Pendulum Modeling in *Mathematica*™ and *Matlab*™

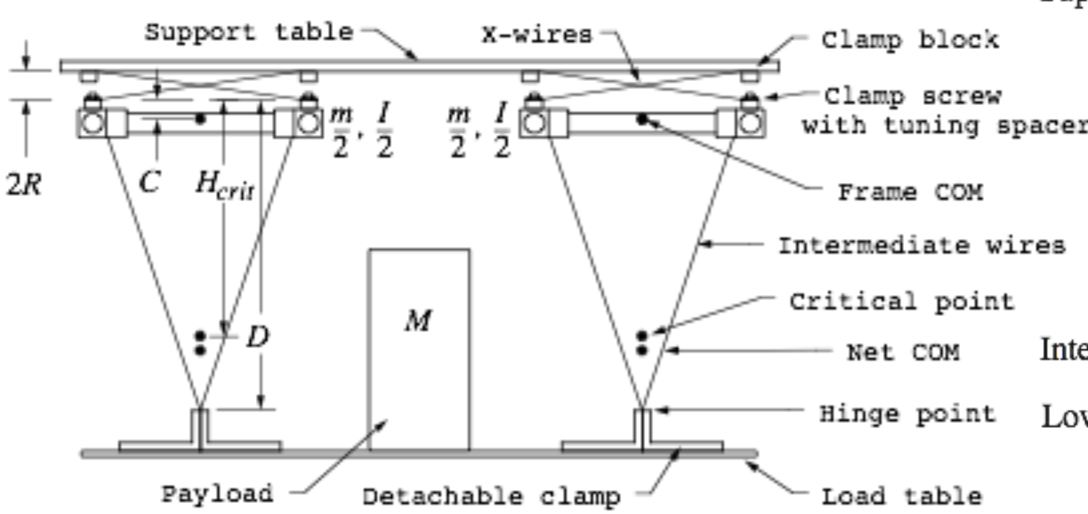
IGR Thermal Noise Group Meeting

4 May 2007

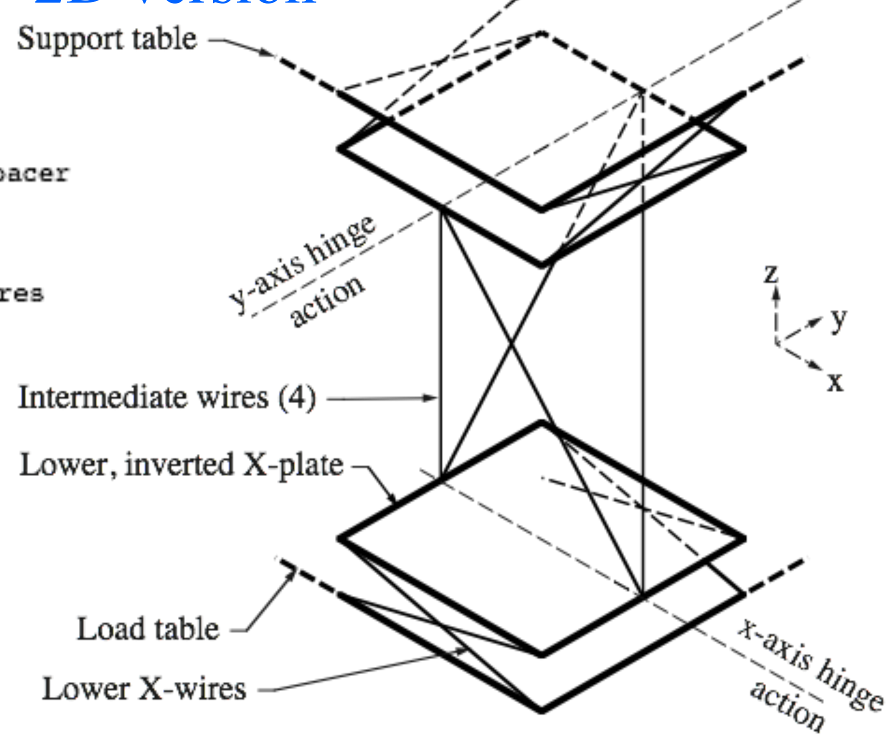
The X-Pendulum

- Developed as a low frequency vibration isolator for TAMA

1D version



2D version



Pendulum Modelling

- Wanted an AdvLIGO SUS design model to go beyond the Matlab model of Torrie, Strain et al.
- Desired features:
  - » Full 3D with provision for asymmetries
  - » Proper blade model
  - » Wire bending elasticity
  - » Arbitrary damping and consequent thermal noise
  - » Export to other environments such as Matlab/Simulink and E2E.
- Mathematica code originally developed for modeling the X-pendulum was available -> reuse and extend.
- See http://www.ligo.caltech.edu/~e2e/SUSmodels
- Manual: T020205-00 (-01 pending)

The Toolkit

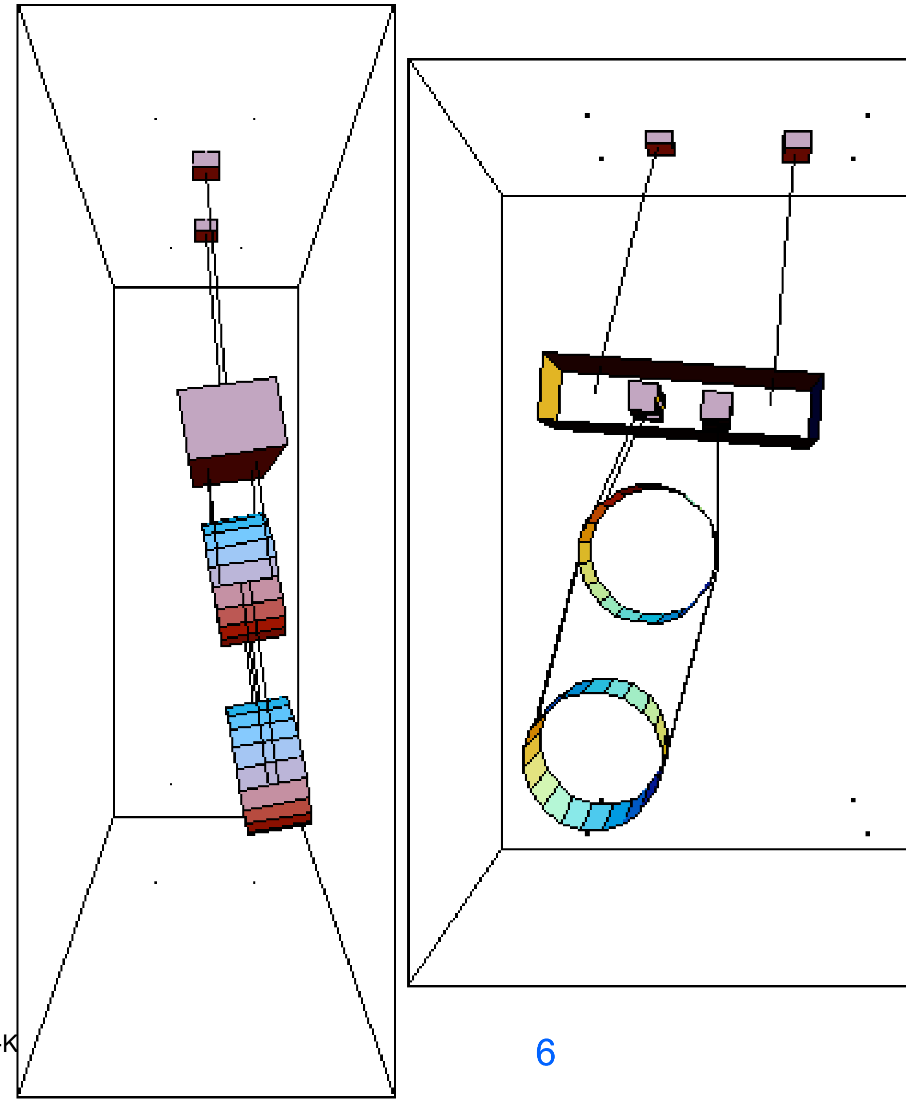
- The toolkit is a Mathematica “package”, PendUtil.nb, for specifying different configurations (e.g., quad, triple etc) in a (relatively) user-friendly way
- Supported features:
  - » 6-DOF rigid bodies for masses (no internal modes)
  - » Springs described by an elasticity tensor and a vector of pre-load 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

Models

- Two major families of models have been defined:
  - » The triple models reflect a generic GEO-style pendulum with 3 masses, 6 blade springs and 10 wires.
  - » The quad models reflect a standard AdvLIGO quad pendulum, with 4 masses, 6 blade springs and 14 wires.
- Many toy models
  - » LIGO-I two-wire pendulum
  - » Simple pendulum
  - » Simple pendulum on a blade spring
  - » Etc
- Steep learning curve but a major new model can be programmed in a day by an experienced user

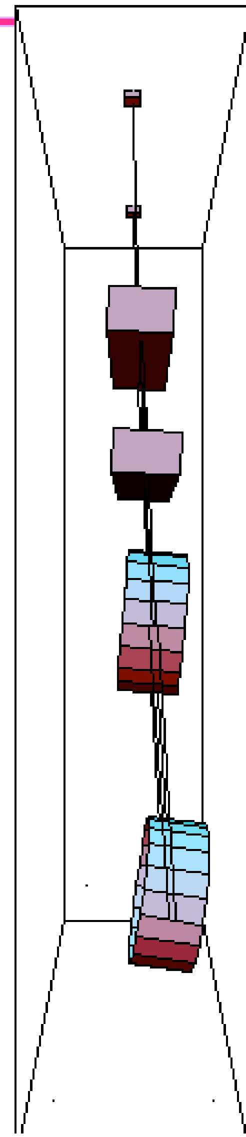
Triple Pendulum Model

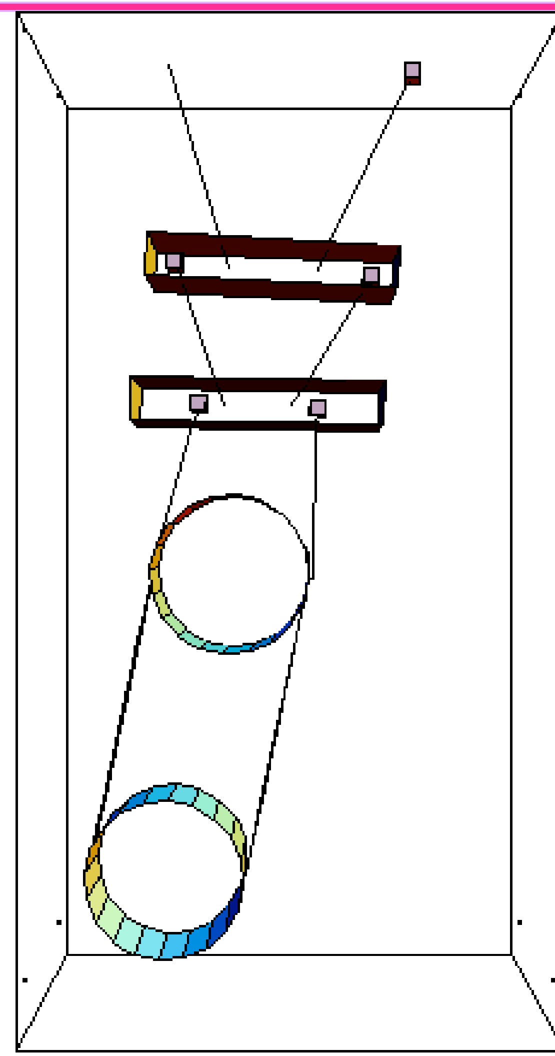
- 2 blade springs
- 2 wires
- “upper” mass
- 4 blade springs
- 4 wires
- “intermediate” mass
- 4 fibres
- optic



Quad Pendulum

- 2 blade springs
- 2 wires
- “top” mass
- 2 blade springs
- 4 wires
- “upper” mass
- 2 blade springs
- 4 wires
- “intermediate” mass
- 4 fibres
- optic





Defining a Model (i)

- Define the "variables" (cf. **x** in the theory - example from the xtra-lite triple):
- `allvars = {`
  - » `x1,y1,z1,yaw1,pitch1,roll1,`
  - » `x2,y2,z2,yaw2,pitch2,roll2,`
  - » `x3,y3,z3,yaw3,pitch3,roll3`
- `};`
- Define the "floats" (cf. **q** in the theory):
  - » `allfloats = {`
  - – `qul,qur,qlf,qlb,qrf,qrb`
  - » `};`
- Define the "parameters" (cf. **s** in the theory):
- `allparams = {`
  - » `x00, y00, z00, yaw00, pitch00, roll00`
- `};`

Defining a Model (ii)

- Define coordinate lists for rigid bodies of interest:
  - `optic = {x3, y3, z3, yaw3, pitch3, roll3};`
  - `support = {x00, y00, z00, yaw00, pitch00, roll00};`

- Define coordinate lists for points on rigid bodies
  - `massUl={0,-n1,d0}; (* left wire attachment point on upper mass *)`

- Define list of gravitational potential terms:
  - `gravlist = {}; (* initialize list *)`
  - `AppendTo[gravlist, m3 g z3]; (* typical item *)`

Defining a Model (iii)

- Define list of wires, each with the following format

- {
  - » *coordinate list defining first mass,*
  - » *attachment point for first mass (local coordinates),*
  - » *attachment vector for first mass,*
  - » *coordinate list defining second mass,*
  - » *attachment point for second mass (local coordinates),*
  - » *attachment vector for second mass,*
  - » *Young's modulus,*
  - » *unstretched length,*
  - » *longitudinal elasticity,*
  - » *vector defining principal axis 1,*
  - » *moment of area along principal axis 1,*
  - » *moment of area along principal axis 2,*
  - » *linear elasticity type,*
  - » *angular elasticity type,*
  - » *torsional elasticity type,*
  - » *shear modulus,*
  - » *cross sectional area for torsional calculations,*
  - » *torsional stiffness geometric factor*
- }

Defining a Model (iv)

- Define list of springs, each with following format:
- `{`
  - *coordinate list defining first mass,*
  - *attachment point for first mass (local coordinates),*
  - *attachment angles for first mass (yaw, pitch, roll),*
  - *coordinate list defining second mass,*
  - *attachment point for second mass (local coordinates),*
  - *attachment angles for second mass (yaw, pitch, roll),*
  - *damping type,*
  - *6x6 elasticity matrix,*
  - *1*6 pre-load force/torque vector*
- `}`
- Define kinetic energy
- `IM3 = {{I3x, 0, 0}, {0, I3y, 0}, {0, 0, I3z}}; (* typical MOI tensor)`
- `kinetic = (`
  - `…`
  - `+(1/2) m3 Plus@@(Dt[b2s[optic,COM],t]^2)`
  - `+(1/2) omegaB[yaw3, pitch3, roll3].IM3.omegaB[yaw3, pitch3, roll3]`
  - `…`
- `);`

Defining a Model (v)

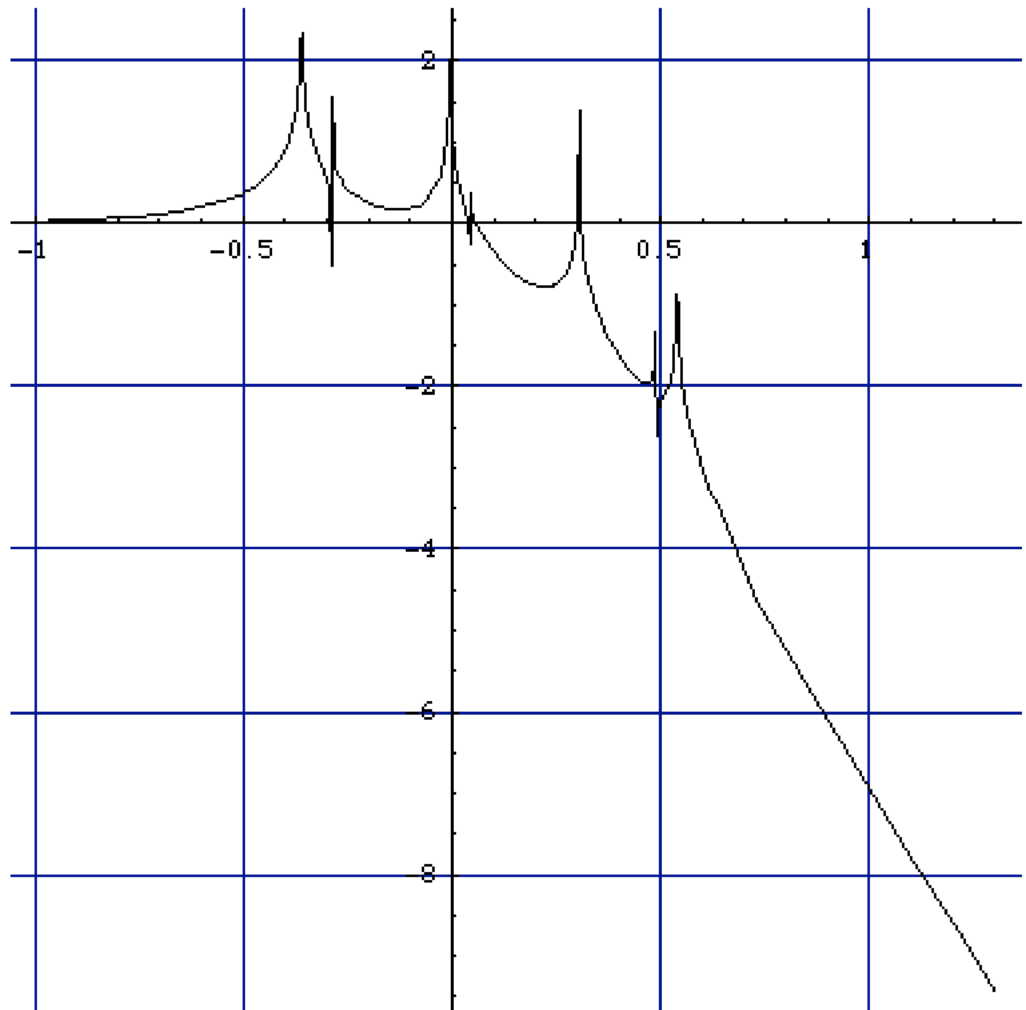
- Define default values of constants
- ```
  defaultvalues = {
  ```
  - » `g -> 9.81, (* value given numerically *)`
  - » …
  - » `m3 -> Pi*r3^2*t3, (* value given in terms of other constants *)`
  - » …
  - » `x00 -> 0, (* value for nominal position of structure *)`
  - » `y00 -> 0,`
  - » `z00 -> 0,`
  - » …
  - » `damping[imag,`*`dampingtype`*`] -> (phi&) (* value for frequency dependence of damping *)`
  - » …
- `};`
- Define starting point for finding equilibrium position:
- `startpos = {`
  - » `x1 ->0,`
  - » `y1 ->0,`
  - » …
- `};`
  ```

Defining a Model (vi)

- Define model-specific utilities:
  - » A function to list eigenmodes in a table
  - » `pretty[eigenvector]`
  - » A function to plot eigenmode shapes
  - » `eigenplot[`*`eigenvector`*`, `*`amplitude`*`, {`*`viewpoint`*`}]`
  - » Vectors representing force and displacement inputs and displacement outputs of interest
  - » `structurerollinput = makeinputvector[roll00];`
  - » `opticxinput = makefinputvector[x3];`
  - » `opticx = makeoutputvector[x3];`
  - » Rotation matrices to put angle variables in a more easily interpretable basis:
  - » `e2ni;`

Sample Output (i)

- Transfer function from x displacement of support to x motion of optic (quad model, reference parameters of 20031114):



Damping

- Damping can be represented by a complex elastic modulus: $k \to k_0\left(\varepsilon'(\omega) + i\varepsilon''(\omega)\right)$

- Strictly, the Kramers-Kronig relation applies:

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

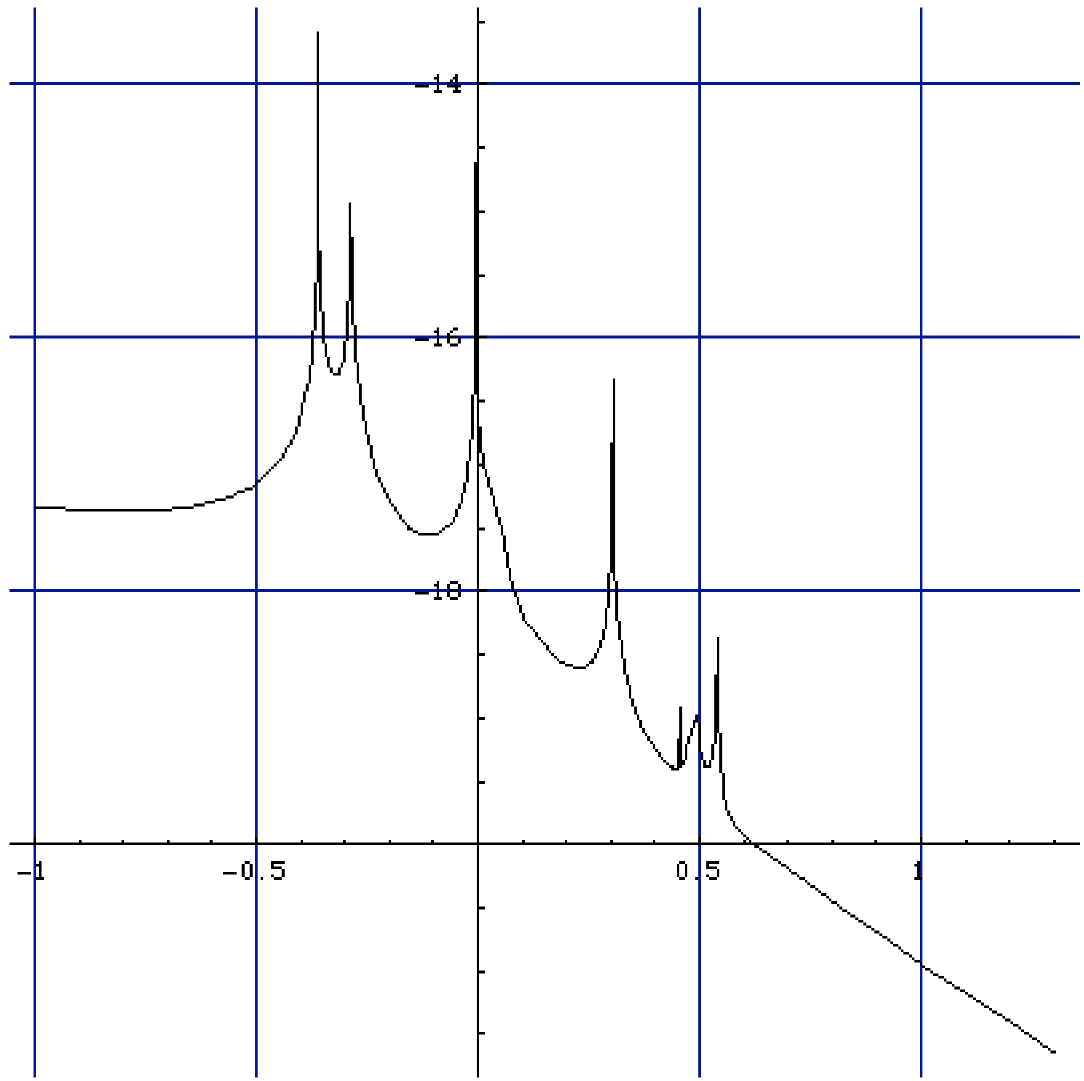
- However often the variation in the real part can be ignored:

$$k \to k_0\left(1 + i\phi(f)\right)$$

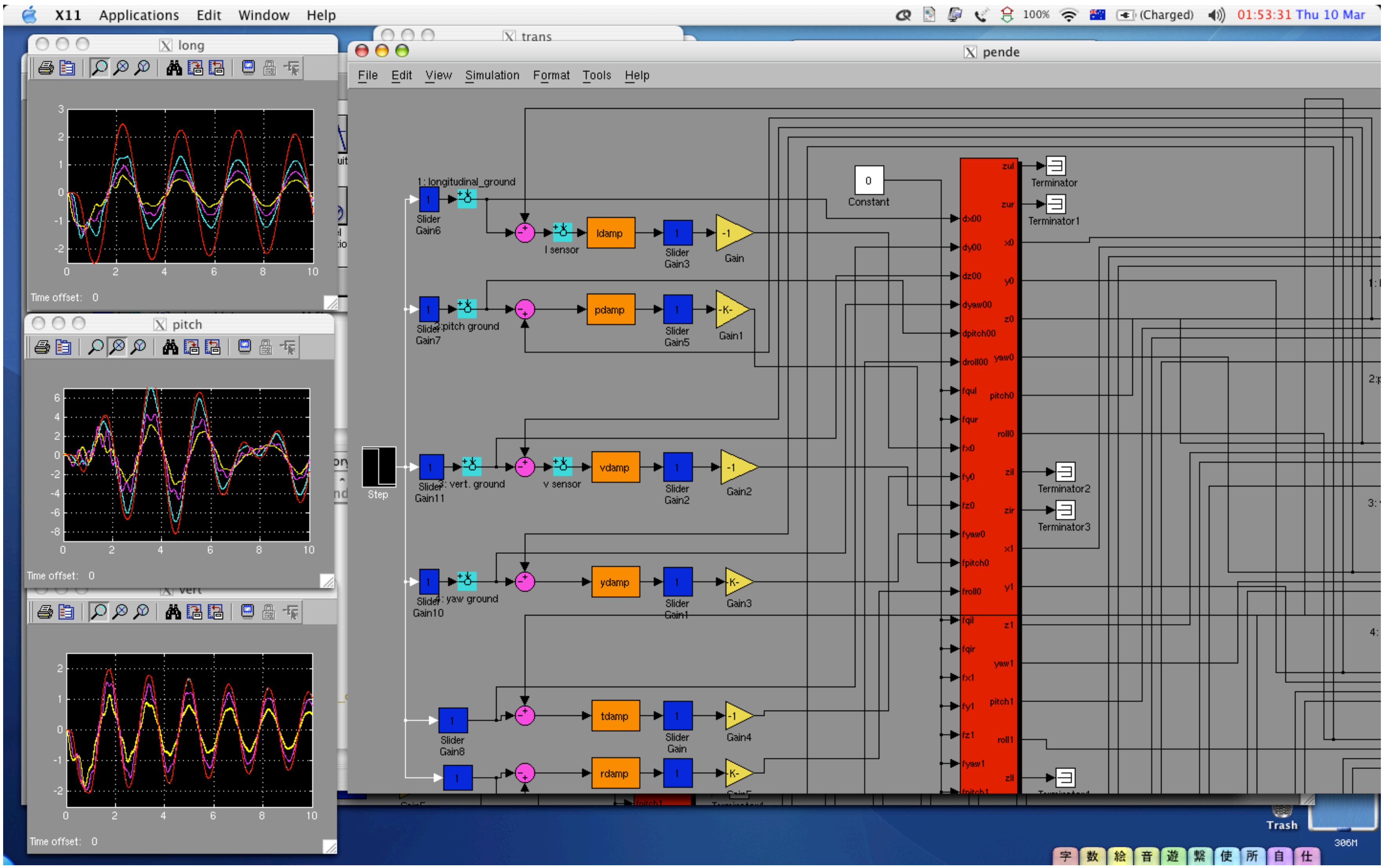
- Need to consider total potential as sum of terms, each with different damping: $\mathbf{P} = \sum \mathbf{P}_i\left(\varepsilon_i'(f) + i\varepsilon_i''(f)\right)$

Sample Output (ii)

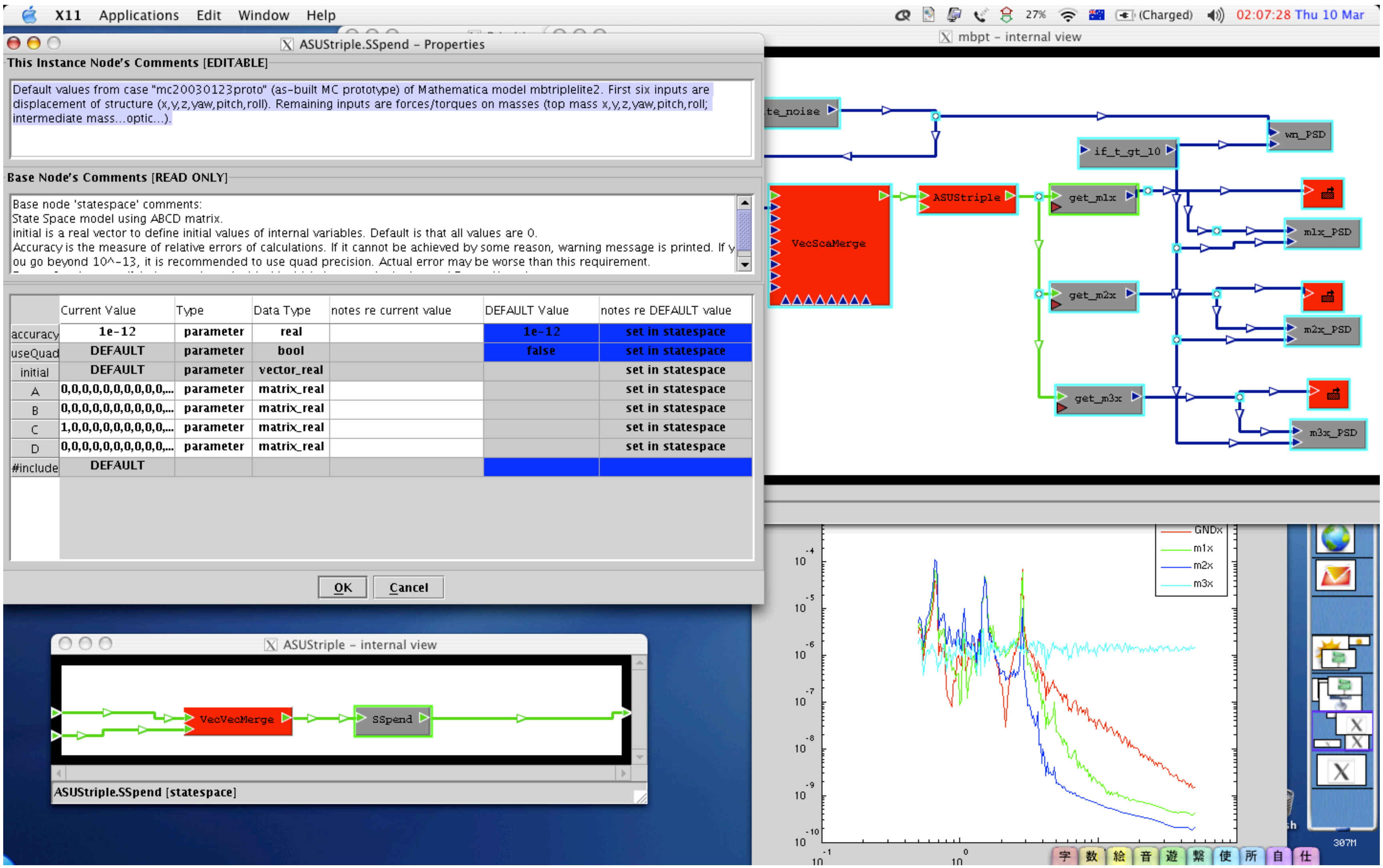
- Thermal noise in x motion of optic (quad model, reference parameters of 20031114):



Export to Matlab/Simulink



Export to E2E



Application to Quad Controls

- Good agreement after adding lots of new physics:
  - » Improved wire flexure correction
  - » Blade lateral compliance
  - » Blade geometric antispring effect
  - » Non-diagonal moment of inertia tensors

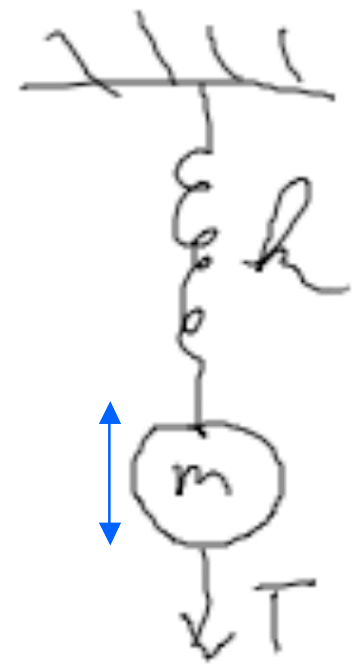
ID	f (theory)	f (exp)
pitch	0.395	0.403
x	0.443	0.440
y	0.464	0.464
z	0.595	0.549
yaw	0.685	0.684
roll	0.810	0.794
x	0.987	0.989
y	1.043	1.038
pitch	1.167	1.355
yaw	1.428	1.428
x	1.981	1.978
y	2.095	2.075
z	2.362	2.222
yaw	2.538	2.515
pitch	2.818	2.576
roll	2.762	2.734
yaw	3.167	3.149
pitch?	3.228	3.162
roll	3.332	3.333
x	3.401	3.381
z	3.793	3.589
roll	5.120	5.029
z	17.700	?
roll	25.741	?

Dissipation Dilution

- Often said: main restoring force in a pendulum is gravitational therefore no loss -> “dissipation dilution”
  - Not true!
- Gravitational force is purely vertical.
- Actual restoring force is sideways component of tension in wire
- Gravity’s only contribution is to tension the wire.
- Other forms of tension are equivalent (cf. violin modes also low-loss
- What is it about tension?

Non-dilution case (vertical)

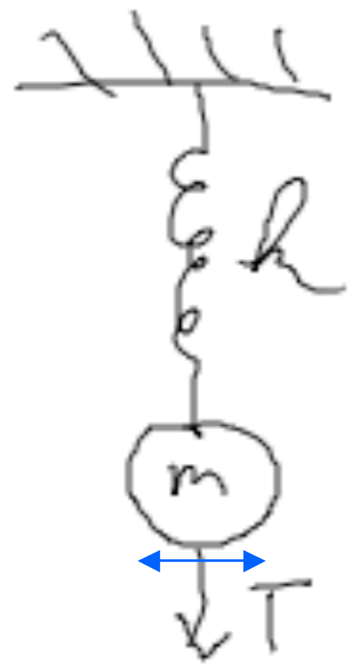
- Mass on spring
- Force: $F = -k(l - l_0)$
- Frequency: $\omega = \sqrt{\frac{k}{m}}$
- Amplitude (phasor): A_L
- Velocity (phasor): $v = i\omega A_L$
- Force (phasor): $F = -A_L k_0 (1 + i\phi)$
- Power (average): $P = Fv = A_L^2 k_0 \phi \omega$
- Energy (max): $E = \frac{1}{2} k_0 A_L^2$
- Decay time (energy): $\frac{2}{\phi\omega}$
- Decay time (amp.) $\frac{1}{\phi\omega}$



Dilution case (horizontal - exactly)

- Constrain mass to move exactly horizontally

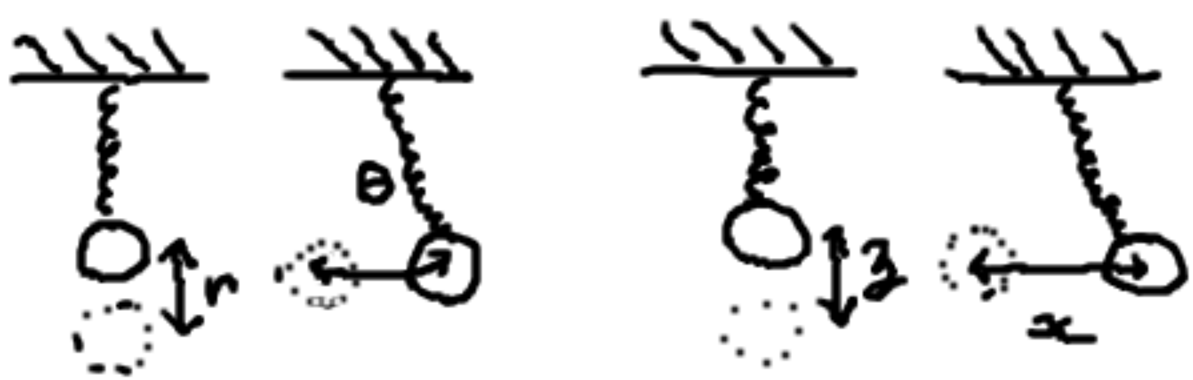
- Restoring force: $$F_T = T\sin\left(\frac{x}{l_{eq}}\right) \approx \frac{Tx}{l_{eq}}$$
- Spring constant: $$k_T = \frac{T}{l_{eq}}$$
- Frequency: $$\omega = \frac{T}{ml_{eq}} = \frac{mg}{ml_{eq}} = \frac{g}{l_{eq}}$$
- Length: $$l = \sqrt{l_{eq}^2 + x^2} \approx l_{eq} + \frac{x^2}{2l_{eq}} \approx l_{eq} + \frac{(A_T\cos\omega t)^2}{2l_{eq}} \approx l_{eq} + \frac{A_T^2}{2l_{eq}}\cos 2\omega t$$
- Power: $$P_T = \frac{A_T^4 k_0 \phi \omega}{2l_{eq}^2}$$
- Energy: $$E_T = \frac{k_0 (l_{eq} - l_0) A_T^2}{2l_{eq}} + O\left[A_T^4\right] \approx \frac{TA_T^2}{2l_{eq}} = \frac{k_T A_T^2}{2}$$



- Energy still 2nd order but power 4th order

But what about pendulums?!

- In a pendulum, mass really moves on an arc.
- Doesn't matter!
- Normal mode analysis can't tell the difference!
- Eigenmodes are always linear in coordinates used.
- Analyze in r,theta -> eigenmode is arc
- Analyze in x, z -> eigenmode is straight line
- Same frequencies!



What about pendulums (ii)

- Two independent reasons why pendulums have low loss.
  - » Restoring force is sideways component of tension
  - » Energy may then be off-loaded into gravitational potential -> stretch of spring less even than second order
- Depends on bounce and pendulum mode frequencies
  - » Usual case, bounce frequency high -> mass moves on arc.
  - » Very low bounce frequencies (superspring) -> mass really does move horizontally

Dissipation Dilution and Mathematica Toolkit

- Solution used in toolkit:
  - » Keep a separate stiffness matrix P_i for each elastic element
  - » For all elasticity types that depend on tension
  - » Compute potential matrix once normally
  - » Recompute with tension zeroed out.
  - » Apply damping to stiffness components that persist with tension off

$$\mathbf{P} = \sum \left(\mathbf{P}_i \big|_{tension\_off} \left(\varepsilon_i'(f) + i\varepsilon_i''(f) \right) \right) + \sum \left(\mathbf{P}_i \big|_{tension\_on} - \mathbf{P}_i \big|_{tension\_off} \right)$$

- Need to do analogous thing for ANSYS
- Difficult because detailed potential data not available, or at least not easy to access.

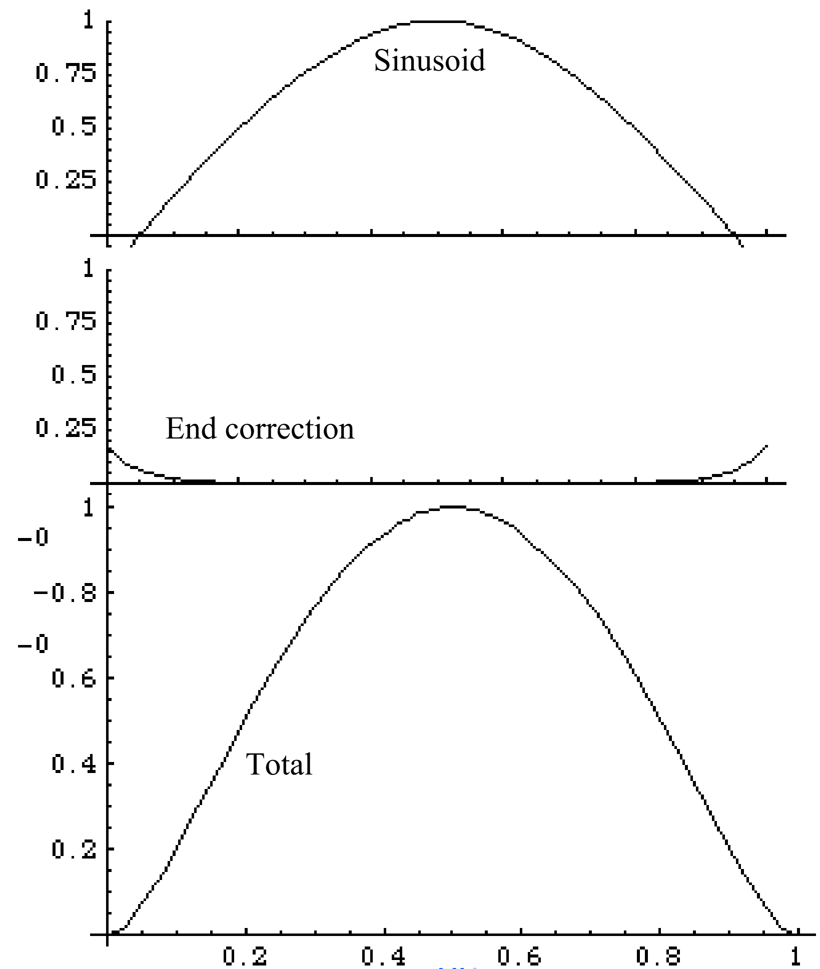
Test Case for ANSYS: Violin modes of a fibre

- Fibre under tension behaves as if shortened by flexure correction at each end
- Energy of two types
  - » Longitudinal stretching from bending out of straight line (low-loss)

$$E_{VL} = \frac{T}{2}\int_0^L \left(\frac{dy}{dx}\right)^2 dx$$

  - » Bending energy (lossy)

$$E_{VB} = \frac{YI}{2}\int_0^L \left(\frac{d^2 y}{dl^2}\right)^2 dl$$



Fibre results

- Fused silica, 350 mm long, 0.45 mm diameter
- Integrand of the two types
  - » Longitudinal ->
  - » Total 17.3 mJ for 10 mm amplitude
  - » Bending ->
  - » Total 0.256 mJ for 10 mm amplitude
- Dissipation dilution factor 67.6
- Will compare to ANSYS