LASER INTERFEROMETER GRAVITATIONAL WAVE OBSERVATORY
- LIGO -
CALIFORNIA INSTITUTE OF TECHNOLOGY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Technical Note **LIGO-T1000458-v6** Date: 5/06/2011

Monolithic Quad Noise Prototype System Identification and Model Fitting

Brett Shapiro

Contents

1 Introduction

1.1 Purpose and Scope

This paper summarizes the system identification methods and model fitting techniques applied to the monolithic quad noise prototype at LASTI. The model used is a version of Mark Barton's Matlab model that includes off-diagonal moments of inertia and lateral blade spring stiffnesses. It does not handle violin modes or thermal noise. A few minor changes to the model were needed to handle the differences between fibers and wires and some specific LASTI issues.

Background on the model can be found in the documents listed in 'Related Documents' section below. T1100163 contains the Matlab source code for the model fitting algorithm. T020205 describes how the basic model was derived and a manual on how to use it. T0800096 includes a history on the evolution of the model in addition to specific details on how wires and flexure points are handled. T040072 defines most of the parameters used in the model. T1100183 includes a list summarizing the system identification tests planned for the production quads.

1.2 Related Documents and References

T1100163: Quad Pendulum Model Fitting Code
T020205: Models of the Advanced LIGO Suspensions in Mathematica
T080096: Wire Attachment Points and Flexure Corrections
T040072: Pendulum Parameter Descriptions and Naming Convention
T1100183: Quad System Identification, Calibration, and Commissioning Tests

1.3 Version History

v1, 3 August 2010: Submitted to DCC
v2, 17 August 2010: Cleans up the model files and parameters.
v3, 29 March 2011: It adds transfer function measurements to the System Identification section. The Related Documents list was expanded to include the model fitting source code at T1100163, and T1100183 which lists the tests planned for the production quads.
v4, 30 March 2011: The model fitting section is modified from BFGS a algorithm to Gauss-Newton, which is usually more appropriate for least squares optimizations. A section on estimating model error was added. -v3 and -v4 constitute a general rewrite.
v5, 6 May 2011: Added Appendix A deriving the inertial impact of the new 'h' parameters.
v6, 6 May 2011: Corrected typos.

2 System Identification Methods

The basic goal of system identification here is to measure as much about the quad's dynamics as possible in order to predict control performance, cross-couplings and the values of the as-built parameters. Due to the relatively large number of physical parameters with uncertainty, the more data collected the better the chance of finding realistic values for these parameters. The data comes in the form of measured resonant frequencies and transfer functions.

2.1 Resonant Frequency Measurements

Resonant frequencies measurements have the advantage that good sensor alignment and calibration is not needed. Further, the resonant frequencies contain the vast majority of the information needed to identify the model parameters. The disadvantage is that they provide little information about cross-couplings. On the LASTI quad they were measured for four different, undamped, configurations to collect as much information as possible (see the illustration in Figure 1):

1. Free quad
2. Triple suspension hanging from a locked top mass
3. Double suspension hanging from locked a upper intermediate mass
4. Single suspension hanging from a locked penultimate mass

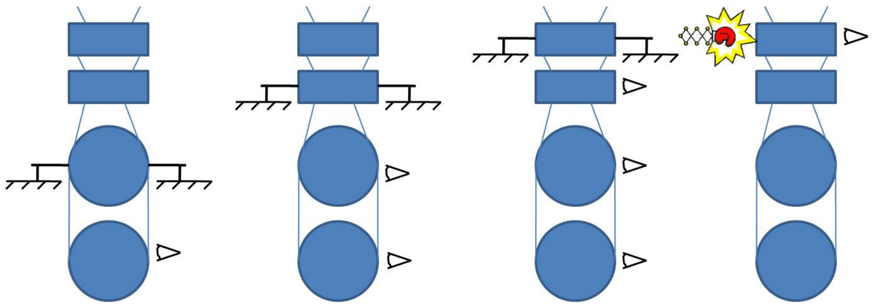

Figure 1: The four locked vs. free states of the quad from which measurements were taken for the system identification. The state on the left shows measurements being taken on the single pendulum of the test mass while the PUM was locked. The next shows measurements on the double pendulum of the lower two stages, then the triple pendulum of the lower three stages, and finally the full quad. The sketched in eye balls indicate where measurements should be taken. The fist indicates where excitations should be applied with the OSEMs to measure transfer functions.

All resonances were measured with a resolution at least as fine as 2 mHz. Locking various stages down changes the resonant frequencies of the free stages and localizes the effect of the uncertain parameters relating to those stages. These four configurations expand the number of resonances from 24 free quad resonances to a total of 60 (24 free + 18 triple + 12 double + 6 single). In theory even more configurations could be collected from free stages above locked stages, e.g. top mass free with UIM locked. These other configuration are not practical however since resonances are extremely sensitive to wire tension and locking a stage down undoubtedly changes wire tension above due to shifts in the position of the masses.

The free quad resonant frequencies were extracted from top mass to top mass transfer functions (these serve Section 2.2 as well). Measurements on the other three configurations were done with the OSEMs on the UIM and PUM and an optical lever on the test mass. These triple, double, and single pendulum resonant frequencies were extracted from power spectra where the excitation was simply from air currents and seismic noise (more than enough).

Nominally, the OSEMs on the UIM and PUM stages measure only X, YAW, and PITCH. However, since many of the flags only partially occluded their respective LED beams, some photodiodes were sensitive Y, Z, and ROLL as well. On the LASTI quad these misalignments were not intentional, however for the purposes of resonant frequencies measurements on production quads, intentional misalignments of well aligned flags will be needed to capture all resonant frequencies. The reaction chain was locked down when using the UIM and PUM OSEMs to ensure that all measurements were of the main chain. In the single pendulum case the optical lever was limited to measurements of pitch and yaw. The x resonances were captured as well through x-pitch coupling, but the others were not visible.

A total of 50 out of the possible 60 resonant frequencies were measured on the LASTI quad. The highest frequency roll mode of the triple suspension configuration was not visible to either the OSEMs or the optical lever. In the double configuration only the optical lever was used since it was not realized the PUM OSEMs could provide off-axis measurements until it was too late. Hence only X, PITCH, and YAW modes were obtained missing 6 of the possible 12. For the single pendulum case, the optical lever was the only option, so again X, PITCH, and YAW were the only visible degrees of freedom, obtaining 3 out of a possible 6. It was then discovered later that since the PUM was locked using the glass silica earthquake stops, the 3 measured single pendulum frequencies might be contaminated by the compliance of the Viton in the stops. Thus only 47 out of a possible 60 frequencies were used in the Matlab model fitting code. It is recommended that the single pendulum resonances be measured before the Teflon transport stops are removed from the PUM since these will lock the PUM more effectively.

2.2 Transfer Function Measurements

Transfer functions have the advantage that they contain cross-coupling information. The disadvantage is that the OSEMs must be well aligned and have sufficient dynamic range.

As stated earlier, transfer functions were measured from the top mass to the top mass for the free quad configuration. This yielded a six-by-six transfer function matrix which contains the drive from every top mass degree of freedom to every top mass degree of freedom. These top mass transfer functions were measured with a resolution no more coarse than 2 mHz. The resolution was motivated by the localization of the resonances rather than the quality of the transfer function itself.

Transfer functions were also measured from the UIM X, YAW, and PITCH degrees of freedom to all six top mass degrees of freedom, also for the free quad configuration. Simultaneous measurements to the UIM and PUM would be ideal, but the alignment of the flags was not trusted. The UIM electronics on their own did not supply sufficient current to the UIM OSEMs to make clean measurements. Consequently, the top mass reaction chain coil driver was swapped with the UIM coil driver. Since the free quad resonances were already localized from the top mass transfer functions a more coarse 5 mHz resolution was used here.

No transfer functions were measured with excitations from the PUM or test mass. The PUM has insufficient dynamic range, and the electrostatic drive cannot be run in air.

The use of these measurements in the model fitting code was purely to increase the accuracy of the rotational inertia values, in particular the off-diagonal inertias of the top mass and UIM. Thus, only the magnitude information above 5 Hz was used, as this is where the inertias dominate over stiffness parameters. The quality of the magnitude information is only as good as the calibration of the sensors and actuators. In order to estimate the electronic gain for the top mass OSEMs, the model was pre-fit by running it through the fitting code with just the resonance frequencies. This creates a good model for x, yaw, and z. The electronic gain was then estimated by comparing the scaling factor between the magnitude of the measured transfer functions and the magnitude of the model transfer functions. The measurements were then scaled dividing them by this value (165.6) to put them in SI units.

3 Model Fitting Method

A fitting routine was written around a version of Mark Barton's Matlab model that varies certain parameters such as inertia, d's, etc in order to minimize the error between the model and the measured data. The algorithm used is a least squares minimization of the model errors using a Gauss-Newton algorithm, which is an approximation to Newton's Method (2nd order). I have explored other algorithms as well including, gradient descent (1st order), Monte Carlo (0th order), and the Broyden-Fletcher-Goldfarb-Shanno (BFGS, 2nd order) algorithm which is a different approximation to Newton's Method. My experience is that the gradient descent and Monte Carlo methods are either extremely slow, not robust, or both when applied to this application. The pseudo-Newton Methods, being of higher order work faster, more accurately, and with greater robustness. The Gauss-Newton methods has given slightly better performance over BFGS because it is better suited for least squares methods since the method converges exactly to Newton's method as the error approaches zero.

3.1 Newton's Method

Newton's method is a second order method that uses both the first and second derivatives to quickly find the minimum of a function. A function $f(\mathbf{x})$ is approximated with a second order Taylor series as

$$V(\theta_{k+1}) \approx f(\theta_k) + \mathbf{g}_k^T(\theta_{k+1} - \theta_k) + \frac{1}{2}(\theta_{\theta+1} - \theta_k)^T \mathbf{H}_k(\theta_{k+1} - \theta_k) \qquad (1)$$

In this application the scaler V would be the sum of the squared error between the model and the measured data. The dependent variable θ is a column vector of length n, and represents the parameters we wish to fit such as inertia and the d's. $\mathbf{g}$ is the column vector gradient, length n, of V with respect to θ. $\mathbf{H}$ is the n by n Hessian matrix of V. The Hessian is the symmetric matrix of second derivatives with respect to θ. The subscript k represents the current iteration number of the algorithm.

Similarly, the gradient can be approximated with a first order Taylor series.

$$\nabla V_{k+1} = \mathbf{g}_{k+1}^T \approx \mathbf{g}_k^T + (\theta_{k+1} - \theta_k)^T \mathbf{H}_k \tag{2}$$

An approximate solution for the minimum can be found at θ_{k+1} by setting the gradient in Eq. 2 to zero.

$$\theta_{k+1} = \theta - \mathbf{H}_k^{-1} \mathbf{g}_k \tag{3}$$

Iterating over this calculation will typically converge to a minimum in relatively few steps. It does not guarantee that the minimum is global.

3.2 Gauss-Newton Algorithm

One of the major difficulties with Newton's method is the calculation of $\mathbf{H}^{-1}$. If a single calculation is needed to find the error, than n calculations are needed to find the gradient, and n^2 calculations are needed to find the Hessian. Then the Hessian needs to be inverted. Thus, computation quickly gets very slow with square of the number of parameters being solved. In this application each calculation includes compiling the model. Thus, if we have ten parameters we want to fit, we need to compile the model at least 100 times just to get the Hessian matrix for a single iteration of the routine. The Gauss-Newton method is an approximation to this method, among others, that sacrifices some accuracy on the Hessian matrix for an improvement on computation time.

The algorithm depends on knowledge of the gradient vector and an approximate Hessian matrix to run. One way to easily estimate the gradient vector is to use the finite difference method. Each single parameter is stepped by a small amount, the model recompiled, and the change in error calculated. The list generated for each parameter is the gradient vector. Care must be taken to use an appropriate step size. Too small a step and roundoff errors dominate, too large and the estimate is not a faithful representation of the gradient at the current point.

The Hessian approximation can be derived by writing the cost V in (1) exactly as a function of the model residuals, the measured difference between the physical quad and the modeled quad. In (4) $\mathbf{r}$ represents the length m column vector of residuals.

$$V(\theta) = \frac{1}{2} \mathbf{r}(\theta)^T \mathbf{r}(\theta) \tag{4}$$

Taking the derivative with respect to the model parameters, θ, gives the gradient vector, $\mathbf{g}$.

$$\mathbf{g}(\theta) = \mathbf{J}(\theta)^T \mathbf{r}(\theta) \tag{5}$$

where $\mathbf{J}$ is the Jacobian matrix of the residuals with respect to the parameters.

$$J_{ij} = \frac{\delta r_i}{\delta \theta_j} \tag{6}$$

$$1 < i < m \tag{7}$$

$$1 < j < n \tag{8}$$

Note that we have derived a second way to calculate the gradient vector. The Jacobian is calculated using finite differences in a manner similar to that described for the first gradient estimation method. In practice whichever gradient calculation method is the most convenient at a particular time is the one used.

The Hessian matrix is derived by taking another derivative with respect to the parameters.

$$\mathbf{H}(\theta) = \mathbf{J}(\theta)^T \mathbf{J}(\theta) + \mathbf{L}(\theta)\mathbf{r} \tag{9}$$

$\mathbf{L}$ is some matrix that includes many second derivatives. Since this matrix is multiplied by $\mathbf{r}$, in the neighborhood of the optimal solution, where $\mathbf{r} = 0$, the Hessian can be approximated using only first order derivatives from the Jacobian.

$$\mathbf{H}(\theta) \approx \mathbf{J}(\theta)^T \mathbf{J}(\theta) \tag{10}$$

The extent of the 'neighborhood' is not well defined a priori. It is possible for Gauss-Newton to fail to converge if the starting model parameters are too far from the true parameters. Nonetheless this method has always converged for the LASTI quad, even given poor estimates for many of the parameters. Thus, based on this experience, if the modeled inertias start within 10% and the $d's$ start within ± 5 mm, then the algorithm will have no trouble converging to the true solution.

A sufficient condition for convergence to a unique global solution is that $\mathbf{J}$ have full column rank everywhere in the region of exploration inside the parameter space. A necessary, but not sufficient, condition for full column rank is that $m \geq n$, i.e. at least as many measurements as there are parameters being fit. It is difficult to numerically guarantee that $\mathbf{J}$ has full column rank everywhere, and not obvious how to show it analytically. However, confidence that this requirement is met can be greatly enhanced by using as many measurements as possible, particularly those that couple well to the parameters being tuned by the algorithm. The rank, or better the condition number (ratio of maximum and minimum singular values), of $\mathbf{J}$ can also be tested for various points in the parameter space, including those the algorithm passes through.

The routine runs through the following steps:

1. Start with an initial guess of the model parameters.
2. For each k, until the stopping condition is met, Calculate the Jacobian matrix $\mathbf{J}_k$.
3. Calculate the search direction $\mathbf{d}_k = -\mathbf{J}_k^+ \mathbf{r}_k$ $(= (\mathbf{J}_k^T \mathbf{J}_k)^{-1} \mathbf{g}_k$, for full column rank $\mathbf{J})$.
4. Do a line search in the direction of $\mathbf{d}_k$ to find an appropriate step size α_k so that $\theta_{k+1} = \theta_k + \alpha_k \mathbf{d}_k$.
5. Go to step 2.

3.3 Implementing Gauss-Newton on the Data

3.3.1 Residual Calculation

One key aspect of the implementation of any fitting routine is how the error is defined. There are many ways to define the error between the model and the data, each of which will yield different results; unless obviously the uncertainties are small enough that the routine converges to exactly the as built parameters, but this is not realistic.

Since most of the data involves measured resonant frequencies, an obvious choice is to compare them to the resonances of the model. Then one still needs to decide how to compare the resonances. Insight comes from the fact that percent errors on physical parameters tend to cause percent errors on mode frequencies. For instance quadrupling the length of a single pendulum halves its frequency. Thus, a shift of say 10 mHz at 0.5 Hz represents much more parameter error than 10 mHz at 5 Hz. So it makes sense to use some sort of percent error on the resonant frequencies, rather than absolute error.

Consequently, resonance residuals are calculated with the following equation, where ω represents frequency in radians/second. These units were chosen instead of Hz because these are the units Matlab returns, and dividing by 2π just wastes time.

$$r_i = \sum_{i=1}^{m} \frac{\omega_{i,measured} - \omega_{i,model}}{\omega_{i,measured}} \tag{11}$$

The remaining data fed into the model comes from magnitudes of the measured transfer functions from 7 Hz to 10 Hz. The residual terms for this data are calculated as the average percent difference between the modeled and measured magnitudes.

3.3.2 Estimating the Expected Parameter Solution Error

Given imperfect measurements we can expect the algorithm to converge to a solution with some error. The minimum size of this error can be estimated using a mathematical device called the Fisher Information Matrix (FIM). A book entitled *Optimal Measurement Methods for Distributed Parameter System Identification* by Dariusz Uciński gives a good description of the FIM.

Conveniently, the FIM is largely dependent on the system Jacobian matrix J, which is already calculated by the Gauss-Newton algorithm as described in Section 3.2. The only other information needed is the variance of the measurement. If C_ϵ is the covariance matrix of the measurement error (diagonal in this case), then the covariance matrix of the parameter error, C_θ, is given by the inverse of the Fisher information matrix M

$$C_\theta = M^{-1} \tag{12}$$

$$M = J^T C_\epsilon^{-1} J \tag{13}$$

or more directly where '+' indicates the Moore-Penrose pseudoinverse,

$$C_\theta = J^+ C_\epsilon J^{+T} \tag{14}$$

If it is desired to apply weighting factors to certain measurements, for example if some measurements are trusted more than others, then C_θ is adjusted by

$$C_\theta = J^+ W C_\epsilon W J^{+T} \tag{15}$$

Where W is a diagonal matrix that weights the relative importance of each measurement. For this application the transfer function data were scaled down by a factor of 10 relative to the resonances, otherwise they tend to dominate the total error while containing the greatest uncertainty. The standard deviation of the resonance measurement error was taken to be the measurement resolution, 2 mHz. The transfer function magnitudes were guessed to be accurate to 3%, but the error is possibly greater.

The minimum standard deviation of the parameter error is the square root of the diagonal elements of C_θ. It is important to realize that the FIM yields only the *minimum* parameter error. The actual error can be much larger. Further, unknown errors in the model (e.g. those parameters that are assumed correct but not, or incorrect measurements) will increase the error. Further still, the FIM depends on the calculation of J, which itself is an estimation based on the assumption of a good model.

3.3.3 Reducing the Model for the Various Sys-Id States

It was necessary to adjust the model to reflect the various sub-pendulum sys-id states. This is a very simple, almost trivial procedure. Locking a mass has the effect of forcing its position and velocity to zero. An easy way to reflect this in the model is to compile the model as usual and simply delete the relevant rows and columns in the resulting state space matrices. For example, when the test mass is free by itself all the rows and columns representing the top three masses are deleted.

In this way the model could be compared to the measured resonances of these sub-pendulums as part of the fitting routine's error calculation.

4 Changes to the Model Compilation Code

A couple of changes to Mark Barton's Matlab script were needed to suit the monolithic quad at LASTI.

4.1 Fiber Flexure Correction and d's

The metal wires have a straight forward way of handling the effective flexure points and wire lengths. In the parameter file, called 'quadopt...', only physical values are entered (at least in the Matlab version). The wire lengths are referenced from clamp break-off to clamp break-off, and the d's represent the vertical distance between the center of mass and the clamp break-off at their respective stages. In the compilation script, called 'ssmake...', these physical values are converted to effective values shifted by the flexure correction of the respective wire. This flexure correction represents the distance from the wire clamp break-off to the theoretical bending point.

When it comes to fibers there is no longer a clear wire break-off point. Instead the fiber cross section changes gradually over many millimeters, away from the weld. Consequently, using the current state of the model one would have to define a false 'physical' break-off. The flexure correction would then be the distance from this imaginary point to the theoretical bending point.

To simplify the definition of these fiber parameters, the physical break-off was done away with entirely in the LASTI version of the model. The LASTI model now references the fiber to the centers of mass of the PUM and test mass, eliminating the $flex_3$ parameter. The effective the fiber length becomes the vertical separation between the PUM and test mass centers of mass subtracted by d_3 and d_4. The center of mass separation is a real, well controlled number during the welding process of 602 mm. Additionally there is no longer a distinction between 'physical' and 'effective' d's for d3 and d4 since they now reference the centers of mass directly as well.

The model now sets the fiber length to

$$l_3 = 0.602 - d_3 - d_4 \tag{16}$$

4.2 X Offset of the UIM Center of Mass

When the monolithic quad was first suspended all the masses immediately pitched to one side. A horizontal offset of 0.262 mm on the ears of the test mass (extracted from measured violin modes, Marielle suggested a theoretical value of 0.23 mm from the test mass polishing dimensions) explains about half the observed pitch. Likely a discrepancy between the two wires in the PUM wire loop is the cause of the other half. Consequently, it was necessary to shift 700 or 800 grams from the front of the UIM to the back. This caused a large, non-negligible, horizontal offset in the center of mass at this stage somewhere between 6 mm and 8 mm. This offset resulted in significant coupling between the PITCH and Z dynamics.

Because of this offset it was necessary to add additional off-diagonal terms in the mass matrix defined by the Matlab model. To this end, a new parameter called h1 representing this horizontal offset was defined in the 'quadopt...' parameter file. The letter 'h' was chosen since this is a horizontal offset. The number 1 represents the UIM. The new off-diagonal mass matrix term is called Im1yz. The parameter is defined by the following equation

$$Im1yz = h1 * m1 \tag{17}$$

where m1 is the mass of the UIM. The 'Im' in the parameter name comes from the fact that this inertia term is between a rotational inertia and a translational mass. The number 1 represents the UIM, and the 'yz' indicates that the term is between rotation around the y axis and translation along the z axis. Appendix A derives the relation between Imyz and h.

The parameter is also defined in the 'quadopt...' parameter file simply so that all inertia terms end up in the same data structure call 'pend'. It could just as easily be computed in the 'ssmake...' file.

The 'symbexport...' file is called by the 'ssmake...' file to compile all the effective parameters into state space matrices. A modification to the mass/inertia matrix 'km' in 'ssmake...' just below the call to 'symbexport...' was made to include this new inertia term.

For consistency similar parameters were added to all the stages of the quad, however they are set to zero. The 0.262 mm ear offset at the test mass was actually included, but it has negligible impact on the dynamics and could easily be left out.

5 Results

The estimated parameter error is given in section 5.1 below. The full list of model parameters is given in Section 5.2. A comparison of measured and modeled top mass transfer functions is given in Section 5.3

5.1 Estimated Parameter Error

The fitting code predicted, given the assumed measurement errors, that the d's are correct to ±0.5 mm. This value seems reasonable for an individual d. However, pairs of d's can shift with little impact. For example,

$$\frac{m2 + m3}{m3} d2 + d3 \approx \text{constant} \tag{18}$$

This is because the upper and lower d's on a given stage influence the dynamics in the same way, adjusted for the different weights carried by each. Consequently, the ±0.5 mm applies to scaled summed pairs of d's rather than individual d's. In retrospect it would be more appropriate to fix one of the d's in each stage. For example the only d's in the top mass and UIM that should have significant error are the lower ones defined by the blade spring tips.

The diagonal inertia terms it claims are good to ±0.1%, which seems overly optimistic. The measured transfer function magnitudes determine these and I probably was too optimstic about the calibration here. The off-diagonal inertia errors are stated in absolute terms of $\pm 0.001\,\text{kgm}^2$. This could be accurate, but I do not really have a feel for it.

Overall the model fitting routine created a model that predicts the dynamics very well. The worst error in mode frequency is no more than a couple percent (out of 47 measured modes). The greatest uncertainty is likely in cross-coupling magnitudes such as pitch-roll, because these depend on well calibrated transfer functions. The calibration here is likely worse than the assumed 3%.

5.2 Parameter List

This lists contains all the model parameters in the adjusted model.

Note, that $flex_3$, d_3, d_4, and l_3 have been redefined. See Section 4.

The model fitting routines adjusted all the inertias including off-diagonal terms, all the d's, the blade spring stiffnesses, and the fiber axial stiffness.

$$mn = 21.924 \tag{19}$$
$$Inx = 0.462385286064711 \tag{20}$$
$$Iny = 0.069939022580041 \tag{21}$$
$$Inz = 0.468244632768256 \tag{22}$$
$$Inxy = -0.029881327248797 \tag{23}$$
$$Inyz = 0.0000439 \tag{24}$$
$$Inzx = 0.00198 \tag{25}$$
$$hn = 0 \tag{26}$$
$$Imnyz = pend.mn * pend.hn \tag{27}$$
$$\tag{28}$$
$$m1 = 22.01 \tag{29}$$
$$I1x = 0.508404670585712 \tag{30}$$
$$I1y = 0.088657492199992 \tag{31}$$
$$I1z = 0.520036661955096 \tag{32}$$
$$I1xy = -0.017414470113595 \tag{33}$$
$$I1yz = 0 \tag{34}$$
$$I1zx = 0 \tag{35}$$
$$h1 = 0.00775 \tag{36}$$
$$Im1yz = m1 * h1 \tag{37}$$
$$\tag{38}$$
$$m2 = 39.68 \tag{39}$$
$$I2x = 0.584099336903170 \tag{40}$$
$$I2y = 0.427609985942287 \tag{41}$$
$$I2z = 0.416616168785308 \tag{42}$$
$$I2xy = 0 \tag{43}$$
$$I2yz = 0 \tag{44}$$
$$I2zx = 0 \tag{45}$$
$$h2 = 0 \tag{46}$$
$$Im2yz = m2 * h2 \tag{47}$$
$$\tag{48}$$
$$m3 = 39.595 \tag{49}$$
$$I3x = 0.560801969547827 \tag{50}$$
$$I3y = 0.402992079867132 \tag{51}$$
$$I3z = 0.419208912160651 \tag{52}$$
$$I3xy = 0 \tag{53}$$
$$I3yz = 0 \tag{54}$$
$$I3zx = 0 \tag{55}$$

$$h3 = 0.000262 \tag{56}$$
$$Im3yz = m3 * h3 \tag{57}$$
$$\tag{58}$$
$$ln = 0.445 \tag{59}$$
$$l1 = 0.311 \tag{60}$$
$$l2 = 0.342147535653900 \tag{61}$$
$$rn = (1.1/2) * 10^-03 \tag{62}$$
$$r1 = (0.711/2) * 10^-03 \tag{63}$$
$$r2 = (0.635/2) * 10^-03 \tag{64}$$
$$r3 = -1 \tag{65}$$
$$r3n = 0.0008/2 \tag{66}$$
$$r3m = 0.0004/2 \tag{67}$$
$$r3 = pend.r3n \tag{68}$$
$$nl = 0.015 \tag{69}$$
$$nwn = 2 \tag{70}$$
$$nw1 = 4 \tag{71}$$
$$nw2 = 4 \tag{72}$$
$$nw3 = 4 \tag{73}$$
$$Yn = 2.12000 * 10^+11 \tag{74}$$
$$Y1 = 2.12000 * 10^+11 \tag{75}$$
$$Y2 = 2.12000 * 10^+11 \tag{76}$$
$$Y3 = 7.2 * 10^10 \tag{77}$$
$$twistlength = 0 \tag{78}$$
$$d3tr = 1.0000 * 10^-03 \tag{79}$$
$$d4tr = 1.0000 * 10^-03 \tag{80}$$
$$sn = 0 \tag{81}$$
$$su = 0.003 \tag{82}$$
$$si = 0.003 \tag{83}$$
$$sl = 0.015 \tag{84}$$
$$nn0 = 0.250 \tag{85}$$
$$nn1 = 0.090 \tag{86}$$
$$n0 = 0.200 \tag{87}$$
$$n1 = 0.060 \tag{88}$$
$$n2 = 0.140 \tag{89}$$
$$n3 = 0.1775 \tag{90}$$
$$n4 = 0.17025 \tag{91}$$
$$n5 = 0.17025 \tag{92}$$
$$sin = (nn1 - nn0)/ln \tag{93}$$

$$si1 = (n1 - n0)/l1 \tag{94}$$
$$si2 = (n3 - n2)/l2 \tag{95}$$
$$si3 = (n5 - n4)/l3 \tag{96}$$
$$kcn = 1429.45595883388 \tag{97}$$
$$kc1 = 1648.69334920402 \tag{98}$$
$$kc2 = 2382.96853678021 \tag{99}$$
$$kxn = 1.0 * 10^5 \tag{100}$$
$$kx1 = 1.0 * 10^5 \tag{101}$$
$$kx2 = 0.8 * 10^5 \tag{102}$$
$$ffn = 0.807 \tag{103}$$
$$ff1 = 0.641 \tag{104}$$
$$ff2 = 0.608 \tag{105}$$
$$kffn = 1 + ffn * tan(asin(sin)) \tag{106}$$
$$kff1 = 1 + ff1 * tan(asin(si1)) \tag{107}$$
$$kff2 = 1 + ff2 * tan(asin(si2)) \tag{108}$$
$$mn3 = mn + m1 + m2 + m3 \tag{109}$$
$$m13 = m1 + m2 + m3 \tag{110}$$
$$m23 = m2 + m3 \tag{111}$$
$$cn = (ln^2 - (nn1 - nn0)^2)^0.5/ln \tag{112}$$
$$c1 = (l1^2 - (n1 - n0)^2)^{0.5}/l1 \tag{113}$$
$$c2 = (l2^2 - (n3 - n2)^2)^{0.5}/l2 \tag{114}$$
$$c3 = (l3^2 - (n5 - n4)^2)^{0.5}/l3 \tag{115}$$
$$flexn = sqrt(2 * ((1/4) * pi * rn^4) * Yn/mn3/g) * cn^{3/2} \tag{116}$$
$$flex1 = sqrt(4 * ((1/4) * pi * r1^4) * Y1/m13/g) * c1^{3/2} \tag{117}$$
$$flex2 = sqrt(4 * ((1/4) * pi * r2^4) * Y2/m23/g) * c2^{3/2} \tag{118}$$
$$flex3 = 0 \tag{119}$$
$$bend1 = g * (m1 + m2 + m3)/(2 * kx1) \tag{120}$$
$$bend2 = g * (m2 + m3)/(2 * kx2) \tag{121}$$
$$\tag{122}$$
$$dm = (1 - 0.9609906717849997)/1000 - flexn \tag{123}$$
$$dn = (1 - 0.735501089622135)/1000 - flex1 + bend1 \tag{124}$$
$$d0 = (1 + 1.393824700702)/1000 - flex1 \tag{125}$$
$$d1 = (1 - 0.115974101777530)/1000 - flex2 + bend2 \tag{126}$$
$$d2 = 0.3/1000 - flex2 \tag{127}$$
$$d3 = (4.3 - 1.750244189175000)/1000 \tag{128}$$
$$d4 = (4.3 - 2.045280917079000)/1000 \tag{129}$$
$$\tag{130}$$
$$l3 = 0.602 - d3 - d4 \tag{131}$$
$$kw3 = 32963.3 \tag{132}$$

5.3 Modeled and Measured Transfer Functions

The section contains the six measured top mass transfer functions against the adjusted model transfer functions. For brevity, a before and after result is shown only for pitch, since this started off by far with the most error. A top mass transfer function is measured by driving and measuring the response of the same degree of freedom at the top mass.

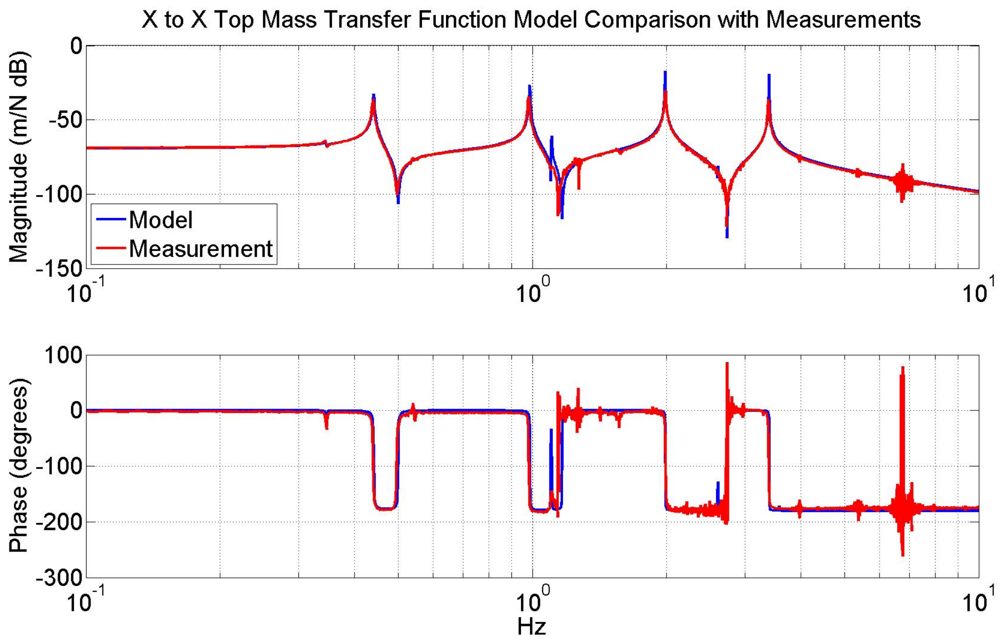

Figure 2: Model fit result against the measured top mass transfer function.

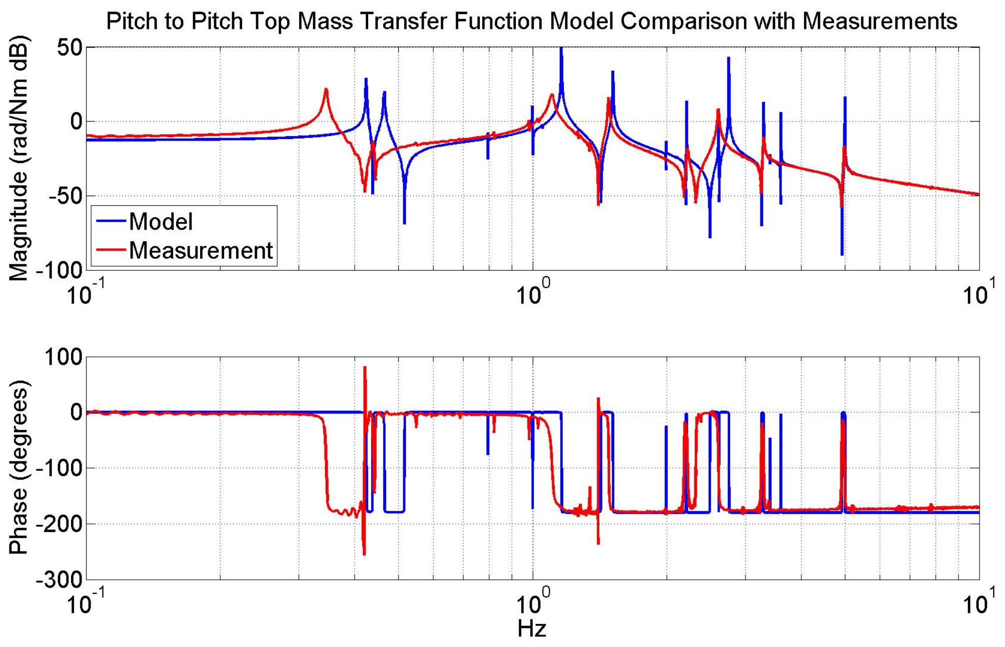

Figure 3: Original monolithic quad model against the measured top mass transfer function.

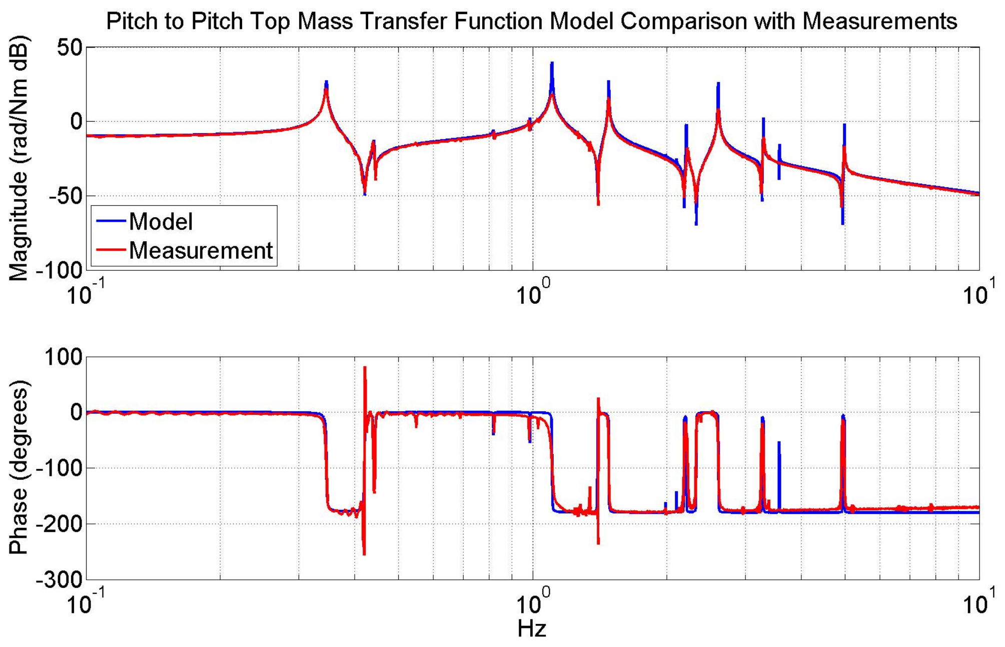

Figure 4: Model fit result against the measured top mass transfer function.

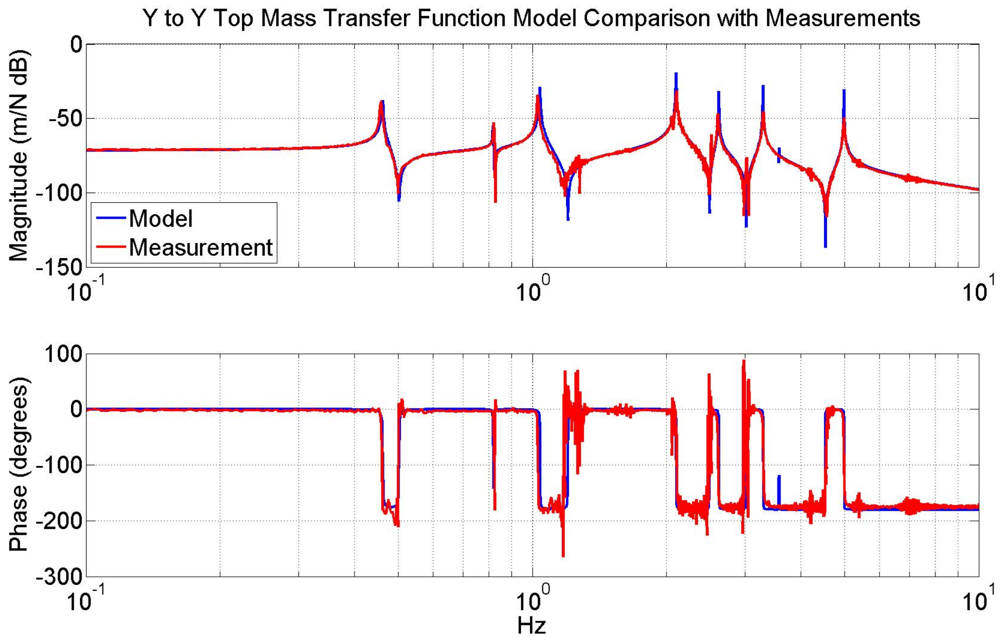

Figure 5: Model fit result against the measured top mass transfer function.

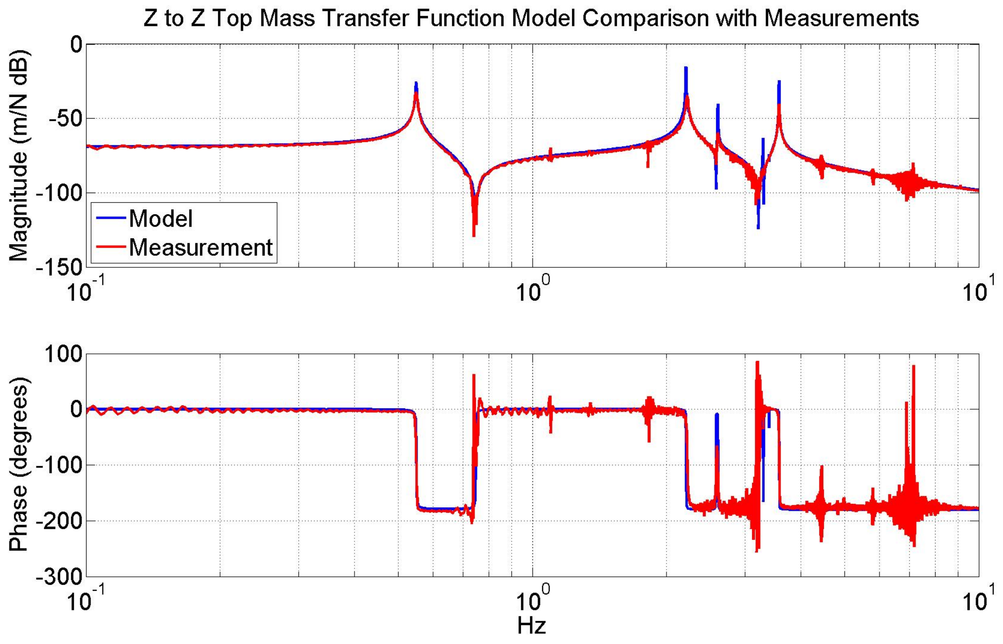

Figure 6: Model fit result against the measured top mass transfer function.

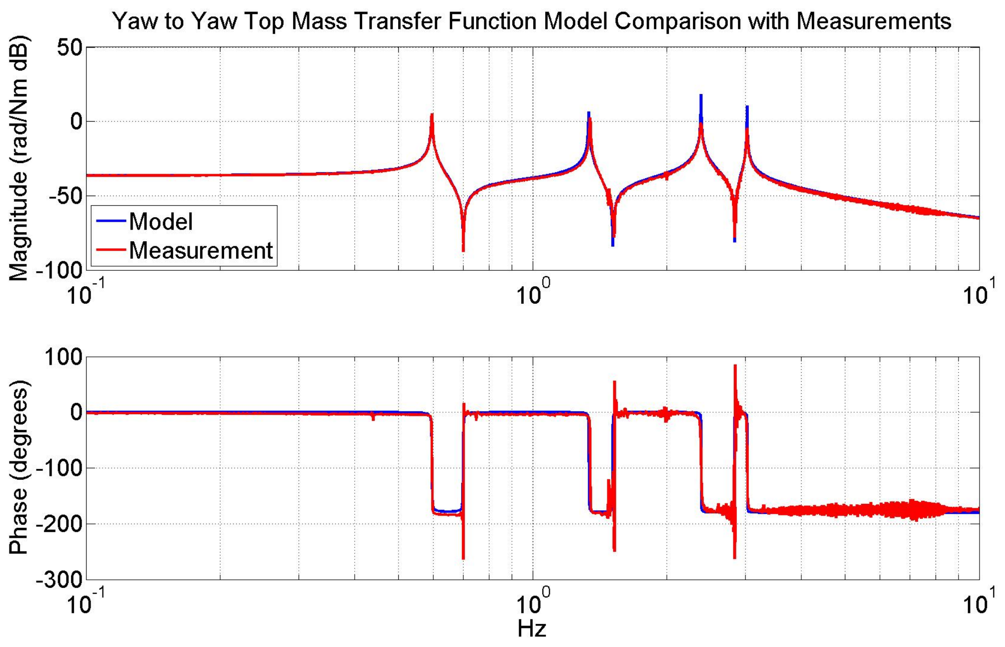

Figure 7: Model fit result against the measured top mass transfer function.

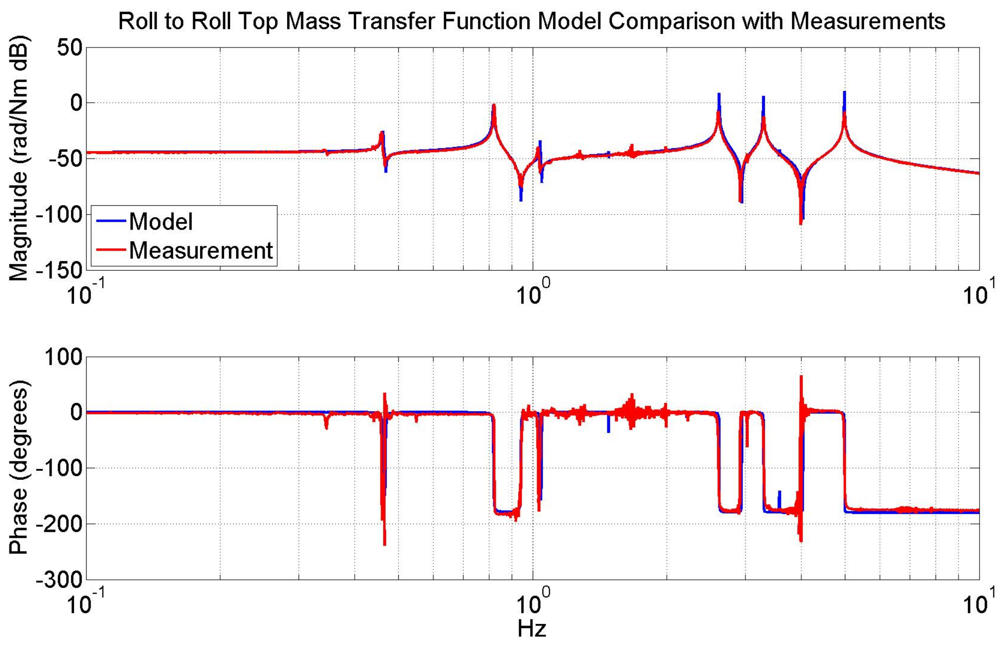

Figure 8: Model fit result against the measured top mass transfer function.

A Deriving the Influence of 'h' on Inertia

This appendix derives the effect the h parameters have on the equations of motion. Translating the center of mass only effects the inertia matrix, since mass by definition is inertial. The stiffness matrix is not influenced. The change in the inertia matrix is derived by translating the coordinate system of the respective mass away from the center of mass by the distance h along the x axis. Physically this means the OSEMs measure the same point on the mass regardless of where the center of mass is actually located.

Figure 9 and the equations below consider the simple case of the vertical (z) and rotation (θ) DOFs of a free mass floating in space. The results are summarized by matrix equations (135) and (143).

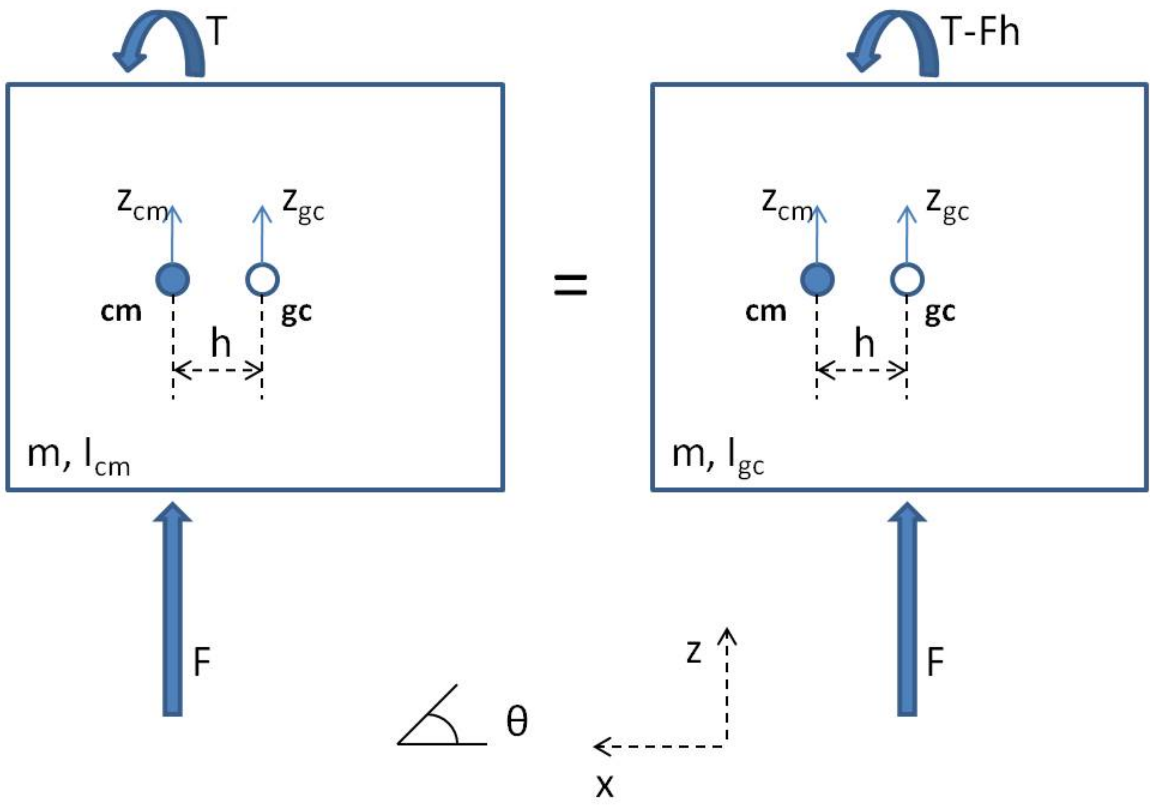

Figure 9: Free body diagram of a free mass with a center of mass offset from the geometric center by h. The left side shows the external forces and torques about the center of mass (cm). The right shows them about the geometric center (gc).

The left side of Figure 9 references the force F and torque T to the center of mass (cm), which is offset from the geometric center (gc) by h. The equations of motion about this point are given by (133) and (134). Equation (135) groups these two equations into matrix notation. Note the diagonal 2×2 inertia matrix in (135).

$$m\ddot{z}_{cm} = F \tag{133}$$

$$I_{cm}\ddot{\theta} = T \tag{134}$$

$$\begin{bmatrix} m & 0 \\ 0 & I_{cm} \end{bmatrix} \begin{bmatrix} \ddot{z_{cm}} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} F \\ T \end{bmatrix} \tag{135}$$

The right side of Figure 9 translates the reference point to the geometric center. The vertical equation of motion of this point is

$$z_{gc} = z_{cm} - h\theta \tag{136}$$

Removing z_{cm} by plugging in (133) yields

$$\ddot{z}_{gc} = \frac{F}{m} - h\theta \tag{137}$$

Then both sides are multiplied by m.

$$m\ddot{z}_{gc} + mh\theta = F \tag{138}$$

The rotation equation about the geometric center is

$$I_{cm}\ddot{\theta} = T - Fh \tag{139}$$

F is removed from this equation by plugging in (138).

$$I_{cm}\ddot{\theta} = T - (m\ddot{z}_{gc} + mh\theta)h \tag{140}$$

Rearranging terms gives the result

$$mh\ddot{z}_{gc} + I_{gc}\ddot{\theta} = T \tag{141}$$

where

$$I_{gc} = I_{cm} + mh^2 \tag{142}$$

Note that (142) is also given by the parallel axis theorem. In matrix notation the equations of motion about the geometric center are given by

$$\begin{bmatrix} m & mh \\ mh & I_{gc} \end{bmatrix} \begin{bmatrix} \ddot{z_{gc}} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} F \\ T \end{bmatrix} \tag{143}$$

Note that the 2×2 inertia matrix is symmetric positive-definite in both cases. This is always true for mechanical systems. As an FYI, stiffness matrices are also always symmetric positive-definite, unless some masses are floating freely in space in which case it is symmetric positive-semidefinite.

Note also the units in each corner of the inertia matrix. The top left corner, representing translational dynamics, has units of kg. The bottom right corner, representing rotational dynamics, has units of kgm^2. The top right and bottom left has units of kgm, indicating these inertia terms represent a coupling term that is truly something between translation and rotation.