LASER INTERFEROMETER GRAVITATIONAL WAVE OBSERVATORY

LIGO Laboratory / LIGO Scientific Collaboration

LIGO- T020011-00-D *ADVANCED LIGO* 14 Jan 2003

Suspension model comparisons

Mark Barton, Calum Torrie

Distribution of this document:
SWG

This is an internal working note
of the LIGO Project.

1 Introduction

1.1 Purpose and Scope

This document summarizes results of comparisons made between Calum Torrie's Matlab model of the GEO triple suspension as well as the version for a quad pendulum by him and Ken Strain, and the corresponding Mathematica models by Mark Barton.

It also summarizes some comparisons made between the Mathematica model and analytic expressions for the thermal noise.

1.2 Version History

1/14/2002 – Release 00 draft

1/23/02 – Release 00 draft 2. Extra data from triple.

5/8/02 – Release 00 draft 3.

1/13/03 – Release 00

2 Matlab vs Mathematica Model Comparison

2.1 Description of Models

2.1.1 The Matlab models

The basis of the Matlab model for the triple pendulum is described in Calum Torrie's thesis. It has three rigid bodies connected by massless wires with longitudinal elasticity. Each of the rigid bodies has the usual six DOFs, but a symmetric system is assumed so that many cross-couplings vanish and the problem decomposes into independent vertical, longitudinal-pitch, transverse-roll, and yaw subsystems, with only three or six DOFs each. The blade springs are not modeled independently but are allowed for by adding their compliance to that of the associated wires. A version for a quad pendulum was then produced by generalizing the matrix elements appropriately.

2.1.2 The Mathematica models

The Mathematica models will be described more fully in a separate document. They are implemented in the symbolic algebra program Mathematica. By exploiting the symbolic algebra capability to do the hard work of deriving the various couplings it is able to be somewhat more ambitious and only a subset of it can be usefully compared to the Matlab. However where it does overlap it has been made as similar as possible, e.g., with identical names for analogous parameters.

The model contains three or four rigid body elements with six DOFs each for the pendulum masses, as in the Matlab. All the names and values of dimensions, masses, moments of inertia are the same as in the Matlab. Similarly, the model contains wires with the same names and values for lengths, attachment points, longitudinal elasticities etc as in the Matlab. (There is some subtlety about what "the same length" means – see below.)

In addition the model contains rigid body elements, also with six DOFs, representing the tips of the blade springs. As explained in more detail below, these new DOFs can be ignored to a considerable extent because the new modes that come with them have frequencies in a completely different range (much higher) than the 18 or 24 low-frequency modes of the three or four pendulum masses that the Matlab calculates.

The model can also include the bending elasticity of the wire. However because this is a minor correction, takes significant computing time, and does not correspond to anything in the Matlab, it has been made optional and was not turned on for any of the Matlab comparison tests. This amounts to taking the results after "Stage 0" of the Mathematica code (see comments therein).

2.2 Differences Requiring Consideration

2.2.1 Structure

The biggest difference between the two models is structural. Both ultimately arrive at a matrix of couplings between various DOFs and take the eigenvalues and eigenvectors of that matrix to determine the mode shapes and frequencies. However in the Matlab, the matrix elements were derived manually and entered in symbolic form, whereas in the Mathematica the symbolic algebra capability is used to derive them on the fly. For efficiency reasons it is desirable to do key steps numerically, so that formulae that can be directly compared to the Matlab are not generated.

Also, while the default version of the Mathematica model is symmetric, it was designed with a view to including asymmetries later. Thus it ignores any symmetry in the model specification and always solves the full 10x6=60 DOF problem (7x6=42 for the triple).

So unfortunately the only output that can easily be compared is the numeric eigenvalues and eigenvectors.

2.2.2 Wire numbers

In the Mathematica model, each of the wires is specified individually and the fact that there are only two rather than four at the top level of the quad is explicit. In the Matlab, the derivation of the matrix elements assumes four wires at each level, and the case of two wires is allowed for by making the elasticity for the top stage half that of the physical wires and putting the separation in the x direction to zero.

2.2.3 Wire lengths

In the real physical system, each wire starts with some unstretched length and then comes to a slightly greater stretched length when put under load. The details of this are allowed for in the derivation of the Matlab matrix elements, and the final formulae are all in terms of stretched lengths. (This is convenient in the design phase where one is principally interested in the final assembled state of the system.) By contrast, the on-the-fly calculation procedure used in the Mathematica naturally takes unstretched lengths as input and derives the equilibrium state of the system from them by minimizing the total potential energy. To make the models more comparable, a facility was added to the Mathematica whereby stretched lengths can be given as input and the appropriate unstretched lengths are derived. (This is impractically difficult for arbitrary asymmetrical models, so symmetry was assumed. This is of course adequate for comparisons with the Matlab, but it needs to be kept in mind because it is the only place the Mathematica makes such assumptions. Moreover it is convenient enough that users will want to leave it switched on for routine design work. Thus they need to remember to switch it off if/when they come to explore the effect of asymmetries.)

2.2.4 Provision for blades

The Mathematica models the blades explicitly and in some detail. By contrast, the Matlab matrix elements are derived for a system without blades and the blades are allowed for by simply adding their compliances to those of the associated wires. This is a fairly rough-and-ready approximation that was resorted to because of the mounting complexity of the algebra, but as it will be seen, it turns out to be adequate for many purposes. However it makes detailed comparison between the two models rather difficult.

Two different approaches were used on the Mathematica side to restore some degree of comparability:

(i) Disable the blades entirely in the Mathematica and move their compliance into the wires as in the Matlab. This has the advantage that the same physical system (a pure compound pendulum with unusually stretchable wires) is being modeled, but the disadvantage that it doesn't exercise the provision for blades in the Mathematica at all.

(ii) Leave the blades in but tweak them to make adding the compliances as valid as possible. (The issue implicit here is that compliances only add linearly (as assumed) if the corresponding springs act along the same line. This is not the case in the actual physical system – the working direction of the blade springs is vertical whereas most of the wires that attach to blade springs are significantly diagonal.)

2.2.5 Blade parameters

The only point of correspondence between the blades in the two models is the compliance in the working direction, which of course has been made the same. The Mathematica model has a large number of additional blade parameters, including blade-tip masses and moments of inertia and the transverse and torsional compliances. Currently the values of these are mostly guesstimates that will need refinement based on experimental data. However the only modes that are sensitive to these values are new, high-frequency modes. The guesstimates for the transverse and torsional stiffnesses are already very high, several orders of magnitude greater than those in the working direction, so that the fundamental 24 modes of the pendulum masses that the Matlab calculates are pretty much independent of them. To further increase comparability the blade-tip masses were set very small (around 0.01 g).

2.2.6 Finding comparable definitions of wire elasticities

The Matlab and the Mathematica both used symbols kwn, kw1, kw2 and kw3 to represent wire elasticities (without any corrections from blades). However the Mathematica defined them strictly on a per-wire basis, whereas the Matlab defined them per-side. This introduced a factor of 2 difference between the top level in each model and the lower level(s). Since both these approaches were convenient in their respective contexts, no effort was made to harmonize the definitions. Instead, compensating factors were inserted in expressions to be compared.

2.2.7 Number of wires per blade

In the triple system being modelled, there is one blade for each wire in all cases, and the Matlab allows for this by adding each blade compliance to the associated wire compliance throughout. This is easy to emulate in the Mathematica by immobilizing the blades and making the same correction to the wire compliances. In the quad system, at the second and third levels from the top, there are two wires on each blade. The Matlab allows for this by applying one of two different fudges depending on the sub-problem. In the vertical, transverse/roll and yaw sub-problems, it makes the obvious generalization and adds half of the compliance of each two-wire blade to each of the associated wires. However for the longitudinal/pitch modes this gives unrealistically low pitch frequencies, because differential stretching of the two wires on each blade is significant. For this case it is better to leave the blade compliances out entirely. This means that there is no single Mathematica model that can match all the Matlab mode frequencies for the quad. Therefore for comparison purposes two separate models were created, one adding the blade compliances in throughout, and the other leaving them out entirely.

2.3 Procedure

2.3.1 Versions

The versions of the Mathematica models used were 2.3 for the quad and v1.2 for the triple.

2.3.2 Identifying the correct Matlab matrix elements

To minimize the possibility of manual error, the "Matlab" values for comparison were actually generated within Mathematica, by cutting and pasting the Matlab formulae for the matrix elements into a Mathematica notebook and applying the same parameter substitutions as used internally by the Mathematica model. For the triple model, the correct matrix elements are given in appendix C2 of Calum Torrie's thesis. Unfortunately at some point a number of typos found their way into some versions of the model. These were found by Wen Ho and many versions were corrected but there are still uncorrected versions (even some marked as corrected!) in circulation. Specifically,

in `k11`, `2*k2*s2^2*c2/I1y` should be `2*k2*s2^2*c2^2/I1y`

in `j13`, `m23*n2*si2/I1x/c2*(n3*si2/l2-d2*c2/l2)` should be `m23*g*n2*si2/I1x/c2*(n3*si2/l2-d2*c2/l2)`,

in `j41`, `m23*g*d1/m2/l2` should be `m23*g*c2*d1/m2/l2` and `+m23*g*si2*n2/m2/l2/c2` should be `+m23*g*si2*n2/m2/l2`,

in `h23`, `h3/m3` should be `h3/m2`.

Similarly, the quad model matrix elements in versions in circulation when this study was begun contained numerous errors. Fortunately most of them had no effect on the mode frequencies for typical parameters because they involve swaps of quantities with the same numeric values, and the remainder had only small effects (e.g., ˜1% for `k34`). However for the purposes of this study, all the elements were rederived by generalizing the (corrected) triple matrix elements using Mathematica.

2.3.3 Validation of input numeric values

The numeric parameters of the physical pendulum in the Matlab model command file are localized in a single file (`quadopt.m`, `triple.m`, or `jbr.m` depending in the model) and corresponds to the substitution list `defaultvalues` (and the numeric version `constval` derived from it) in the Mathematica. The Matlab code was traced and the values of key variables were compared to the values in constval to ensure that all input values had been correctly entered, and all derived values had been correctly computed.

2.3.4 Disabling additional corrections in the Matlab

In the Matlab quad model, there was provision for modeling ribbons by using adjusted values of the parameters `d3` and `d4` (the vertical distances of the fiber attachment points below and above the centers of mass of the intermediate mass and optic respectively) when calculating the transverse-roll modes. (Ribbons are stiff in the transverse direction and so have a different effective flexure point.) This was disabled.

2.4 Quad Model Results

2.4.1 Blades switched off, compliance added to that of wires throughout

This test implements in Mathematica a system with immobile blades and extra-compliant wires, which is equivalent to the system implied by the provision for blades used in the Matlab for the transverse/roll, vertical and yaw subproblems. The Mathematica v2.2 quad model was used as a basis and the compliance of the blades was transferred to the wires by applying the following set of overrides (“`lockedblades`”) to the values in `defaultvalues`.

```
overrides = {
      mbeu -> scale[10^(-5)],
      mbei -> scale[10^(-5)],
      mbel -> scale[10^(-5)],
      kwnusual -> (Yn*An)/ln,
      kw1usual -> (Y1*A1)/l1,
      kw2usual -> (Y2*A2)/l2,
      kw3usual -> (Y3*A3)/l3,
      kbuzusual -> mnb*(2*N[Pi]*ufcn)^2,
      kbizusual -> m1b*(2*N[Pi]*ufc1)^2,
      kblzusual -> m2b*(2*N[Pi]*ufc2)^2,
      kwn -> recipadd[kwnusual, kbuzusual],
      kw1 -> recipadd[kw1usual, kbizusual/2],
      kw2 -> recipadd[kw2usual, kblzusual/2],
      kw3 -> kw3usual,
      kbuz -> 10^5*kbuzusual,
      kbiz -> 10^5*kbizusual,
      kblz -> 10^5*kblzusual,
      kbuy -> 10^5*kbuzusual,
      kbiy -> 10^5*kbizusual,
      kbly -> 10^5*kblzusual,
      kbux -> 10^5*kbuzusual,
      kbix -> 10^5*kbizusual,
                kblx -> 10^5*kblzusual
        };
```

The usual expressions for the wire elasticities are assigned to symbols `kwnusual`, …, `kw3usual` to have them available without letting them feed into the rest of the calculation. Similarly the usual expression for the blade elasticities are assigned to `kbuzusual`, etc. The values assigned to `kwn`,…, `kw3` include the blade elasticities and the values assigned to `kbuz` etc are set very large (i.e., stiff). The blade tip masses `mbeu`, `mbei` and `mbel` are set very small.

The "Matlab" values for all four subproblems, longitudinal/pitch, transverse/roll, vertical and yaw, were obtained by applying the following substitutions for the wire elasticities to the Matlab expressions for the matrix elements and taking the eigenvalues of the resulting matrices:

```
        kkkLB = {
                kn -> recipadd[kwnusual,kbuzusual],
                k1 -> recipadd[2 kw1usual, kbizusual],
                k2 -> recipadd[2 kw2usual, kblzusual],
                k3 -> 2 kw3usual
        };
```

This exactly replicates the algorithm and the results of the real Matlab code for the transverse/roll, vertical and yaw (but not longitudinal/pitch) subproblems, and it has been checked that the two calculations agree. Note that there is no factor 2 in front of `kwnusual`, reflecting the fact that there is only one wire per side at the top level. The "Matlab" values for longitudinal/pitch were also calculated with `kkkLB` and should agree with the Mathematica values from this test but not the real Matlab code.

The results for this test are given in Table 1.

Longitudinal/Pitch		Transverse/Roll		Vertical		Yaw	
Matlab	Math.	Matlab	Math.	Matlab	Math.	Matlab	Math.
3.0320	3.0320	22.5664	22.5664	16.6701	16.6701	4.1881	4.1881
1.9881	1.9881	4.8823	4.8830	3.6905	3.6905	2.8446	2.8446
1.5261	1.5262	3.2820	3.2938	2.1864	2.1864	1.5485	1.5485
1.0256	1.0257	2.8710	2.8562	0.6191	0.6192	0.6711	0.6711
0.8535	0.8532	2.0045	2.0050				
0.7699	0.7697	1.0390	1.0368				
0.4374	0.4373	0.8387	0.8403				
0.4082	0.4083	0.4343	0.4333				

Table 1: Mathematica and Matlab results for quad pendulum with blade compliance added to that of wires throughout.

2.4.2 Blades switched off entirely

This test implements in Mathematica a physical system with immobile blades but realistic wires. This is the physical system implied by the calculation used for the longitudinal/pitch subproblem in the Matlab. (The reason for not adding in the compliance of the blades is that it seriously distorts the mode shapes and frequencies for modes which involve relative pitch motion at the levels where there is a single blade and two wires on each side, i.e, between the top and second masses and between the second and intermediate masses. In other modes involving stretching of the wires, the blade and the two wires in each group move in phase as a single compound spring, but for the pitch modes, the wires move in antiphase, with the spring, which is very stiff in pitch, essentially immobile.) In the Mathematica v2.2 quad model, the compliance of the blades was transferred to the wires by applying the following set of overrides ("`lockedbladesLP`") to the values in `defaultvalues`:

```
overrides = {
      mbeu -> scale[10^(-5)],
      mbei -> scale[10^(-5)],
      mbel -> scale[10^(-5)],
      kbuzusual -> mnb*(2*N[Pi]*ufcn)^2,
      kbizusual -> m1b*(2*N[Pi]*ufc1)^2,
      kblzusual -> m2b*(2*N[Pi]*ufc2)^2,
      kbuz -> 10^5*kbuzusual,
      kbiz -> 10^5*kbizusual,
      kblz -> 10^5*kblzusual,
      kbuy -> 10^5*kbuzusual,
      kbiy -> 10^5*kbizusual,
      kbly -> 10^5*kblzusual,
      kbux -> 10^5*kbuzusual,
      kbix -> 10^5*kbizusual,
      kblx -> 10^5*kblzusual
};
```

The "Matlab" values in the table were obtained by applying the following substitutions for the wire elasticities to the Matlab expressions for the matrix elements and taking the eigenvalues of the resulting matrices:

```
kkkLBLP = {
      kn -> kwn,
      k1 -> 2 kw1,
      k2 -> 2 kw2,
      k3 -> 2 kw3
};
```

Note that there is no factor 2 in front of `kwnusual`, reflecting the fact that there is only one wire per side at the top level. The "Matlab" values for longitudinal/pitch should and do agree with the real Matlab code. The others should agree with the Mathematica results in this test but not the output of the real Matlab code.

The results for this test are given in Table 2.

Longitudinal/Pitch		Transverse/Roll		Vertical		Yaw	
Matlab	Math.	Matlab	Math.	Matlab	Math.	Matlab	Math.
3.0320	3.0321	64.1518	64.1000	48.3877	48.3493	4.2091	4.2091
2.2740	2.2724	51.4532	51.4140	31.8795	31.8562	2.8536	2.8536
1.9860	1.9860	18.0753	18.0743	14.8956	14.8913	1.5487	1.5488
1.6363	1.6359	6.4724	6.4684	6.3740	6.3705	0.6725	0.6724
1.1114	1.1109	2.9259	2.9147				
1.0230	1.0232	1.5540	1.5509				
0.4916	0.4916	1.0042	1.0098				
0.4238	0.4239	0.5178	0.5177				

Table 2: Mathematica and Matlab results for quad pendulum with blade compliance added to that of wires throughout.

2.4.3 Blades not disabled but rotated into the direction of the wires

This test implements in Mathematica a system with blades enabled but with their working direction rotated so as to be in line with the associated wires. This is the closest system with interesting correspondences to the Matlab that doesn’t involve disabling the blades in the Mathematica. It was implemented in the Mathematica with the following overrides (“`diagblades`”):

```
overrides = {
      mbeu -> scale[10^(-5)],
      mbei -> scale[10^(-5)],
      mbel -> scale[10^(-5)],
      vertblades -> False
};
```

As before, the blade tip masses are set very small. Also the vertblades option is set to False, which engages a whole lot of stuff to make the springs line up with the wires. The results are given in Table 3. The “Matlab” values are the final output ones, being a hybrid of the previous two cases, with blade compliance added to wires for all modes except longitudinal/pitch. The agreement in vertical is perfect and all the other modes are within a few percent, which is as good as can be expected given that the systems being modeled are only approximately the same.

Longitudinal/Pitch		Transverse/Roll		Vertical		Yaw	
Matlab	Math.	Matlab	Math.	Matlab	Math.	Matlab	Math.
3.0320	3.0231	22.5664	22.5664	16.6701	16.6701	4.1881	4.2016
2.2740	2.2048	4.8823	4.8829	3.6905	3.6905	2.8446	2.8497
1.9860	1.9818	3.2820	3.2936	2.1864	2.1864	1.5485	1.5481
1.6363	1.5969	2.8710	2.8562	0.6191	0.6191	0.6711	0.6719
1.1114	1.0229	2.0045	2.0050				
1.0230	0.9007	1.0390	1.0368				
0.4916	0.4388	0.8387	0.8402				
0.4238	0.4096	0.4343	0.4331				

Table 3: Mathematica and Matlab results for the quad pendulum with blades rotated into the direction of the wires.

2.5 Triple model results

In the case of the triple model, the same provision for the blades is used for all four subproblems. (This is because, in the triple, at the levels where there are four wires, there are also four springs, one per wire, so that stretch of a wire is always accompanied by downward motion of the associated spring, even in pitch modes.) This means that only one test case (“`lockedblades`”) is required:

```
      overrides = {
            mbeu -> scale[10^(-4)],
            mbel -> scale[10^(-4)],
            kw1usual -> (Y1*A1)/l1,
            kw2usual -> (Y2*A2)/l2,
            kw3usual -> (Y3*A3)/l3,
            kbuzusual -> (m1/2)*(2*N[Pi]*ufc1)^2,
            kblzusual -> (m2/4)*(2*N[Pi]*ufc2)^2,
            kw1 -> recipadd[kw1usual, kbuzusual],
            kw2 -> recipadd[kw2usual, kblzusual],
            kw3 -> kw3usual,
            kbuz -> 10^4*kbuzusual,
            kblz -> 10^4*kblzusual,
            kbuy -> 10^4*kbuzusual,
            kbly -> 10^4*kblzusual,
            kbux -> 10^4*kbuzusual,
            kblx -> 10^4*kblzusual
      };
```

The results for this test are given in Table 4.

Longitudinal/Pitch	Transverse/Roll	Vertical	Yaw

Matlab	Math.	Matlab	Math.	Matlab	Math.	Matlab	Math.
3.5815	3.5814	52.3948	52.3948	36.5561	36.5561	3.0714	3.0713
2.6656	2.6655	3.3391	3.3390	3.8739	3.8738	1.5583	1.5583
2.4043	2.4040	2.4797	2.4796	0.9542	0.9541	0.3879	0.3879
1.3550	1.3550	1.3553	1.3553				
0.6140	0.6140	0.8461	0.8459				
0.5118	0.5115	0.6054	0.6054				

Table 4: Mathematica and Matlab results for wire-only triple pendulum with blade compliance added to that of wires.

3 Thermal noise comparisons

3.1 Procedure

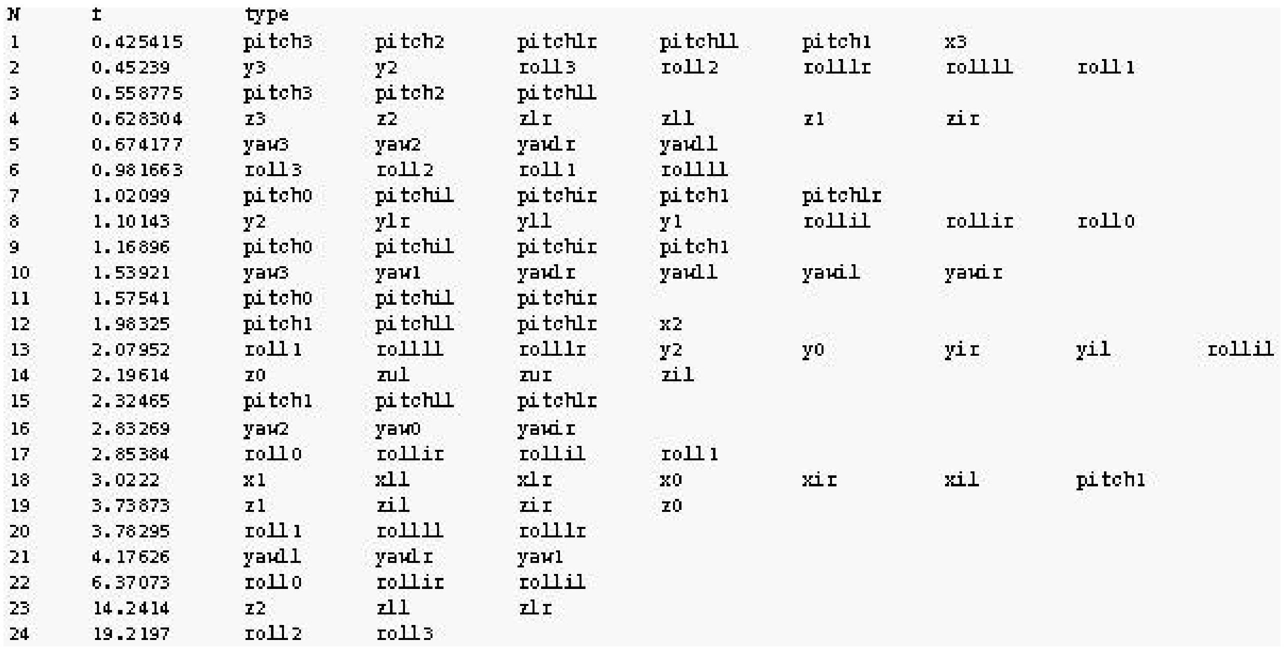
To test the provision for thermal noise in the Mathematica, the thermal noise was calculated for a quad system and compared to analytical results. Because of the complexity of the quad, it is not practical to compute the total thermal noise in closed form. Instead, a “naïve” prediction based on only the last stage was used. Because thermal noise generated in higher stages is filtered by the lower stages, the full thermal noise should asymptote to the naïve prediction at frequencies above the highest mode.

3.1.1 Parameters of the simulated pendulum

The quad system used was similar to the MIT quad prototype except with fused silica fibres instead of steel wires in the last stage. The thickness of the fibres was chosen to minimize thermoelastic damping according to Phil WIllem's prescription: strain = alpha/beta.The results were generated using v2.2 of the quad model with the following overrides:

```
overrides = {
      Y3->ifo[Suspension,Silica,Ey],
      r3->Sqrt[m3 g l3 betasilica/(4 N[Pi] Y3 alphasilica)],
      betasilica->0.00015,
      deltafibre->Ysilica*temperature
            *(alphasilica-betasilica*g*(m3)/(nw3*N[Pi]*r3^2*Ysilica))^2
            /(rhosilica*Csilica),
      taufibre->magicnumber*rhosilica*Csilica*4*(N[Pi]*r3^2)/N[Pi],
      damping[imag,fibretype] -> ((phisilica + phissilica/r3/2)&),
      damping[imag,fibreatype] -> (
            (
                  phisilica
                  + phissilica/r3
                  + deltafibre*(2*N[Pi]*#1*taufibre)
                  /(1+(2*N[Pi]*#1*taufibre)^2)
            )&
            )
        };
```

The results at the end of stage 1 were used. (It turns out that the stage 2 calculation is a very small perturbation.) The mode structure at the end of stage 1 was as follows:



N	f	type							
1	0.425415	pitch3	pitch2	pitchlr	pitchll	pitch1	x3		
2	0.45239	y3	y2	roll3	roll2	rolllr	rollll	roll1	
3	0.558775	pitch3	pitch2	pitchll					
4	0.628304	z3	z2	zlr	zll	z1	zir		
5	0.674177	yaw3	yaw2	yawlr	yawll				
6	0.981663	roll3	roll2	roll1	rollll				
7	1.02099	pitch0	pitchil	pitchir	pitch1	pitchlr			
8	1.10143	y2	ylr	yll	y1	rollil	rollir	roll0	
9	1.16896	pitch0	pitchil	pitchir	pitch1				
10	1.53921	yaw3	yaw1	yawlr	yawll	yawil	yawir		
11	1.57541	pitch0	pitchil	pitchir					
12	1.98325	pitch1	pitchll	pitchlr	x2				
13	2.07952	roll1	rollll	rolllr	y2	y0	yir	yil	rollil
14	2.19614	z0	zul	zur	zil				
15	2.32465	pitch1	pitchll	pitchlr					
16	2.83269	yaw2	yaw0	yawir					
17	2.85384	roll0	rollir	rollil	roll1				
18	3.0222	x1	xll	xlr	x0	xir	xil	pitch1	
19	3.73873	z1	zil	zir	z0				
20	3.78295	roll1	rollll	rolllr					
21	4.17626	yawll	yawlr	yaw1					
22	6.37073	roll0	rollir	rollil					
23	14.2414	z2	zll	zlr					
24	19.2197	roll2	roll3						

Table 3: Mode frequency and structure for stage one of the quad model with silica fibres in the last stage. The variable names below and to the right of "type" are the coordinates with significant displacement in the corresponding eigenmodes. Variables with postscript "3" refer to the optic so the l

3.1.2 Analytical calcululation for comparison

As noted above, "naive" theoretical results for comparison were calculated by taking each of the six DOFs of the final stage separately and applying the fluctuation dissipation theorem formula for a one-dimensional mass-spring system, with appropriate masses (or MOIs), elasticities and frequency-dependent damping factors. The sources of elasticity considered are listed below. The damping for each elasticity type is specified in terms of the tag (e.g., "fibreatype") used in the Mathematica. (The corresponding magnitudes and frequency dependences can be found in the code. Although the expressions used were intended to be representative, the details don't matter for the present purposes because they enter in the same way into the full-model results and the comparison results.)

Longitudinal:

- Pendulum elasticity of 4 wires of length `l3` supporting `m3/4` each:

  `4 m3/4 g/l3` (damping: lossless)

- Flexure elasticity (about the y-axis) of 8 wire endpoints converted to longitudinal elasticity with a lever arm of `l3`:

`8 Sqrt[m3/4 g Y3 M31]/2/l3^2` (damping: "`fibreatype`")

Transverse:

- Pendulum elasticity of 4 wires of length `l3` supporting `m3/4` each:

  `4 m3/4 g/l3` (damping: lossless)

- Flexure elasticity about the x-axis of 8 wire endpoints, converted to longitudinal elasticity with a lever arm of `l3`:

  `8 Sqrt[m3/4 g Y3 M31]/2/l3^2` (damping: "`fibreatype`")

Vertical:

- Direct elasticity of 4 wires of longitudinal elasticity `kw3`:

  `4 kw3` (damping: "`fibretype`")

Yaw:

- Pendulum elasticity of 2 pairs of wires supporting `m3/2` each, converted to a torque about the z axis through a lever arm of the radius of the optic:

  `m3/2 g/l3  tr^2` (damping: lossless)

- Flexure elasticity of 8 wire endpoints, converted to a longitudinal elasticity with a lever arm of `l3`, and then back to a torque (about the z axis) with a lever arm of the radius of the optic:

  `8 Sqrt[m3/4 g Y3 M31]/2/l3^2 n5^2` (damping: "fibreatype")

Pitch:

- Direct elasticity of 4 wires of longitudinal elasticity `kw3`, converted to a torque about the x axis through a lever arm of half the wire separation:

  `4 kw3 sl^2` (damping: "`fibretype`")

- Flexure elasticity of 4 wire endpoints:

  `4 Sqrt[m3/4 g Y3 M31]/2` (damping: "`fibreatype`")

Roll:

- Direct elasticity of 4 wires of longitudinal elasticity `kw3`, converted to a torque about the x axis through a lever arm of 1/2 the radius of the optic:

  `4*kw3*tr^2` (damping: "`fibretype`")

- Flexure elasticity is theoretically relevant as for pitch but small enough to ignore

3.2 Results

As can be seen in Figures 1-6, the thermal noise in the full model asymptotes very rapidly to the naive prediction at frequencies above the highest mode in the transfer function for each DOF.

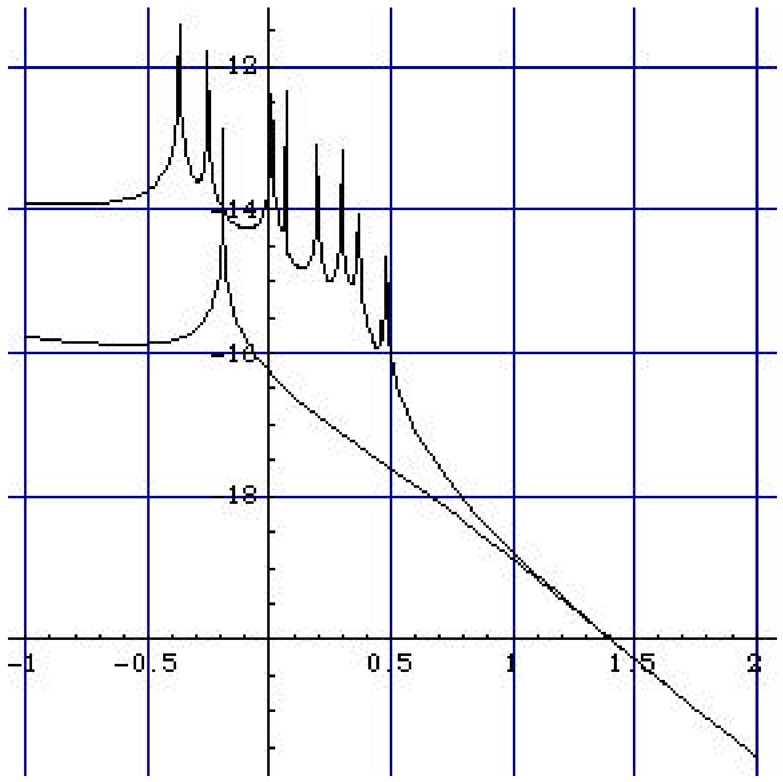


Figure 1: Optic longitudinal thermal noise (m/rtHz) and last-stage prediction against log f.

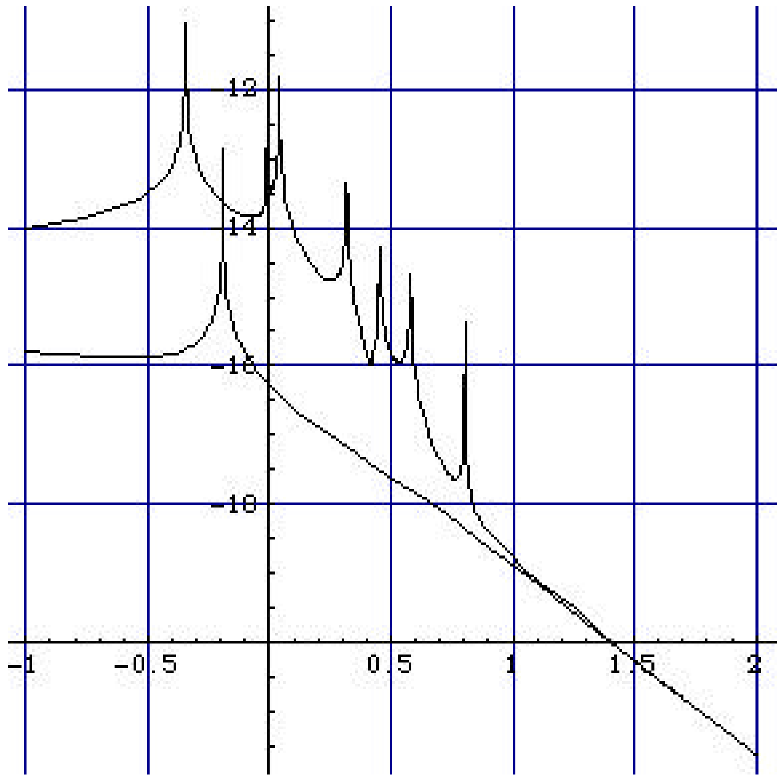


Figure 2: Optic transverse thermal noise (m/rtHz) and last-stage prediction against log f.

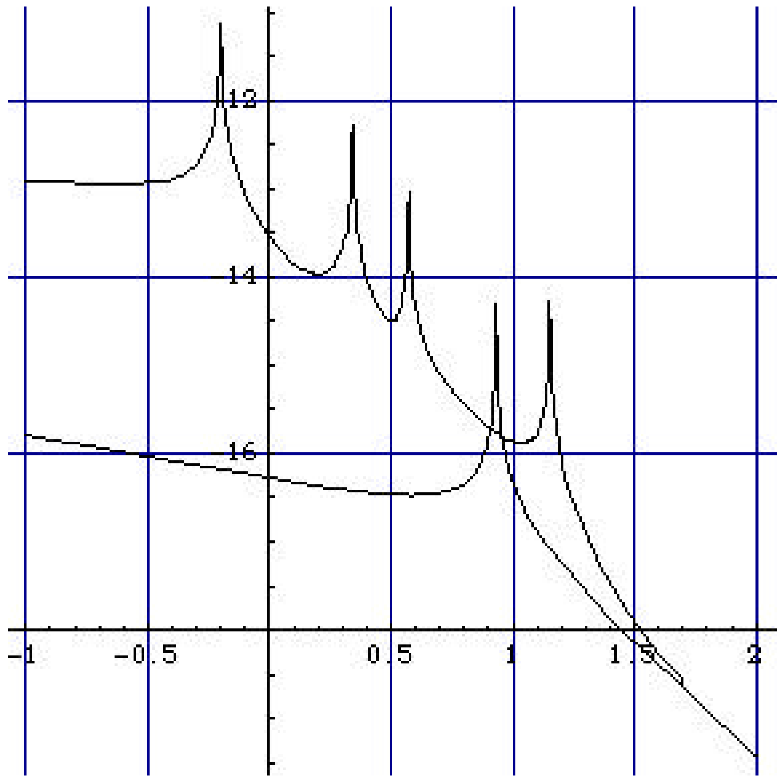


Figure 3: Optic vertical thermal noise (m/rtHz) and last-stage prediction against log f.

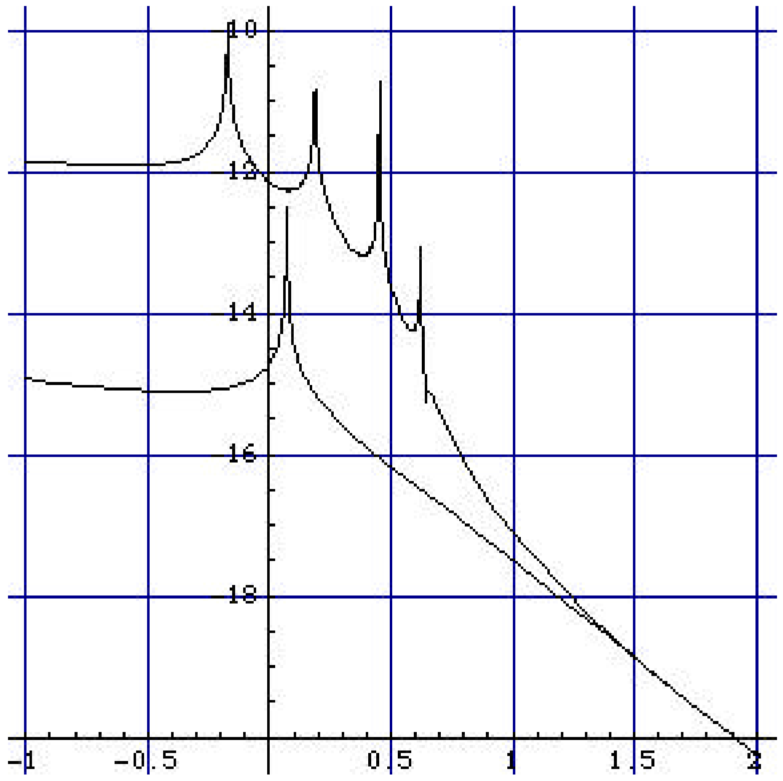


Figure 4: Optic yaw thermal noise (rad/rtHz) and last-stage prediction against log f.

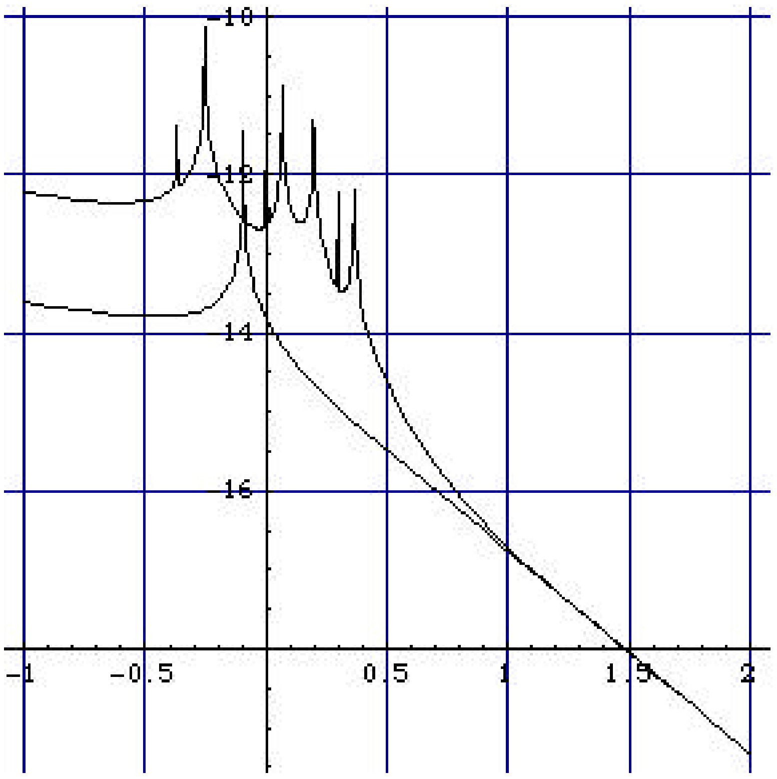


Figure 5: Optic pitch thermal noise (rad/rtHz) and last-stage prediction against log f.

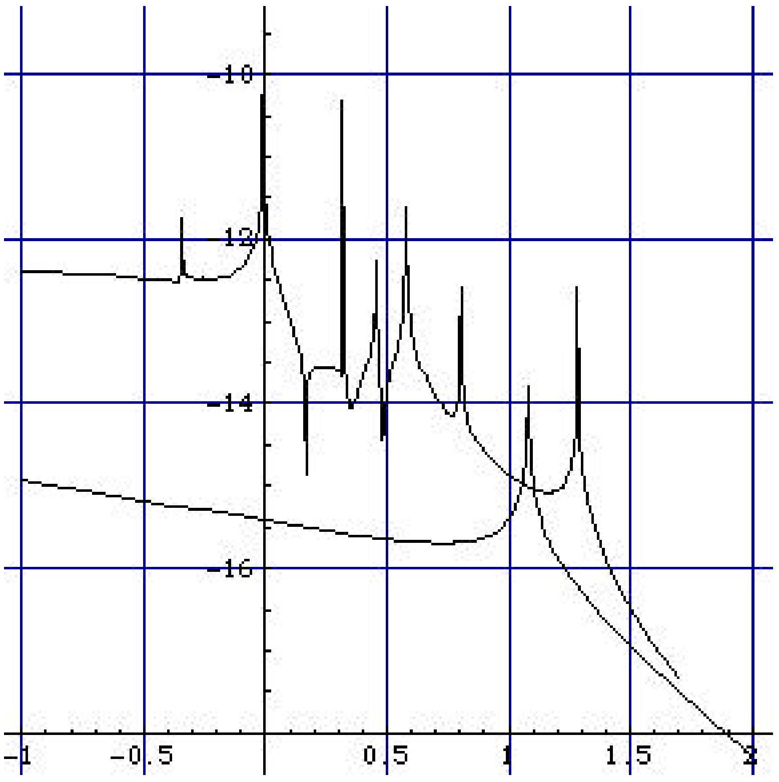


Figure 6: Optic roll thermal noise (m/rtHz) and last-stage prediction against log f.

4 Conclusion

As a result of this comparison a number of bugs in both models were discovered and corrected, and although one can never be sure, it is probable that both models now implement completely correctly the particular subset of the physics that they aim to cover. Both models will probably continue to be useful in the future – the Matlab is familiar to the GEO people and quick to recompute, and the Mathematica will be useful for studies of assymetries, thermal noise and more radical designs. The latest versions of both models, with special cases for designs of interest can be found at

< http://www.ligo.caltech.edu/~mbarton/SUSmodels/>