is the ASD of some signal $x(t)$, and $\hat{N}(f)$ is the ASD of the sensor noise $n(t)$ that we want to measure/model. If we have a noise amplitude spectral model $\hat{N}_{\text{model}}(f;\theta_{\hat{N}_{\text{model}}})$, where $\theta_{\hat{N}_{\text{model}}}$ is a list of parameters that defines the noise $\hat{N}_{\text{model}}$, then we can do a curve fit to obtain the noise model. A example choice of the noise model would be a quadrature sum

$$\hat{N}_{\text{model}}(f;\theta_{\hat{N}_{\text{model}}}) = \left[\left(\frac{N_a}{f^a}\right)^2 + \left(\frac{N_b}{f^b}\right)^2\right]^{\frac{1}{2}}\,, \tag{4.18}$$

where $\theta_{\hat{N}_{\text{model}}} = \{N_a, N_b, a, b\}$ are the model parameters.

The fit can be as simple as a least squares fit, i.e. by minimizing the sum of error squares cost function

$$J_{\text{lsq}}(\theta; A(m), B(m,\theta)) = \sum_m^M \left[A(m) - B(m,\theta)\right]^2 w(m)^2\,, \tag{4.19}$$

where θ is some model parameters of an arbitrary analytical function $B(m,\theta)$, $A(m)$ is the measurement data, m is a common parametrization of the series A and B, and $w(m)$ is a user-designed weighting of each data point (weighting function). Typical choices for m could be time t, frequency f, or simply data index. Minimization of Eqn. (4.19) will give you the optimal parameters θ^* such that the best fit of data A is the model $B(\theta^*)$.

We can use the least squares fit directly but this is not the best for fitting measurement data that is typically viewed in log or log-log plots, where the value of data points varies drastically in orders of magnitude. Instead, we use $J_{\text{lsq}}(\theta; \log A(m), \log B(m,\theta))$.

As for the weighting function $w(f)$ (now as a function of frequency), an easy choice for our purpose would be

$$w(f) = \begin{cases} 0 & \text{if } f_{\text{lower}} < f < f_{\text{upper}}\,, \\ 1 & \text{otherwise}\,, \end{cases} \tag{4.20}$$

where $(f_{\text{lower}}, f_{\text{upper}})$ is a frequency bound that encloses frequency regions that has high SNR, indicated by the peak feature. The choices of the design parameters f_{lower} and f_{upper} can't be easily told here as it requires users to look at the visualized data and make educated guesses. Instead, we will demonstrate the choice of this bound in an example shown in Sec. 4.2.4.

To sum up, we have sensor data $\hat{Y}(f)$, which is an ASD (works for PSD as well.) as shown in Eqn. (4.17), and we would like to model the sensor noise $\hat{N}(f)$ with a model $\hat{N}(f;\theta_{\hat{N}_{\text{model}}})$, where $\theta_{\hat{N}_{\text{model}}}$ is a list of parameters of the sensor noise model. We do so by minimizing the cost function

$$\boxed{J_{\text{noise}}\left(\theta_{\hat{N}_{\text{model}}}\right) = \sum_m^M \left[\log\hat{Y}(f_m) - \log\hat{N}_{\text{model}}(f_m, \theta_{\hat{N}_{\text{model}}})\right]^2 w(f_m)^2}\,, \tag{4.21}$$

where the weighting function $w(f)$ is given in Eqn. (4.20) and f_m are the frequencies where the measured PSD and the model are evaluated. The reason for using the log error $\log\hat{Y}(f) - \log\hat{N}_{\text{model}}(f;\theta_{\hat{N}_{\text{model}}})$ instead of the typical

error is given in Sec. 4.2.3, and will be obvious in the example section 4.2.4. Minimization gives best-fit parameters

$$\theta^*_{\hat{N}_{\text{model}}} = \arg\min_{\theta_{\hat{N}_{\text{model}}}} \hat{J}_{\text{noise}}\left(\theta_{\hat{N}_{\text{model}}}\right), \tag{4.22}$$

such that the noise model $\hat{N}_{\text{model}}\left(f; \theta^*_{\hat{N}_{\text{model}}}\right)$ becomes a best fit of the noise spectral density $\hat{N}(f)$. The minimization can be done using methods that will be discussed in Sec. 4.2.3.

4.2.2 Transfer function modeling

4.2.3 Some tips on using optimization

Minimization with measurement data can be done iteratively using optimization algorithms like gradient descent [33] (local minimization) or stimulated annealing [34] (global minimization). We will not dive into the topic of mathematical optimization since it's a topic on it's own and is out of the scope of this document. Instead, we will simply be using packages that are readily available. A very useful Python submodule is `scipy.optimize` [35]. It provides functions for both local minimization (`scipy.optimize.minimize()`) and global minimization. It comes with many popular algorithms such as the Nelder-Mead algorithm (`scipy.optimize.minimize(..., method="Nelder-Mead")`) [36], which is known to be good for optimization with noisy experiment data, or differential evolution (`scipy.optimize.differential_evolution()`) [37]. The choice of algorithm is very subtle as different algorithms practically perform equally well for our purpose. If you really need to know which one to use, try consulting [32]. However, we do distinguish the choice between a local minimization algorithm and a global one. The former can only be used when with guessed initial parameters, while the latter one can only be used when the boundaries of the parameters is known. Typically, we[8] prefer global optimization approaches because typically we don't know the parameters. While the optimization result can depend on the initial guesses, having a poor guess would yield suboptimal results.

An important note on numerical optimization is parameter scaling [38]. We won't go into details. The idea is to scale the parameters in such that they are all in the same scale, hence the cost function is equally sensitive to all parameters. For some parameters $\theta = \{\theta_1, \theta_2\}$ that has a large dynamic range, it might be helpful if we optimize $\tilde{\theta} = \{\log\theta_1, \log\theta_2\}$ instead. An example where this kind of rescaling is necessary is transfer function fitting with polynomial models, which have coefficients the vary in orders of magnitude. Another example would be N_a and N_b from Eqn. (4.18), which are noise levels that can also vary drastically.

Another useful pre-processing procedure would be to resample the data using an log-spaced frequency axis, instead of using the linearly spaced one as obtained during Fourier transform. With linear-space frequency points, the majority of the data clusters at higher frequencies as viewed from a log-frequency axis, which create bias towards high frequency data. As a result, the best-fit model might fit measurement data at higher frequencies better than that at lower frequencies. A simple resample of the data will redistribute the data uniformly across the log-frequency axis.

4.2.4 Examples

Noise spectrum modeling Here, we will demonstrate how to use optimization to minimize cost functions (4.19) and (4.21) to model a sensor noise ASD $\hat{N}(f)$. The notebook is available here. We will assume a sensor readout $y(t)$ that follows Eqn. (4.16), with a noise $n(t)$ that follows Eqn. (4.18). We will add measurement noise uniformly drawn from $[-3, 3]$ dB to the sensor readout $y(t)$.

Fig. 8 shows the time series of the signal $x(t)$ and the frequency series $\hat{Y}(f)$, $\hat{N}(f)$, $\hat{X}(f)$, and the weighting function $w(f)$. Here, the signal is

$$x(t) = A\Re\left(e^{\sigma + i\omega_n t}\right), \tag{4.23}$$

[8] I mean I (Terrence)

and we set $A = 1$, $\sigma = -0.001$, $\omega_n = 2\pi$. The noise spectral density is

$$\hat{N}(f) = \left[\left(\frac{N_{3.5}}{f^{3.5}} \right)^2 + \left(\frac{N_1}{f^1} \right)^2 \right]^{\frac{1}{2}} , \tag{4.24}$$

where $N_{3.5} = \left(5 \times 10^{-7}\right)^{\frac{1}{2}}$ and $N_1 = \left(1 \times 10^{-7}\right)^{\frac{1}{2}}$. These frequency dependencies (-3.5 and -1) are chosen to mimic the behavior of the geophone noise. This kind of sensor noise profiles has large amplitude variations at different frequencies so it can be used as a benchmark problem for noise modeling. So, the ASD of the sensor readout, in dB is

$$20 \log \hat{Y}(f) = 20 \log \left[\hat{X}(f)^2 + \hat{N}(f)^2 \right]^{\frac{1}{2}} + \mathcal{U}(-3, 3) , \tag{4.25}$$

where $\mathcal{U}(a, b)$ is a random series drawn from a uniform distribution between a and b.

In reality, the only information we have is the sensor readout $\hat{Y}(f)$, as given by the green curve shown in Fig. 8. From the plot, we see a peek feature around $0.7 - 2\,\text{Hz}$, which indicates the presence of the signal $x(t)$. This is something that we do not wish to model. Hence, we set the weighting function

$$w(f) = \begin{cases} 0 & \text{if } 0.7 < f < 2\,\text{Hz} , \\ 1 & \text{otherwise} , \end{cases} \tag{4.26}$$

so it filters out the part of the readout that we don't wish to model.

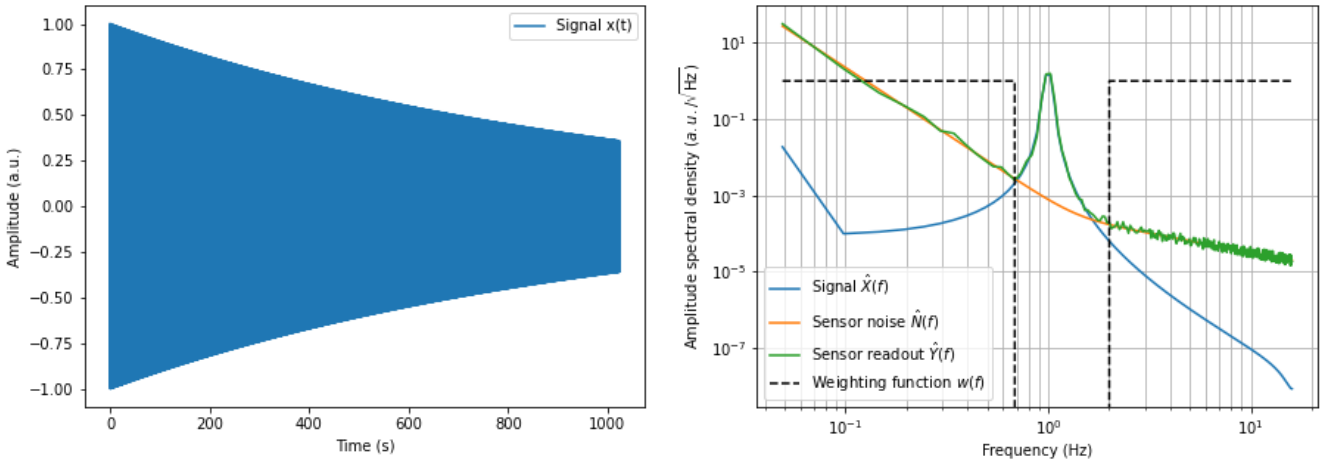

Figure 8: (Left) Time series of the signal $x(t)$. (Right) Amplitude spectral densities of the readout $\hat{Y}(f)$, sensor noise $\hat{N}(f)$, signal $\hat{X}(f)$, and the designed weighting function $w(f)$.

As for the model, we model (4.18). We fit the measurement with 4 different configurations. For the first two, we use least squares fit, which uses Eqn. (4.19), with model parameters $\theta_{\hat{N}_{\text{model}}} = \{N_a, N_b, a, b\}$, and "log parameters" $\tilde{\theta}_{\hat{N}_{\text{model}}} = \{\log N_a, \log N_b, a, b\}$, respectively. For the other two, we use least squares fit with log error, which uses Eqn. (4.21), again with the two different sets of model parameters. The motivation for using log parameters is discussed in Sec. 4.2.3.

Fig. 9 shows the optimization results with the four configurations. As can be seen on the left plot, least squares fit with log error yields noise models that fit the actual noise well across all frequencies. But, the traditional least square fit fails at fitting the data at higher frequencies, where the amplitudes of the noise is orders of magnitude lower than that at lower frequencies. Right plot in Fig. 9 shows the residuals, defined by the square error between the actual noise and the fit, of these four configuration. As shown in the plot, modeling the noise with least square fit with log error and log parameters yields the best result in default settings. This shows the importance in correct scaling of the model parameters.

For all these results, we used `scipy.optimize.minimize()`, with default settings, to minimize the cost functions, and we start from an initial guess of $\{0, 0, 0, 0\}$ in all cases. We emphasize that all these four configuration may

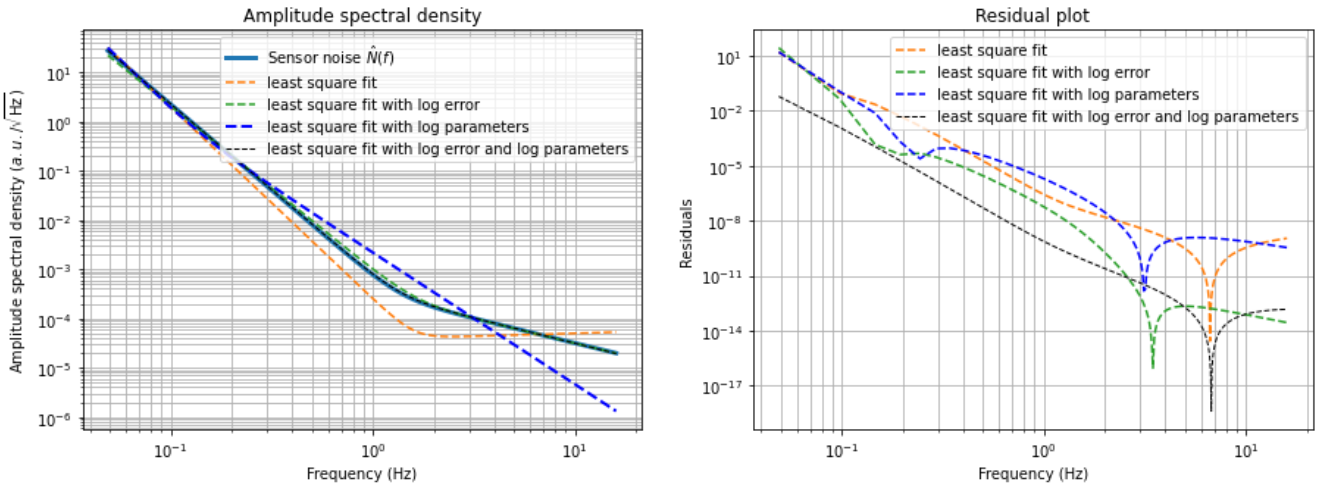

Figure 9: (Left) Fitting results. (Right) Residuals, i.e. error squared.

perform equally well if we further tweak the optimization parameters in `scipy.optimize.minimize()`, or using different optimization algorithms. There are many optimization tweaks that can improve the fitting performance, but we won't discuss here. The results shown in Fig. 9 only serves as a demonstration of how a careful design of the model and the cost functions can yield better results under the same settings.

Transfer function modeling Pending.

4.3 Control matrices

In this section, we will discuss how control matrices (sensor matrices and actuation matrices) can be derived, and how to fine tune them (diagonalization).

As is discussed in Sec. 3.2, there are many sensors and actuators in a single stage of suspension. And in fact, sensors and actuators mostly come in pairs for KAGRA suspensions. For instance, at the preisolator stage, there are 3 linear variable differential transformers (LVDTs) (not counting inertial sensors), and 3 coil-magnetic actuators placed at close proximity to the LVDTs. Another example would be the intermediate-mass stage, which has 6 optical sensors and electromagnetic actuator (OSEMs) or photo-reflective displacement sensors (PS). One exception would be the optics stage which has four coil-magnetic actuators, while only have an optical lever that can measure displacements from 3 degrees of freedom (DoFs). In principle, the number of independent sensors, or I should say, readouts, decides the number of degrees of freedom that we can measure. The minimum number of sensors that is needed is at least the number of degrees of freedom that we want to measure. And, so this is equivalent to saying that, the maximum number of degrees of freedom that we can measure, is the maximum number of sensors. The same goes to actuators.

Having sufficient number of sensors and actuators at a stage gives us the possibility to sense or actuate the motion at different DoFs of a stage. But, the sensors and actuators may not be aligned directly to the degrees of freedom that we would like to measure. For example, the LVDTs at the preisolator stage are located at the outer edge of the preisolator table, which is circular in shape so three LVDTs are sensing the tangential displacement of the preisolator table at three different angles (see [15] for a simple figure). Therefore, the readouts must be transformed from the sensor basis to the basis that we would like to perform control. The same goes to actuation. This is done via control matrices.

The desired basis is typically Cartesian (longitudinal, transverse, and vertical) with Euler angles (roll, pitch, yaw). The Cartesian basis is defined locally for each suspension, where longitudinal direction is typically the normal of the high-reflectivity (HR) side of the optics, vertical is up, and transverse is the direction such that the cross product of the longitudinal direction and transverse direction is vertical. As for the angles, roll, pitch, and yaw are defined as the angular displacements along the longitudinal, transverse, and vertical axis, respectively. These DoFs can be

locally defined for each stages as well. For example, we would say IP yaw, IM yaw, and TM yaw, which are yaw at the preisolator, intermediate mass, and optics, respectively. But, in this section we will focus on stage-wise control matrices, i.e. not mixing sensor readouts from different stages.

In a nutshell, here is how control matrices are derived for each stage. We express each sensor readout $y_i(t)$ as a superposition of the stage's displacements $x_j(t)$,

$$y_i(t) = \sum_j^m C_{ij} x_j(t)\,, \tag{4.27}$$

where $i = 1, 2, 3, \ldots, n$ where n is the number of sensors, and $j = 1, 2, 3, \ldots, m$ where m is the number of degrees of freedom, C_{ij} is the matrix elements of a matrix $\mathbf{C}$ and $\mathbf{C} \in \mathbb{R}^{n\times m}$. C_{ij} is the coupling coefficient of displacements $x_j(t)$ to sensor $y_i(t)$ Then, the sensing matrix, defined by a conversion that maps the sensor readouts to the stage displacements, is simply

$$\mathbf{C}_{\text{sensing}} = \mathbf{C}^\dagger\,, \tag{4.28}$$

where $\mathbf{C}_{\text{sensing}} \in \mathbb{R}^{m\times n}$, such that $\mathbf{x}(t) = \mathbf{C}_{\text{sensing}}\mathbf{y}(t)$.

The same goes to actuation. Suppose we have actuation signals $u_i(t)$ injected to the i^{th} actuator. Since the actuators are not purely acting on a single DoF, the actuation signal reads (at DC only, at higher frequencies there's phase shift. To be discussed in Sec. 4.3.3),

$$u_i(t) = \sum_j^m D_{ij} x_j(t)\,, \tag{4.29}$$

where $i = 1, 2, 3, \ldots, n$ where n is the number of actuators, $j = 1, 2, 3, \ldots, m$ where m is the number of DoFs, and D_{ij} is the matrix elements of a matrix $\mathbf{D}$ and $\mathbf{D} \in \mathbb{R}^{n\times m}$. Here, different from C_{ij}, the scale of D_{ij} doesn't not matter, so long as the row vectors D_{ij} has a direction parallel to the actuation. Again, the actuation matrix, defined by the matrix mapping the actuation to the stage displacements, is simply

$$\mathbf{D}_{\text{actuation}} = \mathbf{D}^\dagger\,, \tag{4.30}$$

and $\mathbf{D}_{\text{actuation}} \in \mathbb{R}^{m\times n}$, such that $\mathbf{x}(t) = \mathbf{D}_{\text{actuation}}\mathbf{u}(t)$.

Now, initial control matrices can be determined from the location and orientation, i.e. from geometry, of the sensors and actuators with respect to the stage. [19] and [18] gives detailed calculation of the actuation and sensing matrices for the SR suspensions and BS suspensions for all stages, except sensing for the optics stage. The sensing of the optics stage utilizes an optical lever setup and the derivation is more involved. The derivation of the initial sensing matrix of optical levers is given in [20]. Details of these initial control matrices will be be discussed here.

Having the initial matrices may not be sufficient as residual cross-couplings may still remain. In Sec. 4.3.1, we will discuss how to modify the sensing matrices so each sensor readout measures one pure DoF. In Sec. 4.3.2, we will discuss how to modify the actuation matrices so each actuation signal purely moves the stage in one direction (at one paricular frequency only). In Sec. 4.3.3, we will discuss a more general actuation matrix, which is a transfer function matrix, i.e. frequency dependency. These methods are referred to sensor and actuation "diagonalization" in KAGRA.

4.3.1 Sensing matrices

Sensor and actuation diagonalization are two separate procedures but, as we shall we, sensor diagonalization must happen before actuation diagonalization, as it depends on the sensing readouts. When sensor diagonalization is performed, the actuators should be re-diagonalized if it was done with a different sensing matrix. Now, let's say we have raw sensing signals $\mathbf{y}(t) = y_i(t)$, where $i = 1, 2, 3, \ldots, n$, and n is the number of sensors we have. These signals are what the sensors measure. They can be raw voltage, or calibrated in some sort of displacement units. We have a initial sensing matrix $\mathbf{C}_{\text{sensing}}^{\text{initial}}$ that we derived from the geometric location and the orientation of the sensors. Hence, we have some displacement readouts

$$\mathbf{x}_{\text{readout}}(t) = \mathbf{C}_{\text{sensing}}^{\text{initial}}\mathbf{y}(t) = \mathbf{C}_{\text{coupling}}\mathbf{x}(t) \approxeq \mathbf{x}(t)\,, \tag{4.31}$$

where $\mathbf{x}(t)$ is the displacements of the suspensions that we would like to measure and $\mathbf{C}_{\text{coupling}}$ is a coupling matrix, with off-diagonal terms being the cross-coupling coefficients.

If the initial sensing matrix is perfect, then the coupling matrix $\mathbf{C}_{\text{coupling}}$ will be the identity matrix $\mathbf{I}$ such that $\mathbf{x}_{\text{readout}}(t) = \mathbf{x}(t)$. But, in general, this is not the case. If the off-diagonal elements in the coupling matrix are not zero, then the displacement readouts $\mathbf{x}_{\text{readout}}(t)$ are cross-coupled, and the goal of diagonalization is to adjust the sensing matrix so individual readout is only coupling to one degree of freedom. To decouple the readouts, we simply need to modify the sensing matrix such that

$$\boxed{\mathbf{C}_{\text{sensing}} = (\mathbf{C}_{\text{coupling}})^{-1}\, \mathbf{C}_{\text{sensing}}^{\text{initial}}}\,. \tag{4.32}$$

Depending on the suspensions' real-time model architecture, this can be done by multiplying an addition matrix after the initial sensing matrix, or it can be done by simply modifying the existing matrix.

To measure the coupling matrix $\mathbf{C}_{\text{coupling}}$, we have to rely on resonance features of the suspension. Some resonant modes of the suspension only involve a single direction. If these modes have distinguishable frequencies from other modes, then we can estimate the off-diagonal coupling coefficients ratios between the i^{th} readout and j^{th} readout,

$$\boxed{C_{\text{coupling},ij} = \begin{cases} \left|\frac{X_{\text{readout},i}(s)}{X_{\text{readout},j}(s)}\right|_{f=f_j}\,, & \angle\frac{X_{\text{readout},i}(s)}{X_{\text{readout},j}(s)}\Big|_{f=f_j} \approx 0\,, \\ -\left|\frac{X_{\text{readout},i}(s)}{X_{\text{readout},j}(s)}\right|_{f=f_j}\,, & \angle\frac{X_{\text{readout},i}(s)}{X_{\text{readout},j}(s)}\Big|_{f=f_j} \approx \pm\pi\,, \end{cases}} \tag{4.33}$$

where f_j is the resonance frequency of the x_j degree of freedom and $X_{\text{readout},i}(s)$ is the Laplace transform of the i^{th} readout of $\mathbf{x}_{\text{readout}}(t)$. And note here we have explicitly specify the s dependency to distinguish transfer functions and matrices.

Here, a few assumptions are made. First of all, assume that the suspension stages are oscillating to the extent the peaks in the spectrum are high enough for us to measure coupling ratios, or to say that the coupling ratios are small. So, the suspensions must be excite via external actuation, typically using the actuators. There're two ways this can be done, 1) we can use the actuators to give a "kick" to the suspension, i.e. impulse input, so it freely swings in damped oscillation or 2) we can inject white noise actuation to excite the modes. Mathematically, the resulting spectrum should be the same, as an impulse input is effectively white noise in the frequency domain. But, the white noise actuation might be easier to implement, as the damped oscillation excited by the impulse may result in low SNR towards steady-state. Secondly, we assume that the sensors are calibrated such that the diagonal terms of the coupling matrix are ones. Thirdly, we assume that the coupling ratios measured are purely due to linear readout cross-coupling so, if there's cross-coupling, the ratio between the readouts at resonance frequencies must have a 0 relative or $\pm\pi$ phase. If the ratios has a phase that is not close to 0 or $\pm\pi$ but the magnitude of the ratios are not small, this would mean that the results are not conclusive so we should not assign the ratio to the coupling matrix. This could happen due to low SNR, mechanical coupling, or other reasons that are not yet clearly studied. Unless the reason is known, we should not remove these spurious cross-couplings as it would be overcompensating.

4.3.2 Actuation matrices

The idea for actuation diagonalization is similar to that of sensor diagonalization but with some minor differences.

Let's say we have raw actuation signals $\mathbf{u}(t)$ and displacements $\mathbf{x}(t)$. These raw actuation signals are voltage or "counts" applied to the actuators, and the $\mathbf{u}(t)$ frame is in the actuator frame, which is not aligned to the displacements $\mathbf{x}$. We wish to have actuations signals $\mathbf{u}_{\mathbf{x}}(t)$ that are parallel to the displacements $\mathbf{x}(t)$. So $\mathbf{u}_{\mathbf{x}}(t)$ is something like actuation signal in longitudinal, transverse, ..., etc, and this is the variable that the control system manupulates. Whereas, the raw actuation signal $\mathbf{u}(t)$ is like actuation in first actuator, second actuator, ..., etc. The actuation signals need to be converted back to the raw actuation signals via multiplying the actuation signals by the actuation matrix, i.e.

$$\mathbf{u}(t) = \mathbf{D}_{\text{actuation}}\mathbf{u}_{\mathbf{x}}(t)\,, \tag{4.34}$$

where $\mathbf{D}_{\text{actuation}}$ is the actuation matrix.

Now, let's say we have initial actuation matrix $\mathbf{D}_{\text{actuation}} = \mathbf{D}^{\text{initial}}_{\text{actuation}}$. If the actuation is not diagonalized, the displacement due to actuation signals reads

$$\mathbf{x}(t) = \mathbf{D}_{\text{coupling}}\mathbf{u}_{\mathbf{x}}(t)\,, \tag{4.35}$$

where $\mathrm{D}_{\text{coupling}}$ is the coupling matrix due to actuation. The diagonal elements of $\mathbf{D}_{\text{coupling}}$ are the actuation efficiencies, i.e. the transfer function gain from actuation to displacement. Here, it would be tempting to multiply the actuation signal by a matrix $(\mathbf{D}_{\text{coupling}})^{-1}$, such that

$$\mathbf{x}(t) = \mathbf{D}_{\text{coupling}}\left(\mathbf{D}_{\text{coupling}}\right)^{-1}\mathbf{u}_{\mathbf{x}}(t)\,. \tag{4.36}$$

This will diagonalization the actuation, but it is not the ideal approach. This is because this will make the actuation efficiencies from $\mathbf{u}_{\mathbf{x}}(t)$ to $\mathbf{x}(t)$ identically ones. This may cause problems practically as we might want to compare actuation efficiencies or actuation transfer functions from previous measurements. So instead, it follows that if we multiple the actuation signals $\mathbf{u}_{\mathbf{x}}(t)$ by $\left(\mathbf{D}_{\text{coupling}}\right)^{-1}\mathbf{D}_{\text{efficiency}}$ before converting to the raw actuation signals via the initial actuation matrix $\mathbf{D}^{\text{initial}}_{\text{actuation}}$, then the actuation-displacement relation becomes

$$\mathbf{x}(t) = \mathbf{D}_{\text{coupling}}\left[\left(\mathbf{D}_{\text{coupling}}\right)^{-1}\mathbf{D}_{\text{efficiency}}\right]\mathbf{u}_{\mathbf{x}}(t)\,, \tag{4.37}$$

where $\mathbf{D}_{\text{efficiency}}$ is a diagonal matrix with elements being the actuation efficiencies at which the coupling matrix was measured. The actuation efficiency matrix $\mathbf{D}_{\text{efficiency}}$ is a free parameter that we can set. So, for initial setup of the suspensions, this can be any arbitrary desired values. However, for diagonalization, it's better to maintain the static gain of the actuation transfer functions, i.e. maintaining the previous actuation efficiencies. This is equivalent to saying that we should set the actuation efficiency matrix to the diagonal elements of the actuation coupling matrix $\mathbf{D}_{\text{coupling}}$

$$\mathbf{D}_{\text{efficiency}} = \text{diag}\left(\mathbf{D}_{\text{coupling}}\right)\,. \tag{4.38}$$

Now, coming back to the actuation matrix $\mathbf{D}_{\text{actuation}}$. If we must modify the actuation matrix for diagonalization (alternatively we can have matrices in between actuation signals and the initial actuation matrix in the system), then Eqn. (4.37) is equivalent to saying that we should modify the actuation matrix from the initial actuation matrix $\mathbf{D}^{\text{initial}}_{\text{actuation}}$ to

$$\boxed{\mathbf{D}_{\text{actuation}} = \mathbf{D}^{\text{initial}}_{\text{actuation}}\left(\mathbf{D}_{\text{coupling}}\right)^{-1}\mathbf{D}_{\text{efficiency}}}\,. \tag{4.39}$$

To measure the actuation coupling, we need to measure the displacement $\mathbf{x}(t)$ due to actuation $\mathbf{u}_{\mathbf{x}}(t)$. However, to measure $\mathbf{x}(t)$, we can only rely on the sensing readout $\mathbf{x}_{\text{readout}}(t)$ from Sec. 4.3.1. Therefore, it is important to complete sensor diagonalization before actuation diagnalization such that $\mathbf{x}_{\text{readout}}(t) = \mathbf{x}(t)$ Also, note that the level of diagonalization that actuation can achieve is at best the level of diagonalization of the sensors. To measure the actuation coupling matrix, we need to inject actuation signals, typically in the form of a DC offset or a sinusoidal single-frequency line injection. DC offset inject is good because it would mean that there's no relative-phase between actuation and displacement. If we choose to use a line injection, we must choose frequency at which both diagonal transfer functions and cross-transfer functions has 0 or $\pm\pi$ phase. This is tricky because we can't measure the transfer function unless we diagonalize the actuation! Another way to approach this is to choose a frequency f_{inject} such that the measured $\frac{\mathbf{X}(s)}{\mathbf{U}_{\mathbf{x}}(s)}\Big|_{f=f_{\text{inject}}}$ has 0 or $\pm\pi$ phase. So, the actuation coupling matrix can be measured as

$$\boxed{D_{\text{coupling},ij} = \begin{cases} \left|\frac{X_i(s)}{U_{\mathbf{x},j}(s)}\right|_{f=f_{\text{inject}}}, & \angle\frac{X_j(s)}{U_{\mathbf{x},j}(s)}\Big|_{f=f_{\text{inject}}} \approx \angle\frac{X_i(s)}{U_{\mathbf{x},i}(s)}\Big|_{f=f_{\text{inject}}}, \\ -\left|\frac{X_i(s)}{U_{\mathbf{x},j}(s)}\right|_{f=f_{\text{inject}}}, & \angle\frac{X_j(s)}{U_{\mathbf{x},j}(s)}\Big|_{f=f_{\text{inject}}} \approx \angle -\frac{X_i(s)}{U_{\mathbf{x},i}(s)}\Big|_{f=f_{\text{inject}}}, \end{cases}} \tag{4.40}$$

where $i, j = 1, 2, 3, \ldots, n$ and n is the degrees of freedom.

Here, there's a few caveats. First of all, what we are doing here is **not** actually actuation diagonalization, instead, what we are trying to do is to align the actuators to the suspension stage's degrees of freedom. But, this alignment has frequency dependency, if the cross-transfer functions are not zero, which is generally true. This means what we cannot minimize actuation cross-coupling at all frequencies, except at a signal frequency, which is actually not that useful. The good thing is that actuation diagonalization is not that important for local damping control, if controls

are all engaged for all DoFs. Actuation cross-coupling will be treated as disturbance from one DoF to another, and will be suppressed by active control. However, actuation coupling will become important when suspension control is not engaged, such as during interferometer locking or observation. Therefore, we need some way to actually diagnalize the actuations and this will be discussed in Sec. 4.3.3.

4.3.3 Frequency dependent matrices

As is discussed in Sec. 4.3.2, a scalar-valued actuation matrix is not useful for diagonalizing the actuators. With a scalar actuation matrix, the actuation can only be diagonalized at a single frequency so actuation signals in one DoF at other frequencies can still other DoFs. To correctly model the actuation system, we have to consider transfer function matrix, with input $\mathbf{U_x}(s) = [U_{x_1}(s), U_{x_2}(s), \dots, U_{x_n}(s)]^T$ and output (displacement) $\mathbf{X}(s) = [X_1(s), X_2(s), \dots, X_n(s)]^T$, where n is the number of DoFs. The input-output relation can be written as

$$\mathbf{X}(s) = \mathbf{P}(s)\mathbf{U_x}(s)\,, \tag{4.41}$$

where the actuation plant $\mathbf{P}(s)$, i.e. the transfer function matrix is

$$\mathbf{P}(s) = \begin{bmatrix} P_{11}(s) & P_{12}(s) & \dots & P_{1n}(s) \\ P_{21}(s) & P_{22}(s) & \dots & P_{2n}(s) \\ \dots & \dots & \ddots & \vdots \\ P_{n1}(s) & P_{n2}(s) & \dots & P_{nn}(s) \end{bmatrix}\,, \tag{4.42}$$

where $P_{ij}(s)$ is the SISO transfer function from the j^{th} actuation $U_j(s)$ to i^{th} displacement $X_i(s)$, and note that $P/\mathbf{P}$ here means plant/path/transfer function, and **not** power spectral density. $\mathbf{P}(s)$ is the plant of the suspension and cannot be changed. The idea of diagonalization is to multiply the actuation signals by a transfer function matrix, i.e. "the" actuation matrix, such that the resultant off-diagonal transfer functions (cross-transfer functions) are zero. Mathematically, we want to design an actuation matrix $\mathbf{D}_{\text{actuation}}(s)$, such that

$$\mathbf{X}(s) = \mathbf{P}(s)\mathbf{D}_{\text{actuation}}(s)\mathbf{U_x}(s)\,, \tag{4.43}$$

and $\mathbf{P}(s)\mathbf{D}_{\text{actuation}}(s)$ is a diagonal transfer function matrix, preferably with diagonal entries $P_{ii}(s)$. It follows that if we set

$$\mathbf{D}_{\text{actuation}}(s) = \mathbf{P}(s)^{-1}\text{diag}\left(\mathbf{P}(s)\right)\,, \tag{4.44}$$

then we can obtain

$$\begin{bmatrix} X_1(s) \\ X_2(s) \\ \vdots \\ X_n(s) \end{bmatrix} = \begin{bmatrix} P_{11}(s) & & & \\ & P_{22}(s) & & \\ & & \ddots & \\ & & & P_{nn}(s) \end{bmatrix} \begin{bmatrix} U_{x_1}(s) \\ U_{x_2}(s) \\ \vdots \\ U_{x_n}(s) \end{bmatrix}\,, \tag{4.45}$$

which is exactly what we want, a diagonalized actuation plant.

There are at least two ways to obtain the actuation matrix using Eqn. (4.44). The first way is to measure the transfer functions $P_{ij}(s)$ as a complex-valued frequency series and then use a transfer function model to fit it using methods discussed in Sec. 4.2.2. With the analytical form of the transfer functions, we can construct the transfer function matrix by evaluating Eqn. (4.44). The calculation of the inverse of a transfer function matrix is rather involved as it's not easy to obtain the analytical form of the inverse. If the matrix is small, we can easily express individual elements of the inverse transfer function matrix as a combination of the elements of the transfer function matrix. However, this can become tedious as the number of DoFs exceed 3. Alternatively, we can use the `inv()` function in MATLAB, which apparently works with transfer function matrix [39]. But, not every one can access MATLAB and we like open-source softwares like Python. So, this brings us to the second way of obtaining the actuation matrix. Instead of modeling the individual transfer functions $P_{ij}(s)$, we take the frequency series of the transfer functions and then compute the inverse of the transfer function matrix at each frequency so we can obtain the actuation matrix as a complex-valued frequency series using Eqn. (4.44). From here, we can use transfer functions models to fit the individual elements of $\mathbf{D}_{\text{actuation}}(s)$ using methods from Sec. 4.2.2.

4.3.4 Examples

Pending

Sensor diagonalization

Actuation diagonalization

Frequency dependent diagonalization

4.4 Inter-calibration

Inter-calibration is defined by the tasks that matches the calibration between two sensors. One example where inter-calibration would be useful is sensor correction (to be discussed in Sec. 4.5.2) where the seismometer readout is used to cancel the seismic noise coupling in the relative displacement sensor readouts from the preisolator. Other usage would be to inter-calibrate the sensor readouts from different stages of a suspension for consistency, diagnosis, sensor fusion, or hierarchical control.

Now, let's say we have two sensor readouts $y_1(t)$ and $y_2(t)$ measuring a common signal $x(t)$. So, assuming that the sensor dynamics is a constant (calibrated), the readouts read

$$y_1(t) = a_1 \left[x(t) + n_1(t)\right] , \tag{4.46}$$

and

$$y_2(t) = a_2 \left[x(t) + n_2(t)\right] , \tag{4.47}$$

where $n_1(t)$ and $n_2(t)$ are the sensor noises of sensing readout $y_1(t)$ and $y_2(t)$, and a_1 and a_2 are constants that represents the calibration error. We cannot measure the calibration errors directly unless we have a third, well-calibrated sensor. So, for simplicity, let's assume $a_1 = 1$, and we only care about relative calibration error. The goal is to find a constant inter-calibration gain $k = k^*$, such that $y_1 - k^* y_2 = n_1 - n_2$, or in other words, $k^* a_2 = 1$. To do so, one might be tempted to minimize the cost function

$$J(k) = \left\langle \left[y_1(t) - k y_2(t)\right]^2 \right\rangle , \tag{4.48}$$

for $t = t_1, t_2, \ldots, T$, where T is a very large number, and $\langle \cdot \rangle$ denotes the average. To see why this might work , we expand Eqn. (4.48)

$$\begin{aligned} \left\langle \left[y_1(t) - k y_2(t)\right]^2 \right\rangle &= \left\langle \left[(1 - k a_2) x(t) + n_1(t) - k a_2 n_2(t)\right]^2 \right\rangle \\ &= (1 - k a_2)^2 \left\langle x(t)^2 \right\rangle + \left\langle n_1(t)^2 \right\rangle + (k a_2)^2 \left\langle n_2(t)^2 \right\rangle , \end{aligned} \tag{4.49}$$

where we assumed $x(t)$, $n_1(t)$, and $n_2(t)$ are uncorrelated, zero-mean signals so the cross-terms are zero. Inspecting Eqn. (4.49), we see that the second term $\left\langle n_1(t)^2 \right\rangle$ is not minimizable so minimizing $J(k)$ is equivalent to minimzing

$$J'(k) = (1 - k a_2)^2 \left\langle x(t)^2 \right\rangle + (k a_2)^2 \left\langle n_2(t)^2 \right\rangle , \tag{4.50}$$

which has a minimum of

$$\min J'(k) = \frac{\left\langle x(t)^2 \right\rangle^2 \left\langle n_2(t)^2 \right\rangle + \left\langle x(t)^2 \right\rangle \left\langle n_2(t)^2 \right\rangle^2}{\left[\left\langle x(t)^2 \right\rangle + \left\langle n_2(t)^2 \right\rangle\right]^2} , \tag{4.51}$$

at

$$k = \frac{1}{a_2} \frac{\left\langle x(t)^2 \right\rangle}{\left\langle x(t)^2 \right\rangle + \left\langle n_2(t)^2 \right\rangle} , \tag{4.52}$$

which is only correct when the SNR is high, i.e. $\langle x(t)^2 \rangle \gg \langle n_2(t)^2 \rangle$. Therefore, this cannot be used for a general case.

Typically, the signal $x(t)$ is localized at certain frequencies so it's To deal with low SNR signals, we consider the Laplace transform (frequency domain) of the signals instead. In Laplace domain, the sensor readouts read

$$Y_1(s) = a_1 X(s) + N_1(s)\,, \tag{4.53}$$

and

$$Y_2(s) = a_2 X(s) + N_2(s)\,. \tag{4.54}$$

Again, for simplicity we let $a_1 = 1$ and we want to find $k = k^* = 1/a_2$. We do so by minimizing the cost function

$$\boxed{J_f(k) = \sum_f |Y_1(s) - kY_2(s)|\, w(f)}\,, \tag{4.55}$$

where f is frequency, $s = i2\pi f$, i is the imaginary number, $w(f)$ is a weighting function. Eqn. (4.55) is no different from Eqn. (4.48) if the weighing function is equal to one at all frequencies $w(f) = 1$. But, we can choose to weight individual data point a each frequency, and it makes sense to choose $w(f)$ such that the value is high at frequencies where the SNR is high. A natural candidate of the weighing function would be the coherence function between $y_1(t)$ and $y_2(t)$, so

$$\boxed{w(f) = C_{y_1 y_2}(f)\,,} \tag{4.56}$$

where $C_{y_1 y_2}(f)$ is the coherence function between the sensor readout $y_1(t)$ and $y_2(t)$. Also note that the definition of coherence function is given in Eqn. (A.4). This is because a high coherence would indicate a high SNR data point, while low coherence would indicate a low SNR data point. Althernatively, we can choose the weighing function to be 1 if the coherence exceeds certain threshold and 0 at other frequencies, i.e.

$$\boxed{w(f) = \begin{cases} 1\,, & C_{y_1 y_2}(f) \geq C_{\text{threshold}}\,, \\ 0\,, & C_{y_1 y_2}(f) < C_{\text{threshold}}\,, \end{cases}} \tag{4.57}$$

where $C_{\text{threshold}}$ is a threshold between 0 to 1. A reasonable value for $C_{\text{threshold}}$ would be 0.5. Using this weighting function effectively masks out the data points with low SNR.

Another interesting cost function to consider is

$$\boxed{J_{\log}(k) = \sum_f \left| \log \frac{1}{k} \frac{Y_1(s)}{Y_2(s)} \right| w(f)\,,} \tag{4.58}$$

since we often have measurements of the ratios between $Y_1(s)$ and $Y_2(s)$ as a transfer function measurement, but not individual $Y_1(s)$ and $Y_2(s)$. Minimizing this cost function effectively says that we optimize k such that the transfer function $\frac{1}{k}\frac{Y_1(s)}{Y_2(s)}$ is 1, which can only happen when $k = k^* = 1/a_2$, i.e. the desired inter-calibration gain.

4.4.1 Example

pending

4.5 Sensor fusion

As is discussed in Sec. 3.2, sensing noise is a limit of the control performance of the suspensions, and it dictates the best possible performance that we can obtain. Therefore, it's very important to minimize the sensor noise as much as possible and as early as possible since controller design depends on the sensing noise. One method that is used in KAGRA for minimizing sensor noise is sensor fusion. Sensor fusion is a technique that combines multiple

sensors into a virtual "super sensor" that has better performance compared to that of the individual sensors. The sensors are measuring the a common signal but has different noise characteristics.

In KAGRA, sensor fusion is used mostly for the preisolator, where active seismic isolation is critical. In particular, LVDTs and Geophones are blended using complementary filters so the displacement readouts at the preisolator benefits from the advantages from the goephones (seismic noise-free and low sensor noise at higher frequencies) and LVDTs (low sensor noise at lower frequencies). Sensor correction is used to remove the seismic noise coupling from the LVDT readouts at the preisolator using the seismometer readout. Strictly speaking, sensor correction also utilizes complementary filter. For optimal performance, sensor correction should be considered as a complementary filter problem. But, sensor correction was not conventionally considered this way during the first attempt in KAGRA [40, 41]. Instead, sensor correction filters were constructed as a high-pass filter using heuristics. So, we will make a distinction between complementary filter problems and sensor correction filters in this section. Also, note that, before sensor any sensor fusion tasks, the sensors involved must be inter-calibrated so to avoid weird calibration mismatch. The inter-calibration techniques are given in Sec. 4.4.

This section is organized as follows. In Sec. 4.5.1, we will discuss some predefined complementary filters and how to design and optimize the filter parameters. In Sec. 4.5.2, we will introduce the purpose of sensor correction and we will discuss how was it design and how can we improve it. In Sec. 4.5.3, we will provide examples on these complementary and sensor correction filters design.

4.5.1 Sensor fusion using complementary filter

Fig. 10 shows the block diagram of a sensor fusion configuration using two filters $H_1(s)$ and $H_2(s)$ to blend two sensors that has different noise characteristics $N_1(s)$ and $N_2(s)$. As shown in the figure, the resultant "super sensor"

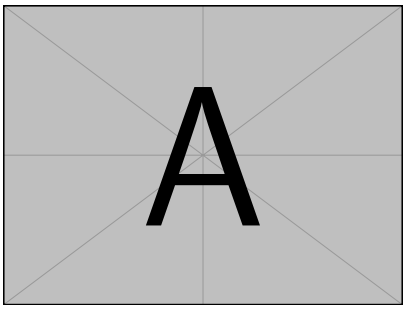

Figure 10: 2-sensor blending using complementary filters

readout has a super sensor noise $N_{\text{super}}(s)$, and it's given by

$$N_{\text{super}}(s) = H_1(s)N_1(s) + H_2(s)N_2(s)\,. \tag{4.59}$$

Note that we omitted the common signal that the sensors are measuring and simply focus on the noises. Since we expect the super sensor readout to contain the original common signal, the filters $H_1(s)$ and $H_2(s)$ must be complementary, as in

$$H_1(s) + H_2(s) = 1\,. \tag{4.60}$$

Alternatively, we can express the second filter $H_2(s)$ as $1 - H_1(s)$, and so the super sensor noise is a function of one filter,

$$N_{\text{super}}(s; H_1(s)) = H_1(s)N_1(s) + [1 - H_1(s)]\, N_2(S)\,. \tag{4.61}$$

The amplitude spectral density of the super sensor noise is then

$$\hat{N}_{\text{super}}(f; H_1(s)) = \left[|H_1(s)|^2\, \hat{N}_1^2(f) + |1 - H_1(s)|^2\, \hat{N}_2^2(f)\right]^{\frac{1}{2}}\,, \tag{4.62}$$

where $\hat{N}(s)$ denotes the amplitude spectral densities.

4.5.2 Sensor correction

4.5.3 Examples

4.6 Time series simulation of a given PSD

4.7 Controller design

5 Suspension Commissioning "Advanced" Methods

5.1 Control optimization methods

5.2 PReQua and modal damping

5.3 Suspension characterization using dynamic mode decomposition

5.4 And More

References

[1] T. Sekiguchi. *Study of Low Frequency Vibration Isolation System for Large Scale Gravitational Wave Detectors*. PhD thesis, Tokyo U., 2016. https://gwdoc.icrr.u-tokyo.ac.jp/cgi-bin/private/DocDB/ShowDocument?docid=4155.

[2] Masayuki Nakano. PReQua. https://gwdoc.icrr.u-tokyo.ac.jp/cgi-bin/private/DocDB/ShowDocument?docid=11861.

[3] Terrence Tsang. SR2 OPLEV diagonalization. https://klog.icrr.u-tokyo.ac.jp/osl/?r=8411.

[4] Mark Barton. Comment to SR2 OPLEV diagonalization. https://klog.icrr.u-tokyo.ac.jp/osl/?r=8419.

[5] Yoshinori Fujii. PRM inspection, one magnet had come off. https://klog.icrr.u-tokyo.ac.jp/osl/?r=8255.

[6] Masayuki Nakano. MICH get locked for 45 min!! https://klog.icrr.u-tokyo.ac.jp/osl/?r=8182.

[7] Fabian Arellano. SRM work: Bottom Filter, Payload and Top Filter limit switches. https://klog.icrr.u-tokyo.ac.jp/osl/?r=15518.

[8] Eric D. Black. An introduction to Pound–Drever–Hall laser frequency stabilization. *American Journal of Physics*, 69(1):79–87, 2001.

[9] Yuta Michimura. What to do with light levers at KAGRA (Translated). https://gwdoc.icrr.u-tokyo.ac.jp/cgi-bin/private/DocDB/ShowDocument?docid=2403.

[10] Yoichi Aso, Yuta Michimura, Kentaro Somiya, Masaki Ando, Osamu Miyakawa, Takanori Sekiguchi, Daisuke Tatsumi, and Hiroaki Yamamoto. Interferometer design of the KAGRA gravitational wave detector. *Phys. Rev. D*, 88:043007, Aug 2013.

[11] Guido Mueller. Beam jitter coupling in advanced LIGO. *Opt. Express*, 13(18):7118–7132, Sep 2005.

[12] Kouseki Miyo. Suspension Commissioning. https://gwdoc.icrr.u-tokyo.ac.jp/cgi-bin/private/DocDB/ShowDocument?docid=12444.

[13] Kouseki Miyo. Seismic noise of KAGRA mine. https://gwdoc.icrr.u-tokyo.ac.jp/cgi-bin/private/DocDB/ShowDocument?docid=10436.

[14] Jameson Graef Rollins. Advanced LIGO Guardian: Overview and Coder's Introduction. https://dcc.ligo.org/LIGO-G1400016/public.

[15] Kouseki Miyo. All of the Vibration Isolation System in KAGRA. https://gwdoc.icrr.u-tokyo.ac.jp/cgi-bin/private/DocDB/ShowDocument?docid=12144.

[16] Katsuhiko Ogata. *Modern Control Engineering*.

[17] Jenő Hetthéssy László Keviczky, Ruth Bars and Csilla Bányász. *Control Engineering*.

[18] Mark Barton and Enzo Tapia. BS Suspension Diagonalization Matrices. https://gwdoc.icrr.u-tokyo.ac.jp/cgi-bin/private/DocDB/ShowDocument?docid=7205.

[19] Mark Barton. SR Suspension Diagonalization Matrices. https://gwdoc.icrr.u-tokyo.ac.jp/cgi-bin/private/DocDB/ShowDocument?docid=7663.

[20] Tsang Terrence Tak Lun. Sensing Matrices for Optical Levers (OpLev) of the KAGRA Main Optics. https://gwdoc.icrr.u-tokyo.ac.jp/cgi-bin/private/DocDB/ShowDocument?docid=12874.

[21] T. Akutsu et al. Vibration isolation systems for the beam splitter and signal recycling mirrors of the KAGRA gravitational wave detector. *Class. Quant. Grav.*, 38(6):065011, 2021.

[22] Tomotada Akutsu et al. Compact integrated optical sensors and electromagnetic actuators for vibration isolation systems in the gravitational-wave detector KAGRA. *Rev. Sci. Instrum.*, 91(11):115001, 2020.

[23] Enzo Tapia. Lecture 4: Use of OSEMs for VIS. https://gwdoc.icrr.u-tokyo.ac.jp/cgi-bin/private/DocDB/ShowDocument?docid=8873.

[24] Takafumi Ushiba et al. Cryogenic suspension design for a kilometer-scale gravitational-wave detector. *Class. Quant. Grav.*, 38(8):085013, 2021.

[25] Simon Zeidler. Length-Sensing OpLevs for KAGRA. https://gwdoc.icrr.u-tokyo.ac.jp/cgi-bin/private/DocDB/ShowDocument?docid=5788.

[26] Kazuhiro Agatsuma. Optical lever for KAGRA. https://gwdoc.icrr.u-tokyo.ac.jp/cgi-bin/private/DocDB/ShowDocument?docid=2396.

[27] Aaron Barzilai, Tom VanZandt, and Tom Kenny. Technique for measurement of the noise of a sensor in the presence of large background signals. *Review of Scientific Instruments*, 69:2767–2772, 07 1998.

[28] R. Sleeman, A. Wettum, and J. Trampert. Three-channel correlation analysis: A new technique to measure instrumental noise of digitizers and seismic sensors. *Bulletin of the Seismological Society of America*, 96:258–271, 2006.

[29] Ryutaro Takahashi. Status of ACC development II. https://gwdoc.icrr.u-tokyo.ac.jp/cgi-bin/private/DocDB/ShowDocument?docid=11774.

[30] Kouseki Miyo. Trillium Compact 120-SV1 Information. https://gwdoc.icrr.u-tokyo.ac.jp/cgi-bin/private/DocDB/ShowDocument?docid=8440.

[31] Tsang Terrence Tak Lun. Noise Estimation using Correlation Methods. https://kontrol.readthedocs.io/en/latest/tutorials/noise_estimation_using_correlation_methods.html.

[32] Gaël Varoquaux. Mathematical optimization: finding minima of functions. https://scipy-lectures.org/advanced/mathematical_optimization/#a-review-of-the-different-optimizers.

[33] Wikipedia contributors. Gradient descent — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Gradient_descent&oldid=1019572955, 2021. [Online; accessed 23-May-2021].

[34] Wikipedia contributors. Simulated annealing — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Simulated_annealing&oldid=1017509035, 2021. [Online; accessed 23-May-2021].

[35] SciPy. Optimization and root finding (scipy.optimize. https://docs.scipy.org/doc/scipy/reference/optimize.html.

[36] Wikipedia contributors. Nelder–mead method — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Nelder%E2%80%93Mead_method&oldid=1018086682, 2021. [Online; accessed 23-May-2021].

[37] Wikipedia contributors. Differential evolution — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Differential_evolution&oldid=1016542156, 2021. [Online; accessed 23-May-2021].

[38] Wikipedia contributors. Preconditioner — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Preconditioner&oldid=1012689908, 2021. [Online; accessed 23-May-2021].

[39] MathWorks. Invert models - MATLAB inv - MathWorks. https://www.mathworks.com/help/control/ref/lti.inv.html.

[40] Fujii Yoshinori. A quick test for sensor-correction-gain tuning. https://klog.icrr.u-tokyo.ac.jp/osl/?r=10164.

[41] Fujii Yoshinori. Comment to A quick test for sensor-correction-gain tuning. https://klog.icrr.u-tokyo.ac.jp/osl/?r=10179.

Appendices

A Spectral density

A.1 Cross power spectral density and power spectral density

The cross power spectral density (CPSD) between a signal $x(t)$ and $y(t)$ is defined as

$$P_{xy}(f) \equiv \int_{-\infty}^{\infty} R_{xy}(\tau) e^{-i2\pi f\tau}\, d\tau\,, \tag{A.1}$$

where f is the frequency, i is the imaginary number, and

$$R_{xy}(\tau) \equiv \lim_{T\to\infty} \frac{1}{T} \int_0^T x(t)y(t-\tau)\, dt \tag{A.2}$$

is the correlation function between $x(t)$ and $y(t)$. The power spectral density (PSD) of a single signal $x(t)$ is then defined as the cross power spectral density between the signal $x(t)$ and itself, i.e. $P_{xx}(f)$.

A.2 Amplitude spectral density

The amplitude spectral density of a signal x(t) is simply defined as

$$\hat{x}(f) \equiv [P_{xx}(f)]^{\frac{1}{2}}\,. \tag{A.3}$$

A.3 Coherence

The coherence between two signals $x(t)$ and $y(t)$ is then defined as

$$C_{xy}(f) \equiv \frac{|P_{xy}(f)|^2}{P_{xx}(f)P_{yy}(f)}\,. \tag{A.4}$$

A.4 Sum of signals

If we have two signals that can be expressed by superposition of other signals, i.e. $x(t) = x_1(t) + x_2(t)$ and $y(t) = y_1(t) + y_2(t)$, then the cross spectral density between $x(t)$ and $y(t)$ is

$$\begin{aligned} P_{xy}(f) &= \lim_{T\to\infty} \frac{1}{T} \int_0^T \int_{-\infty}^{\infty} \left[x_1(t) + x_2(t)\right] \left[y_1(t-\tau) + y_2(t-\tau)\right] e^{-i2\pi f\tau}\, dt \\ &= P_{x_1y_1}(f) + P_{x_1y_2}(f) + P_{x_2y_1}(f) + P_{x_2y_2}(f)\,. \end{aligned} \tag{A.5}$$

A.5 Uncorrelated signals

If $x(t)$ and $y(t)$ are uncorrelated signals, then their correlation function $R_{xy}(\tau)$ and $R_{yx}(\tau)$ is 0, and so is their cross power spectral density. For a signal that can be expressed a sum of two uncorrelation signals, $z(t) = x(t) + y(t)$, then its PSD is

$$\begin{aligned} P_{zz}(f) &= P_{xx}(f) + P_{xy}(f) + P_{yx}(f) + P_{yy}(f) \\ &= P_{xx}(f) + P_{yy}(f)\,. \end{aligned} \tag{A.6}$$

A.6 Transfer function

The transfer function with input $x(t)$ and output $y(t)$ is can be estimated by

$$P_{X\to Y} = \frac{P_{yx}(f)}{P_{xx}(f)}\,. \tag{A.7}$$

B Mathematical optimization

B.1 The arg min function

For some cost function $J(\theta)$, where θ is some parameters, we say θ^* is the optimizer such that

$$J(\theta^*) = \min_{\theta} J(\theta). \tag{B.1}$$

Another way to express θ^* is

$$\theta^* = \arg\min_{\theta} J(\theta), \tag{B.2}$$

which is the mathematical way of saying θ^* is the argument in the set $\{\theta\}$, that minimizes J.

B.1.1 Example

For a function $f(x) = x^2 + 1$, and $x \in \mathbb{R}$, $\min_{x\in\mathbb{R}} f(x) = 1$ and and $x^* = \arg\min_{x\in\mathcal{R}} f(x) = 0$.