Suspension Dynamics Modeling for (TAMA and) LIGO and KAGRA

Mark Barton

July GW Seminar

7/24/15

Prehistory

- Started modeling the X-pendulum I developed for TAMA300 at ICRR 1993-7.
- Originally Mathematica v2.2 (now v10.1)!

MBP15 HD 〉 Users 〉 mbarton 〉 Work 〉 TAMA 〉 Calculations 〉 2DX Analysis 〉 1stPrototype

Name	Date Modified	Kind	Size	Access
2DXOrigOpt.m	Wednesday, June 5, 1996 at 12:55 PM	Objec…code	254.4 KB	-rwxrw…x (777)

Glasgow Matlab Models

- Calum Torrie modeled the GEO triple as part of his PhD (LIGO-P000040-v1).
- Lots of hand-coded Matlab – hard to check and hard to modify. Also lots of simplifications to the physics.
- Ken Strain extended Calum's code to model first aLIGO QUAD prototype.
- These were being used to design aLIGO suspensions.
- Needed something to compare these models against and allow for future improvements.

The Mathematica Toolkit, PendUtil.nb

- The X-pendulum model was developed into a general toolkit.
- Implemented as a Mathematica “package”, PendUtil.nb, for specifying different configurations (e.g., quad, triple etc) in a (relatively) user-friendly way
- Supported features:
  - 6-DOF rigid bodies for masses (no internal modes)
  - Springs described by an elasticity tensor and a vector of pre-load forces
  - Massless wires (i.e., no violin modes) but detailed elasticity model from beam equation
  - Arbitrary frequency-dependent damping on all sources of elasticity
  - Symbolic up to the point of minimizing the potential to find the equilibrium position
  - Calculates elasticity and mass matrices semi-numerically (symbolic partial derivatives of functions with mostly numeric coefficients)
  - Eigenfrequencies and eigenmodes calculated numerically
  - Arbitrary frequency dependent damping on each different elastic element
  - Transfer functions
  - Thermal noise plots
  - Export of state-space matrices to Matlab and E2E

Based on standard method of mass/stiffness matrices

- Express the potential energy of the system in terms of the coordinates:

$$E_P = E_P(x_1,\ldots x_n) = E_P(\mathbf{x})$$

- Express the kinetic energy of the system in terms of the coordinates and velocities:

$$E_K = E_K(x_1,\ldots x_n,\dot{x}_1,\ldots \dot{x}_n)$$

- Minimize the potential energy to find the equilibrium values of the coordinates.

$$\mathbf{x}_{eq} = \left(x_{1(eq)},\ldots x_{n(eq)}\right)^{\mathbf{T}}$$

- Compute the matrix of second derivatives of potential, a.k.a., the potential energy matrix or the stiffness matrix.

$$E_P = E_P(\mathbf{x}_{eq}) + \tfrac{1}{2}(\mathbf{x}-\mathbf{x}_{eq})^{\mathbf{T}}\,\mathbf{K}(\mathbf{x}-\mathbf{x}_{eq}) \qquad \mathbf{K}: K_{ij} = \left.\frac{\partial E_P}{\partial x_i\,\partial x_j}\right|_{\mathbf{x}=\mathbf{x}_{eq}}$$

- Compute the matrix of second derivatives of kinetic energy, a.k.a., the kinetic energy matrix or the mass matrix:

$$E_K = \tfrac{1}{2}\dot{\mathbf{x}}^{\mathbf{T}}\mathbf{M}\dot{\mathbf{x}} \qquad \mathbf{M}: M_{ij} = \left.\frac{\partial E_K}{\partial \dot{x}_i\,\partial \dot{x}_j}\right|_{\substack{\dot{\mathbf{x}}=0\\ \mathbf{x}=\mathbf{x}_{eq}}}$$

- Write matrix equations of motion and solve for particular numerical frequencies using

$$\dot{\mathbf{x}} = i\omega\mathbf{x} = 2\pi i f\mathbf{x} \qquad \ddot{\mathbf{x}} = -(2\pi f)^2\mathbf{x}$$

$$\mathbf{M}\ddot{\mathbf{x}} = -\mathbf{K}\mathbf{x} + \mathbf{C}\mathbf{x}_{external} + \mathbf{f}_{external}$$

$$\mathbf{x} = \left(\mathbf{K} - (2\pi f)^2\,\mathbf{M}\right)^{-1}\left(\mathbf{C}\mathbf{x}_{external} + \mathbf{f}_{external}\right)$$

- Or, assume a sinusoidal solution with no external forces

$$\mathbf{K}\mathbf{e}_i = \omega_i^{\,2}\mathbf{M}\mathbf{e}_i$$

- Do a simultaneous diagonalization to obtain the eigenfrequencies $f_i = \omega_i/2\pi$ and eigenmodes $\mathbf{e}_i$:

$$\mathbf{x}_i(t) = \mathbf{x}_{eq} + \mathbf{e}_i e^{\omega_i t}$$

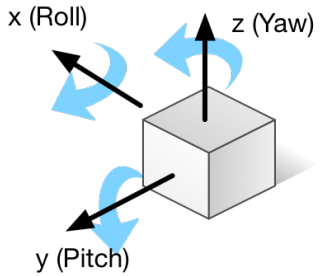
- See Goldstein, "Classical Mechanics", 3rd Ed.

Rigid-body mass coordinates

- Six coordinates for center of mass (COM): x, y, z, yaw (Y), pitch (P), roll (R)
- Three extra "body" coordinates for non-COM points such as wire attachments

$$\begin{pmatrix} x_{space} \\ y_{space} \\ z_{space} \end{pmatrix} = \begin{pmatrix} x_{COM} \\ y_{COM} \\ z_{COM} \end{pmatrix} + \begin{pmatrix} \cos Y & -\sin Y & 0 \\ \sin Y & \cos Y & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos P & 0 & -\sin P \\ 0 & 1 & 0 \\ \sin P & 0 & \cos P \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos R & -\sin R \\ 0 & \sin R & \cos R \end{pmatrix} \begin{pmatrix} x_{body} \\ y_{body} \\ z_{body} \end{pmatrix}$$

$$= \begin{pmatrix} x_{COM} \\ y_{COM} \\ z_{COM} \end{pmatrix} + \begin{pmatrix} \cos P \cos Y & \cos Y \sin P \sin R - \cos R \sin Y & \cos R \cos Y \sin P + \sin R \sin Y \\ \cos P \sin Y & \cos R \cos Y + \sin P \sin R \sin Y & \cos R \sin P \sin Y - \cos Y \sin R \\ -\sin P & \cos P \sin R & \cos P \cos R \end{pmatrix} \begin{pmatrix} x_{body} \\ y_{body} \\ z_{body} \end{pmatrix}$$



- The model definer needs to provide a list of the COM coordinates of all the masses.

Kinetic energy

- There is typically a linear term in terms of mass *m* plus angular term in terms of moment of inertia tensor ***I***:

$$E_K = \tfrac{1}{2} m(\dot{x}^2 + \dot{y}^2 + \dot{z}^2) + \tfrac{1}{2} \begin{pmatrix} \omega_x & \omega_y & \omega_z \end{pmatrix} \begin{pmatrix} I_{xx} & I_{xy} & I_{zx} \\ I_{xy} & I_{yy} & I_{yz} \\ I_{xz} & I_{yz} & I_{zz} \end{pmatrix} \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix}$$

- ***I*** is in body coordinates so the angular velocity vector needs to be transformed:

$$\begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} = \begin{pmatrix} -\sin P & 0 & 1 \\ \cos P \sin R & \cos R & 0 \\ \cos P \cos R & -\sin R & 0 \end{pmatrix} \begin{pmatrix} \dot{Y} \\ \dot{P} \\ \dot{R} \end{pmatrix}$$

- The model definer needs to supply a Mathematica expression for the kinetic energy for all masses, with terms similar to the above.

Gravitational energy

$$E_G = mgz$$

- The model definer has to supply a Mathematica expression giving the total gravitational potential for all masses.

Wire energy

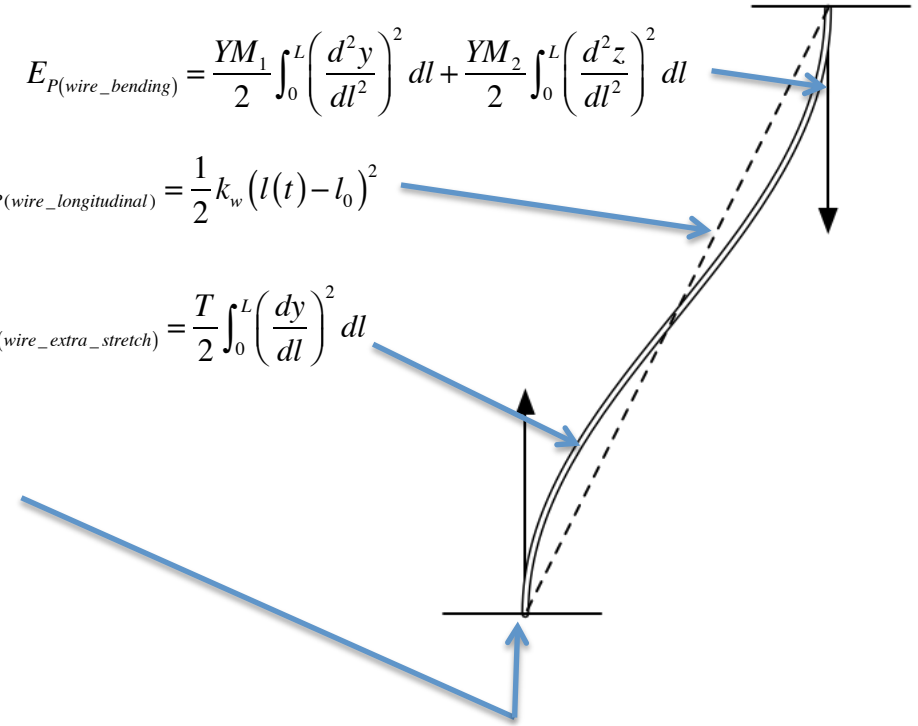
- Wire energy is broken into four terms:
  - Longitudinal stretch based on straight-line distance
  - Bending near the endpoints
  - Extra longitudinal stretch due to bending
  - Torsion

$$E_{P(wire\_bending)} = \frac{YM_1}{2}\int_0^L \left(\frac{d^2y}{dl^2}\right)^2 dl + \frac{YM_2}{2}\int_0^L \left(\frac{d^2z}{dl^2}\right)^2 dl$$

$$E_{P(wire\_longitudinal)} = \frac{1}{2}k_w\left(l(t) - l_0\right)^2$$

$$E_{P(wire\_extra\_stretch)} = \frac{T}{2}\int_0^L \left(\frac{dy}{dl}\right)^2 dl$$

$$E_{P(wire\_torsional)} = \frac{1}{2}\left(\frac{TJ}{A} + GJ_e\right)\Delta\theta_T^2$$

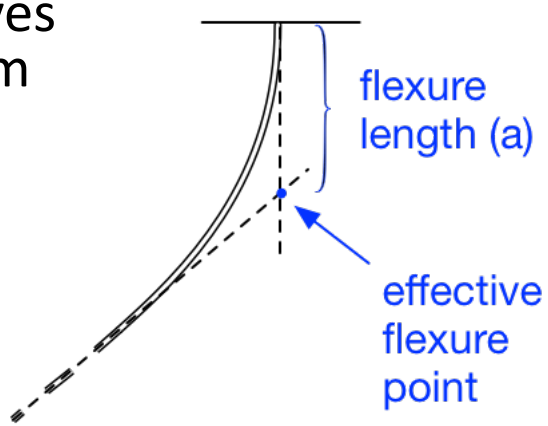


- Model definer needs to supply a list of parameters for each wire, including Young's modulus, length, moment of area, attachment points, etc, etc.

Flexure correction

- Wire bending terms use some complicated 3D geometry and are slow to calculate.
- It turns out there's a shortcut: the wire behaves as if attached with a hinge at a distance *a* from actual break-off point, where

$$a = \sqrt{\frac{EI}{T}}$$



- E=Young's modulus, I = second moment of area, T = tension
- This trick was not used in the Mathematica toolkit but has been used in the exported Matlab code.

Springs

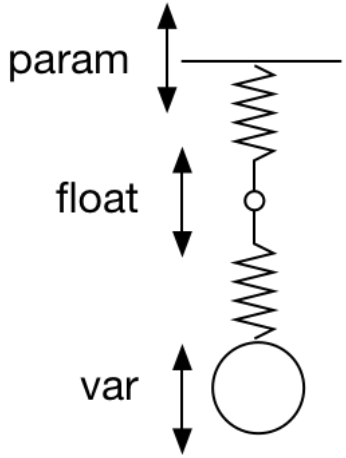
- Simple zero-length spring model connecting a point on one mass to a point on another.
- 6x6 matrix of elastic constants plus a 6x1 vector of pre-load forces:

$$E_{P(spring)} = \frac{1}{2}\begin{pmatrix} \Delta x & \Delta y & \Delta z & \Delta Y & \Delta P & \Delta R \end{pmatrix}\begin{pmatrix} K_{xx} & K_{xy} & K_{xz} & K_{xY} & K_{xP} & K_{xR} \\ K_{xy} & K_{yy} & K_{yz} & K_{yY} & K_{yP} & K_{yR} \\ K_{xz} & K_{yz} & K_{zz} & K_{zY} & K_{zP} & K_{zR} \\ K_{xY} & K_{yY} & K_{zY} & K_{YY} & K_{YP} & K_{YR} \\ K_{xP} & K_{yP} & K_{zP} & K_{YP} & K_{PP} & K_{PR} \\ K_{xR} & K_{yR} & K_{zR} & K_{YR} & K_{PR} & K_{RR} \end{pmatrix}\begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \\ \Delta Y \\ \Delta P \\ \Delta R \end{pmatrix} + \begin{pmatrix} f_x & f_y & f_z & f_Y & f_P & f_R \end{pmatrix}\begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \\ \Delta Y \\ \Delta P \\ \Delta R \end{pmatrix}$$

- Model definer has to provide a list of springs with masses, attachment points and elastic constant and pre-load information.

Extra coordinates

- As well as the coordinates used in the normal mode analysis ("**vars**"), two other sorts are important:
  - "**params**" – the support and other objects which are stationary during normal modes but move for transfer functions
  - "**floats**" – things like wire-spring junctions with no associated mass



Stiffness matrix for extra coordinates

- It is convenient to calculate a master stiffness matrix with partial derivatives between all types of coordinates:

$$\mathbf{K}_{master} = \begin{array}{c} \begin{matrix} \text{"vars"} & \text{"floats"} & \text{"params"} \end{matrix} \\ \begin{pmatrix} \mathbf{K} & \mathbf{C}_{XQ} & \mathbf{C}_{XS} \\ \mathbf{C}_{QX} & \mathbf{Q} & \mathbf{C}_{QS} \\ \mathbf{C}_{SX} & \mathbf{C}_{SQ} & \mathbf{S} \end{pmatrix} \end{array} \begin{matrix} \\ \text{"vars"} \\ \text{"floats"} \\ \text{"params"} \end{matrix}$$

- K, Q and S give the coupling among vars, floats and params within their respective groups.
- C_{XQ}, C_{QS} and C_{SX} give the coupling between groups.
- Because the "floats" are dependent on the others, they need to be eliminated:

$$\mathbf{K}_{effective} = \mathbf{K} - \mathbf{C}_{XQ}\mathbf{Q}^{-1}\mathbf{C}_{QX} \qquad \mathbf{C}_{SX(effective)} = \mathbf{C}_{SX} - \mathbf{C}_{SQ}\mathbf{Q}^{-1}\mathbf{C}_{QX} \qquad \mathbf{S}_{effective} = \mathbf{S} - \mathbf{C}_{SQ}\mathbf{Q}^{-1}\mathbf{C}_{QS}$$

Damping

- Lossiness in elastic components can be represented by a complex elastic constant:

$$k \to k_0\left(\varepsilon'(\omega) + i\varepsilon''(\omega)\right)$$

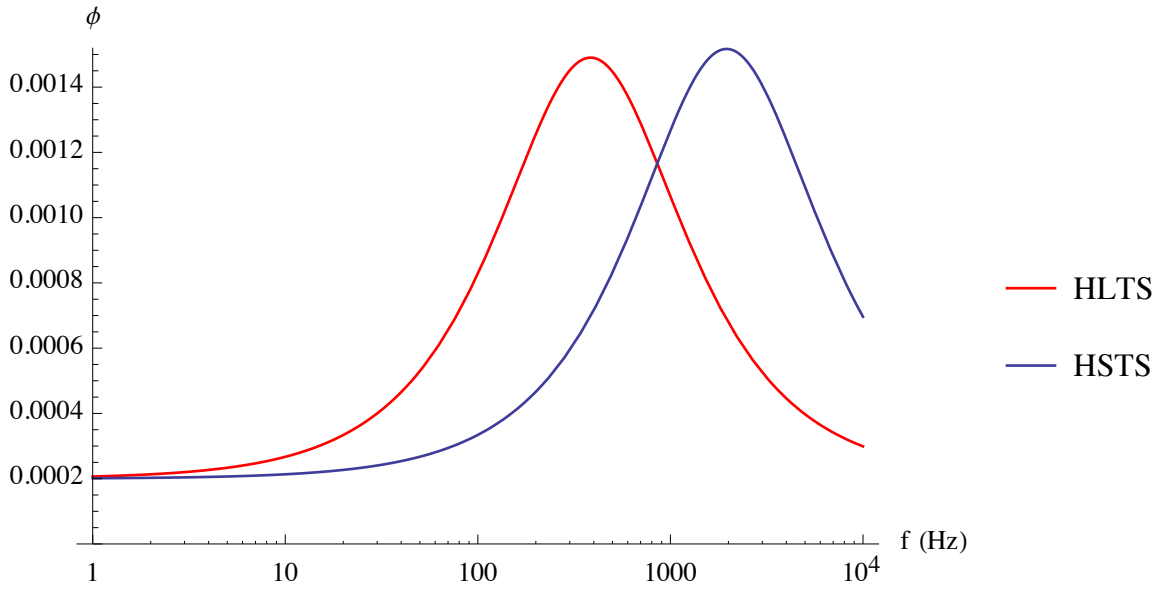
- The real and imaginary parts should satisfy the Krämers Krönig relationship:

$$\varepsilon'(\omega) - 1 = \frac{2}{\pi} PV \int_{-\infty}^{\infty} \frac{\varepsilon''(x)}{x - \omega} dx \qquad \varepsilon''(\omega) = -\frac{2}{\pi} PV \int_{-\infty}^{\infty} \frac{\varepsilon'(x) - 1}{x - \omega} dx$$

- If losses are small, the real part can be assumed to be constant: $k \to k_0\left(1 + i\phi(f)\right)$
- The model definer can specify a different damping function for each elastic component.
- The toolkit keeps track of which damping function applies to which coefficients in the master stiffness matrix.

Wire/Fibre damping

- Wire ϕ usually has a frequency-independent "structural" term and a frequency dependent "thermoelastic " term.



- Thermoelastic damping has a characteristic peak at the timescale of heat flow across the wire.

Thermal Noise

- Suspension thermal noise is a potential limiting factor in GW detectors.
- Noise is given in terms of damping by Fluctuation Dissipation Theorem:

$$x^2(\omega) = \frac{4k_B T\,\mathrm{Re}\left(Y(\omega)\right)}{\omega^2}$$

- Y is the admittance (v/F) at a test point where the thermal noise is to be calculated.

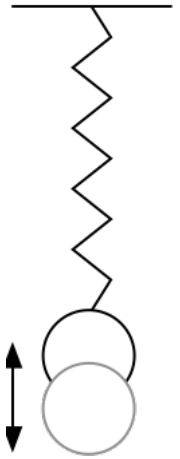
Dissipation Dilution (i)

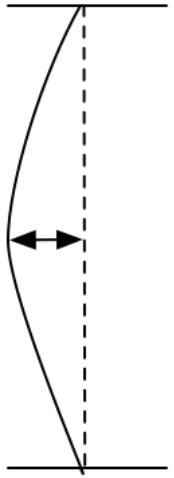
- In a simple mass-spring system, the quality factor of the oscillation depends purely on the spring material: $Q = \frac{1}{\phi_{material}}$

- However systems such as pendulums and wires are used in GW detectors because with certain geometries, the damping (dissipation of energy) is "diluted": $Q = \frac{D}{\phi_{material}} \gg \frac{1}{\phi_{material}}$

Dissipation Dilution (ii)

- The dissipation dilution factor *D* depends on the fraction of the energy involved in first-order stress changes of the material.
- Pulling a pendulum or violin string sideways creates only a second-order or smaller stretch -> dilution.
- Quick test: calculate the contribution to the potential matrix for each potential term twice, once with and without all tensions zeroed.





Calculation Procedure (i)

- The model calculation notebook is run.
- It loads the model definition notebook, which loads the toolkit, the model definition, and a default set of numerical values.
- The calculation notebook can then selectively override some of the numerical values.
- The wire and spring lists are processed to create a list of potential terms, each with a damping function.
- The total potential is computed, with numerical values for all quantities except the "var" coordinates. (Usually the wire bending potential terms are omitted for speed.)
- The total potential is minimized to find the equilibrium position.

Calculation Procedure (ii)

- Numerical values for everything but the coordinates are substituted into the potential terms.
- The potential terms are differentiated to find the stiffness matrix elements. Each term is processed separately for two reasons:
  - for speed (each term depends on only a few coordinates, so most derivatives are zero and don't need to be computed)
  - to keep contributions with different damping functions separate.
- The process is repeated with the tension switched off, to allow dissipation dilution to be calculated.

Calculation Procedure (iii)

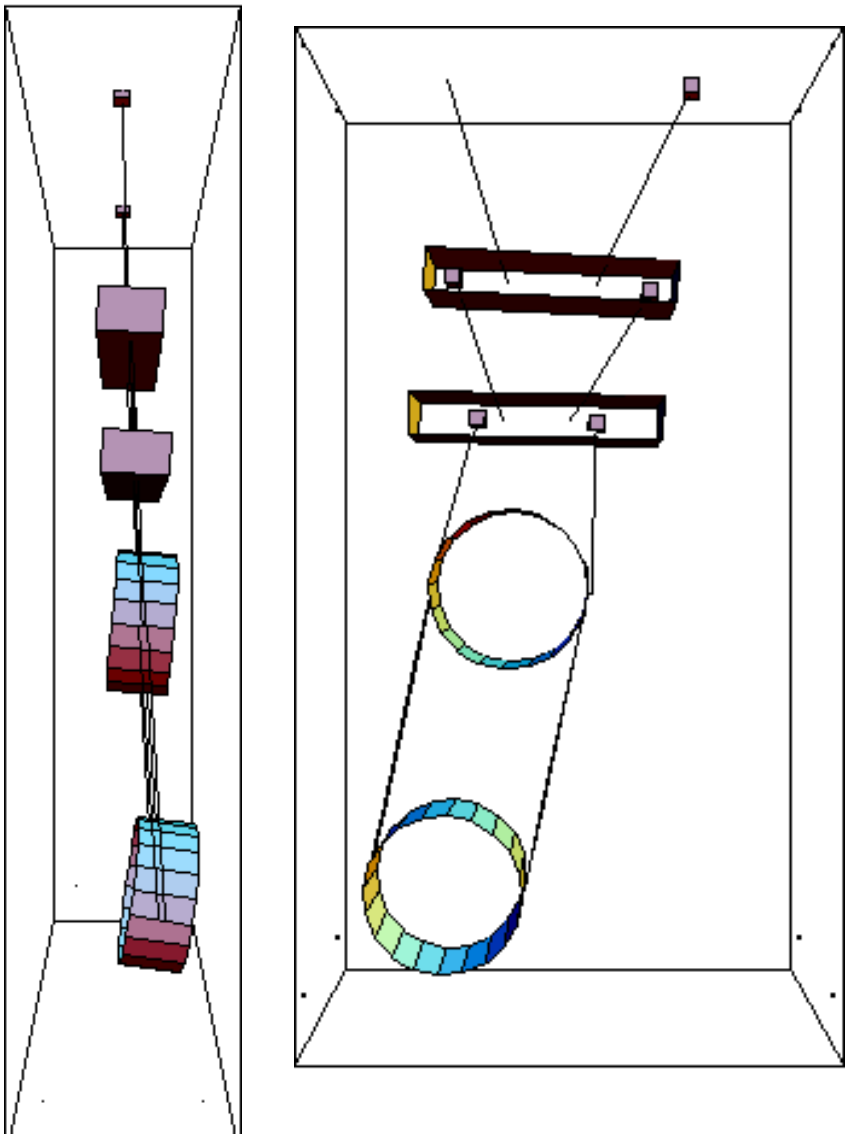
- Versions of the stiffness matrix with and without damping are calculated.
- The stiffness matrix without damping (a totally numerical matrix) is used to calculate the eigenfrequencies and eigenmodes.
- The stiffness matrix with damping (a mostly numerical matrix that is a Mathematica function of frequency, f) is used for all other purposes.
- Mathematica functions are provided to allow transfer functions from/to selected inputs/outputs, thermal noise plots at selected test points, eigenmode shape plots etc.
- If the small but time-consuming wire bending/torsion potential terms were omitted at the beginning, they are computed and added in.

Models for LIGO

- Models defined for all the LIGO suspensions (no KAGRA yet):
  - QUAD: single chain of AdvLIGO quad pendulum, with 4 masses, 6 blade springs and 14 wires.
  - BSFM, HSTS, HLTS: 3 masses, 6 blade springs and 10 wires.
  - OFMC, TMTS: 2 masses, 2 or 4 blades, 6 wires
  - HAUX, HTTS, OFIS: 1 mass, assorted blade/wire combinations
  - Many toy models

Example model: LIGO quad pendulum

- 2 blade springs
- 2 wires
- top mass
- 2 blade springs
- 4 wires
- upper intermediate mass
- 2 blade springs
- 4 wires
- intermediate mass
- 4 fibres (or wires)
- optic (or reaction mass)



Sample Output (i)
Table of Mode Freqs/Shapes

Model "20140304TMproductionTM"

The aLIGO quad suspension main chain, with monolithic final stage (i.e., fused silica fibres supporting the test mass)

N	f	type		
1	0.432096	pitch3	pitch2	pitch1
2	0.461778	y3	roll3	roll2
3	0.522439	pitch3	pitch2	
4	0.552861	z3	z2	
5	0.599185	yaw3	yaw2	
6	0.86847	roll1	roll3	roll2
7	0.99234	pitch0	pitch1	x2
8	1.04328	y2	y1	y3
9	1.33906	pitch0	pitch1	
10	1.34913	yaw3	yaw1	
11	1.59901	pitch0	pitch2	
12	1.98325	pitch0	x0	x1
13	2.10062	roll1	y0	y1
14	2.22968	z0	z1	
15	2.39081	yaw0	yaw2	
16	2.64322	pitch1	roll1	roll0
17	2.74352	pitch1	pitch0	
18	3.0388	yaw1	yaw0	
19	3.30053	pitch0	roll0	roll1
20	3.39905	x0	x1	pitch0
21	3.56066	z1	z0	
22	5.08788	roll0	pitch0	
23	9.68987	z2	z3	
24	13.8069	roll2	roll3	

Sample Output (ii)
Individual Mode Shape

#1

In[34]:=
```
Hz2[[-1]]
```

Out[34]=
```
0.432096
```

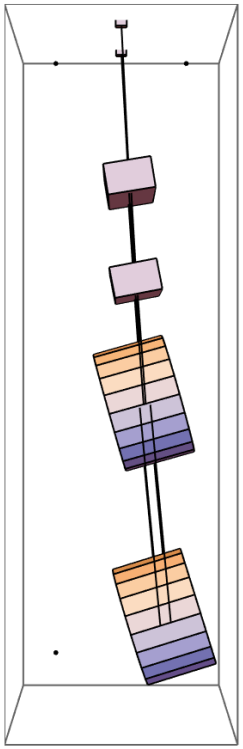
In[35]:=
```
pretty[Chop[e2ni.eigenvectors2[[-1]], 10^-4]]
```

Out[35]//TableForm=

	x	y	z	yaw	pitch	roll
Mass N	0.0446664	0	0	0	-0.324194	0.00109208
Mass U	0.0791596	0.000232412	0	0	-0.380551	0.00221296
Mass 2	0.12786	0.000408359	0	0	-0.54992	0.00186767
optic	0.244087	0.000781052	0	0	-0.602881	0.00187027

In[111]:=
```
DoWithStatus["Plotting stage 2 mode 1",
 eigenplot[eigenvectors2[[-1]], 0.5, {0, -3, 0}, floatmatrix2]]
```

Out[111]=

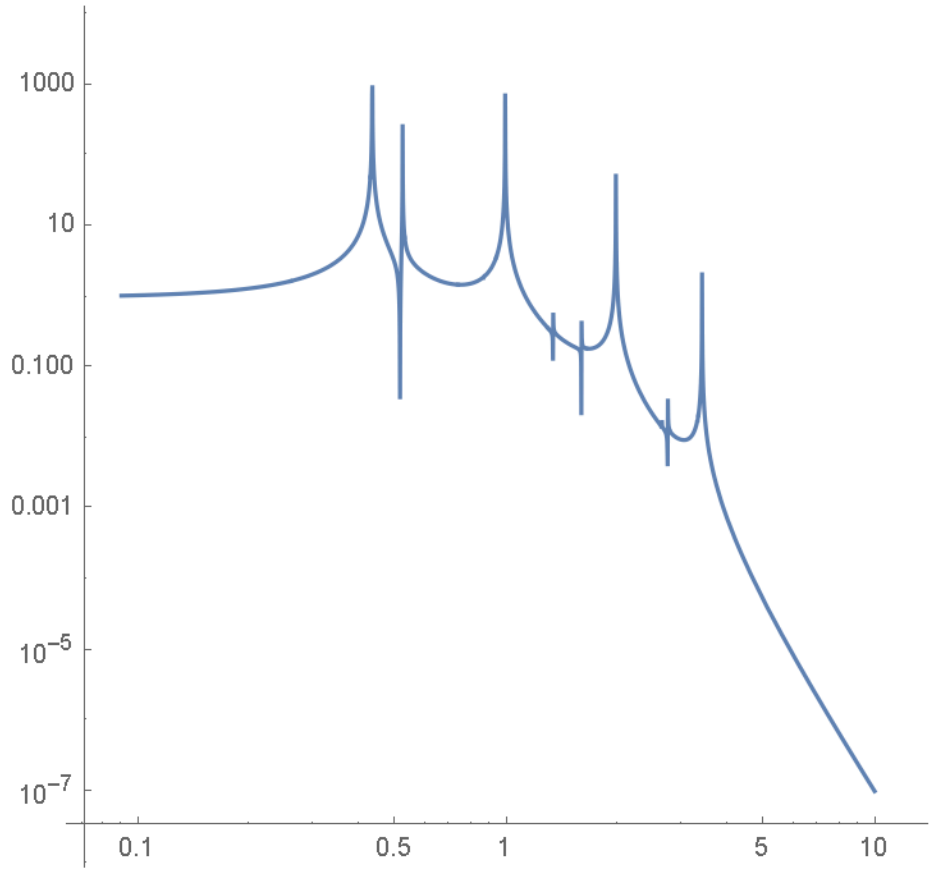


Sample Output (iii)
Transfer Function

Transfer function from x motion of the support to x motion of the optic

In[106]:=
```
DoWithStatus["Plotting stage 2 x transfer function",
    plotTF[eom2,coupling2,supportxinput,opticxoutput,0.09,10]
]
```

Out[106]=

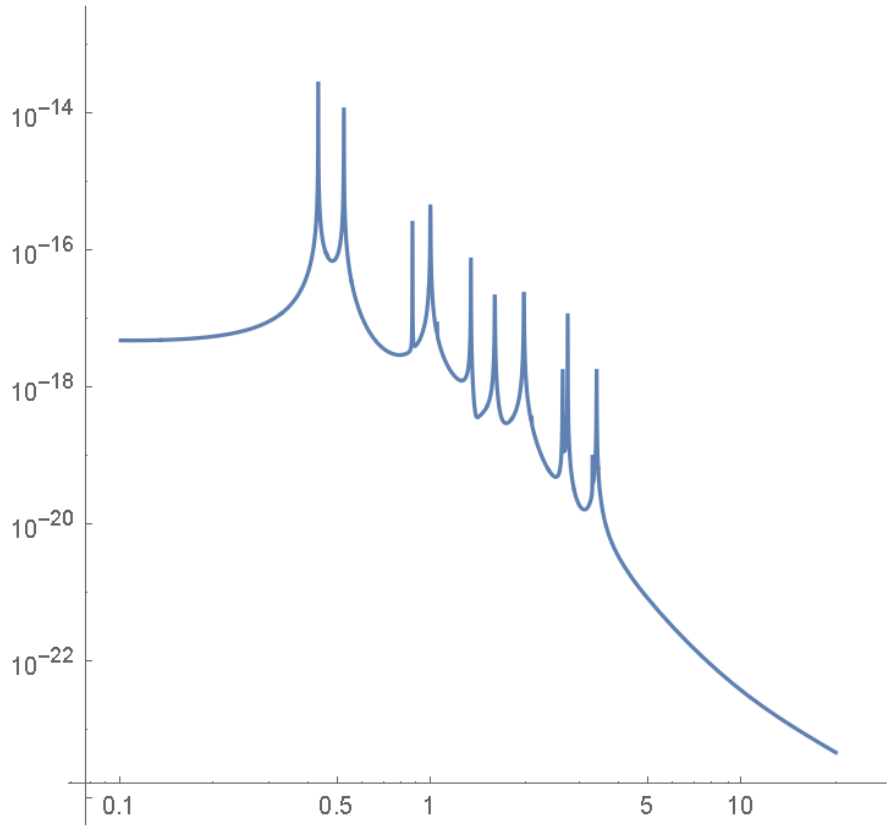


Sample Output (iv) Thermal Noise

Thermal noise in x motion of the optic, scaled to give strain noise for four optics

In[107]:=
```
DoWithStatus["Plotting stage 2 x thermal noise",
    plotTN[eom2,opticxoutput,0.1,20,Sqrt[4]/ifo[Infrastructure,Length]]
]
```

Out[107]=



State Space Terminology

- For simulations, it is convenient to write the equations of motion in state-space format. In traditional notation:

$$\begin{pmatrix} \dot{\mathbf{x}} \\ \mathbf{y} \end{pmatrix} = \begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ \mathbf{u} \end{pmatrix}$$

  - x: vector of state coordinates
  - u: vector of inputs
  - y: vector of outputs
  - A: state matrix (maps state to rate of change)
  - B: input matrix (maps u to rate of change)
  - C: output matrix (maps state to output)
  - D: feedthrough matrix (maps inputs directly to output)

SS State/Inputs/Outputs for Suspension Models

- In mechanical simulations, the EOMs are second-order, so the full state needs to be the coordinates ("vars") plus the velocities:

$$\mathbf{x} \rightarrow \begin{pmatrix} \mathbf{x} \\ \dot{\mathbf{x}} \end{pmatrix} \qquad \begin{pmatrix} \dot{\mathbf{x}} \\ \ddot{\mathbf{x}} \end{pmatrix} = \mathbf{A} \begin{pmatrix} \mathbf{x} \\ \dot{\mathbf{x}} \end{pmatrix} + \mathbf{B}\mathbf{u}$$

- A good set of inputs is the support coordinates ("params") plus input forces on the main coordinates:

$$\mathbf{u} = \begin{pmatrix} \mathbf{s} \\ \dot{\mathbf{s}} \\ \mathbf{f}_x \end{pmatrix}$$

- A good set of outputs is all the main coordinates plus the reaction forces on the support:

$$\mathbf{y} = \begin{pmatrix} \mathbf{x} \\ \mathbf{f}_s \end{pmatrix}$$

Full State Space for Suspension Models

$$\begin{pmatrix} \dot{\mathbf{x}} \\ \ddot{\mathbf{x}} \end{pmatrix}^{\langle 2N_x \times 1 \rangle} = \begin{pmatrix} \mathbf{0}^{\langle N_x \times N_x \rangle} & \mathbf{I}^{\langle N_x \times N_x \rangle} \\ -\mathbf{M}^{-1}\mathbf{K}^{\langle N_x \times N_x \rangle} & -\mathbf{M}^{-1}\mathbf{E_A}^{\langle N_x \times N_x \rangle} \end{pmatrix}^{\langle 2N_x \times 2N_x \rangle} \begin{pmatrix} \mathbf{x} \\ \dot{\mathbf{x}} \end{pmatrix}^{\langle 2N_x \times 1 \rangle}$$

$$+ \begin{pmatrix} \mathbf{0}^{\langle N_x \times N_s \rangle} & \mathbf{0}^{\langle N_x \times N_s \rangle} & \mathbf{0}^{\langle N_x \times N_x \rangle} \\ -\mathbf{M}^{-1}\mathbf{C}^{\langle N_x \times N_s \rangle} & -\mathbf{M}^{-1}\mathbf{E_B}^{\langle N_x \times N_s \rangle} & \mathbf{M}^{-1\langle N_x \times N_x \rangle} \end{pmatrix} \begin{pmatrix} \mathbf{s} \\ \dot{\mathbf{s}} \\ \mathbf{f_x} \end{pmatrix}^{\langle (2N_s+N_x) \times 1 \rangle}$$

$$\begin{pmatrix} \mathbf{x} \\ \mathbf{f_s} \end{pmatrix}^{\langle (N_x+N_s) \times 1 \rangle} = \begin{pmatrix} \mathbf{I}^{\langle N_x \times N_x \rangle} & \mathbf{0}^{\langle N_x \times N_x \rangle} \\ -\mathbf{C}^{\langle N_s \times N_x \rangle} & \mathbf{0}^{\langle N_s \times N_x \rangle} \end{pmatrix}^{\langle (N_x+N_s) \times 2N_x \rangle} \begin{pmatrix} \mathbf{x} \\ \dot{\mathbf{x}} \end{pmatrix}^{\langle 2N_x \times 1 \rangle}$$

$$+ \begin{pmatrix} \mathbf{0}^{\langle N_x \times N_s \rangle} & \mathbf{0}^{\langle N_x \times N_s \rangle} & \mathbf{0}^{\langle N_x \times N_x \rangle} \\ -\mathbf{S}^{\langle N_s \times N_s \rangle} & -\mathbf{E_D}^{\langle N_x \times N_s \rangle} & \mathbf{0}^{\langle N_s \times N_x \rangle} \end{pmatrix}^{\langle (N_x+N_s) \times (N_s+N_x) \rangle} \begin{pmatrix} \mathbf{s} \\ \dot{\mathbf{s}} \\ \mathbf{f_x} \end{pmatrix}^{\langle (2N_s+N_x) \times 1 \rangle}$$

- K is the stiffness matrix; M is the mass matrix
- C is the coupling stiffness matrix; S is the support stiffness
- The E matrices are arrays of velocity damping coefficients.
- Sizes of matrices and vectors are given as <Nrows x Ncolumns>.

Matlab Export

- All the components of the SS A/B/C/D matrices can easily be exported from Mathematica as numbers or Matlab code.
- For symbolic export, it's better to export M and calculate M^{-1} in Matlab.

Usage at LIGO

- Mathematica models were used to study thermal noise and asymmetrical suspensions.
- The GEO Matlab models were retained but had the hand-written state-space matrices replaced with improved code generated by the corresponding Mathematica model.
- Matlab models were used for analysis of experimental data and for control design.
- All Mathematica models are hosted on the SUS SVN (Subversion version control system) at https://redoubt.ligo-wa.caltech.edu/websvn/listing.php?repname=sus&path=%2Ftrunk%2FCommon%2FMathematicaModels%2F#path_trunk_Common_MathematicaModels_ (needs LIGO.org credentials).
- Matlab models are in the same SVN but scattered about.

References

- LIGO-T020205 - Models of the Advanced LIGO Suspensions in Mathematica™
- LIGO-T080188 - Models of the Advanced LIGO Suspensions in MATLAB
- https://awiki.ligo-wa.caltech.edu/aLIGO/Suspensions/OpsManual (needs LIGO credentials or see LIGO-E1200633).
- LIGO-T070101 - Dissipation dilution
- LIGO-T080096 - Flexure corrections

KAGRA

- Sekiguchi-san has used some of the ideas and written his own package.
- Much more automated and easy to use.
- Probably no need (certainly no plan) to redo KAGRA models with LIGO toolkit.
- Not too hard to do if there is a need for say detailed thermal noise plots.