Adding ECD to Matlab/ Mathematica SUS models

M. Barton

for SUS Testing Meeting 1/7/14,
revised 1/15/15

Damping for HAUX/HTTS

- There is a suite of Matlab models for different suspension types, including a single pendulum with blades for HAUX/HTTS.
- None of these have built-in eddy current damping – any damping has been added by wrapping a control system around the model in Simulink or the like.
- HAUX and HTTS have built-in ECD and it is inconvenient not having this as part of the model.
- Partial support was added a few months ago by hacking the state space A matrix. This was enough to support standard SUS testing by giving valid optic force/torque to optic displacement TFs.
- Rana et al. have requested support for structure displacement to optic displacement TFs. This was trickier, and required adding a set of new structure-velocity inputs and hacking the B matrix.
- The changes have been implemented but there is an outstanding bug.

Status Quo Ante

- Each Mathematica model exports symbolic matrix elements in Matlab format, e.g., `symbexport1bladesfull.m` for single model with blades.
- Damping had not been exported due to incompatible representations.
  - Mathematica uses frequency dependent complex matrixes acting on a state vector of (just) displacements.
    - Can easily represent arbitrary frequency dependence, including structural, thermoelastic and velocity damping terms.
    - Less convenient in time domain.
  - Matlab uses state-space formalism
    - Convenient for Matlab Simulink and linear analysis tools.
    - Can only represent velocity damping.
    - In principle, the velocity damping component of the total damping could be automatically exported from Mathematica, but historically hasn't been.
    - => Re-add from scratch.

Mathematica

- Equation of motion (T020205):

$$\mathbf{M}\ddot{\mathbf{x}} = -\mathbf{K}_{eff}\left(\mathbf{x}-\mathbf{x}_{eq}\right)+\mathbf{f}_x - \mathbf{C}_{XS(eff)}\left(\mathbf{s}-\mathbf{s}_{nom}\right)$$

$$\mathbf{x} = \mathbf{x}_{eq} + \frac{\mathbf{f}_x - \mathbf{C}_{XS(eff)}\left(\mathbf{s}-\mathbf{s}_{nom}\right)}{\mathbf{K}_{eff} - \left(2\pi f\right)^2 \mathbf{M}\left(\mathbf{x}-\mathbf{x}_{eq}\right)}$$

- All matrices complex and/or frequency-dependent.
- Solved numerically for individual values of f.
- ECD can be added as a spring with damping factor 0+2πif.
- Curious sign of coupling matrix $C_{XS(eff)}$ is artifact of definition as submatrix of master stiffness matrix – main entries are typically negative (see T020205).

Matlab (Old)

- Usual Matlab model uses a state-space formulation:
- State is optic displacements and velocities.
- Inputs are structure displacements and direct forces/ torques.
- Outputs are pendulum displacements and (optionally) reaction force.

$$\begin{pmatrix}\begin{pmatrix}\dot{\mathbf{x}}\\ \ddot{\mathbf{x}}\end{pmatrix}\\ \begin{pmatrix}\mathbf{x}\\ \mathbf{f}_s\end{pmatrix}\end{pmatrix} = \begin{pmatrix}\mathbf{A} & \mathbf{B}\\ \mathbf{C} & \mathbf{D}\end{pmatrix}\begin{pmatrix}\begin{pmatrix}\mathbf{x}\\ \dot{\mathbf{x}}\end{pmatrix}\\ \begin{pmatrix}\mathbf{s}\\ \mathbf{f}_x\end{pmatrix}\end{pmatrix}$$

- A/B equation becomes

$$\begin{pmatrix}\dot{\mathbf{x}}\\ \ddot{\mathbf{x}}\end{pmatrix} = \begin{pmatrix}\mathbf{0} & \mathbf{I}\\ -\mathbf{M}^{-1}\mathbf{K} & \mathbf{0}\end{pmatrix}\begin{pmatrix}\mathbf{x}\\ \dot{\mathbf{x}}\end{pmatrix} + \begin{pmatrix}\mathbf{0} & \mathbf{0}\\ -\mathbf{M}^{-1}\mathbf{C} & \mathbf{M}^{-1}\end{pmatrix}\begin{pmatrix}\mathbf{s}\\ \mathbf{f}_\mathbf{x}\end{pmatrix}$$

- C/D equation becomes

$$\begin{pmatrix}\mathbf{x}\\ \mathbf{f}_\mathbf{s}\end{pmatrix} = \begin{pmatrix}\mathbf{I} & \mathbf{0}\\ -\mathbf{C} & \mathbf{0}\end{pmatrix}\begin{pmatrix}\mathbf{x}\\ \dot{\mathbf{x}}\end{pmatrix} + \begin{pmatrix}\mathbf{0} & \mathbf{0}\\ -\mathbf{S} & \mathbf{0}\end{pmatrix}\begin{pmatrix}\mathbf{s}\\ \mathbf{f}_\mathbf{x}\end{pmatrix}$$

- Velocity damping should go here in A matrix.
- No $\dot{\mathbf{s}}$ input, so no place to put in B matrix.

Matlab (New – **A&B** stuff)

- Velocity damping corner now filled out in **A**:

$$\begin{pmatrix} \dot{\mathbf{x}} \\ \ddot{\mathbf{x}} \end{pmatrix} = \begin{pmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{M}^{-1}\mathbf{K} & -\mathbf{M}^{-1}\mathbf{E} \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ \dot{\mathbf{x}} \end{pmatrix} + \begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ -\mathbf{M}^{-1}\mathbf{C} & \mathbf{M}^{-1}\mathbf{E}\mathbf{O} & \mathbf{M}^{-1} \end{pmatrix} \begin{pmatrix} \mathbf{s} \\ \dot{\mathbf{s}} \\ \mathbf{f_x} \end{pmatrix}$$

- New $\dot{\mathbf{s}}$ input, corresponding new column in **B**
- **E** matrix maps relative velocities to damping forces/torques.
- **O** matrix maps velocities at structure origin to velocities near ECD.

$$\mathbf{E} = \begin{pmatrix} b_x & 0 & 0 & 0 & 0 & 0 \\ 0 & b_y & 0 & 0 & 0 & 0 \\ 0 & 0 & b_z & 0 & 0 & 0 \\ 0 & 0 & 0 & b_{yaw} & 0 & 0 \\ 0 & 0 & 0 & 0 & b_{pitch} & 0 \\ 0 & 0 & 0 & 0 & 0 & b_{roll} \end{pmatrix} \qquad \mathbf{O} = \begin{pmatrix} 1 & 0 & 0 & 0 & -\mathtt{tl}_0 & 0 \\ 0 & 1 & 0 & 0 & 0 & +\mathtt{tl}_0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

- $\mathtt{tl}_0$ is the "true length" from the suspension point down to the COM of the optic.

Bug described in –v1 of this presentation turned out to be a sign error in the LP term of O.

Matlab (New – **C&D** stuff)

- New **o** (for OSEM/ECD position) output, corresponding rows in **C** and **D**:

$$\begin{pmatrix} \mathbf{x} \\ \mathbf{o} \\ \mathbf{f_s} \end{pmatrix} = \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \\ -\mathbf{C} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ \dot{\mathbf{x}} \end{pmatrix} + \begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{O} & \mathbf{0} & \mathbf{0} \\ -\mathbf{S} & \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{s} \\ \dot{\mathbf{s}} \\ \mathbf{f_x} \end{pmatrix}$$

- New column in **D** for new $\dot{\mathbf{s}}$ input:
- As before with **A** and **B**,

$$\mathbf{O} = \begin{pmatrix} 1 & 0 & 0 & 0 & -\mathtt{tl}_0 & 0 \\ 0 & 1 & 0 & 0 & 0 & \mathtt{tl}_0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

maps structure origin position or velocity to the position/velocity near the optic, and $\mathtt{tl}_0$ is the "true length" from the suspension point down to the COM of the optic.

Matlab (New – other stuff)

- New fields in `pend` data structure in `hauxopt_damp.m`:

```
pend.B0xx = 0.5*pend.m0;             pend.B0yy = 1.*pend.m0;
pend.B0zz = 1.*pend.m0;              pend.B0yawyaw = 0.05*pend.I0z;
pend.B0pitchpitch = 2.*pend.I0y;     pend.B0roll = 1.*pend.I0x;
```

- New stuff in `ssmake1MBf_damp.m`:

```
E = diag([pend.B0xx,pend.B0yy,pend.B0zz,...
    pend.B0yawyaw,pend.B0pitchpitch,pend.B0rollroll]);
O = [1 0 0 0 -pend.tl0 0; 0 1 0 0 0 pend.tl0; 0 0 1 0 0 0; ...
       0 0 0 1 0 0; 0 0 0 0 1 0; 0 0 0 0 0 1];
```

```
mbsingleA = [...
  zeros(6) eye(6)
  -km\(xm-cqxm'/qm*cqxm) -km\E
];
```

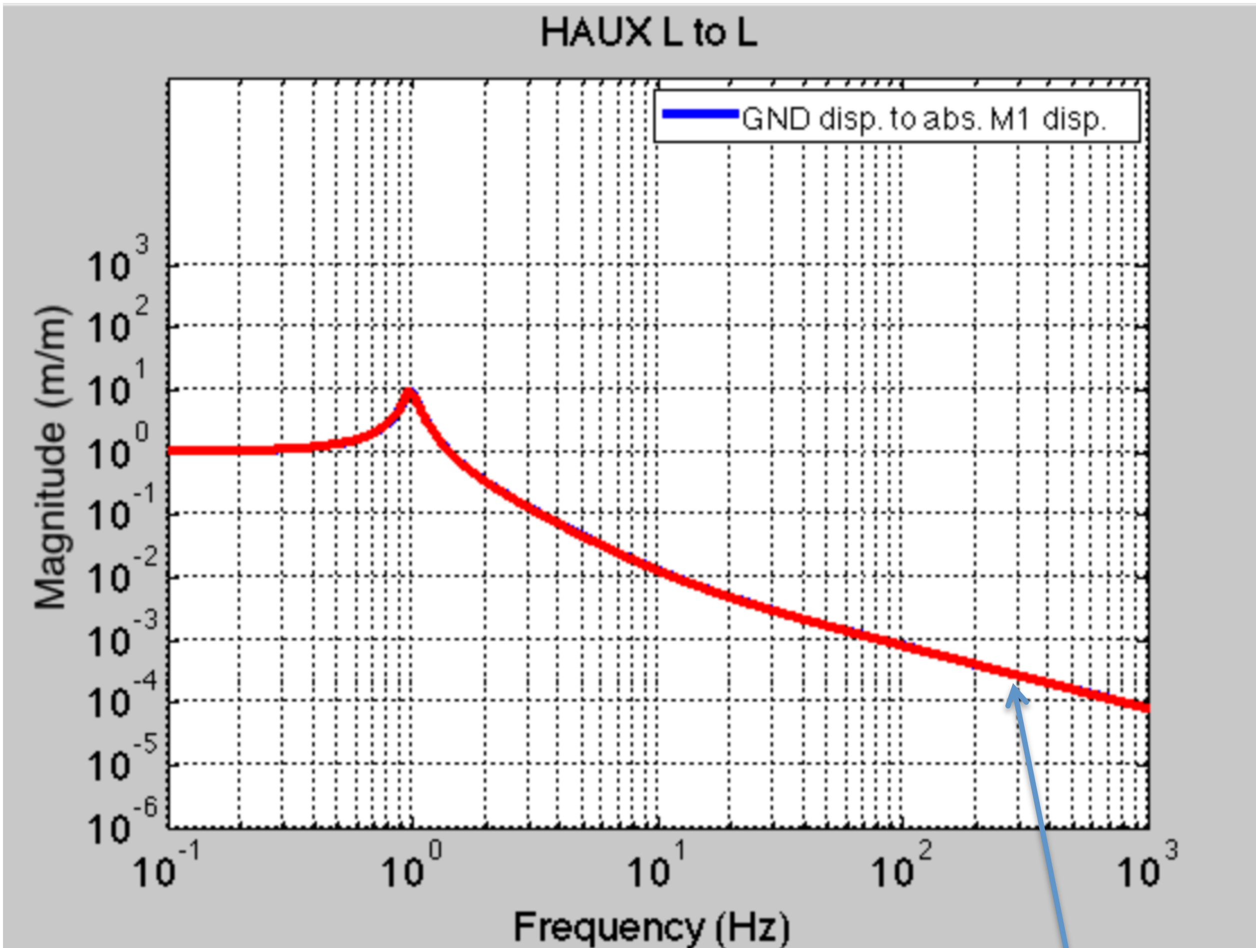
```
mbsingleC = [...
  eye(6) zeros(6,6)
  zeros(6,12)
];
```

```
bm1 = -km\(cxsm-cqxm'/qm*cqsm);
bm2 = km\(E*O);
bm3 = km\eye(6);
mbsingleB = [...
  zeros(6,18)
  bm1 bm2 bm3
];
```

```
mbsingleD = [...
  zeros(6,18)
  O zeros(6,12)
];
```

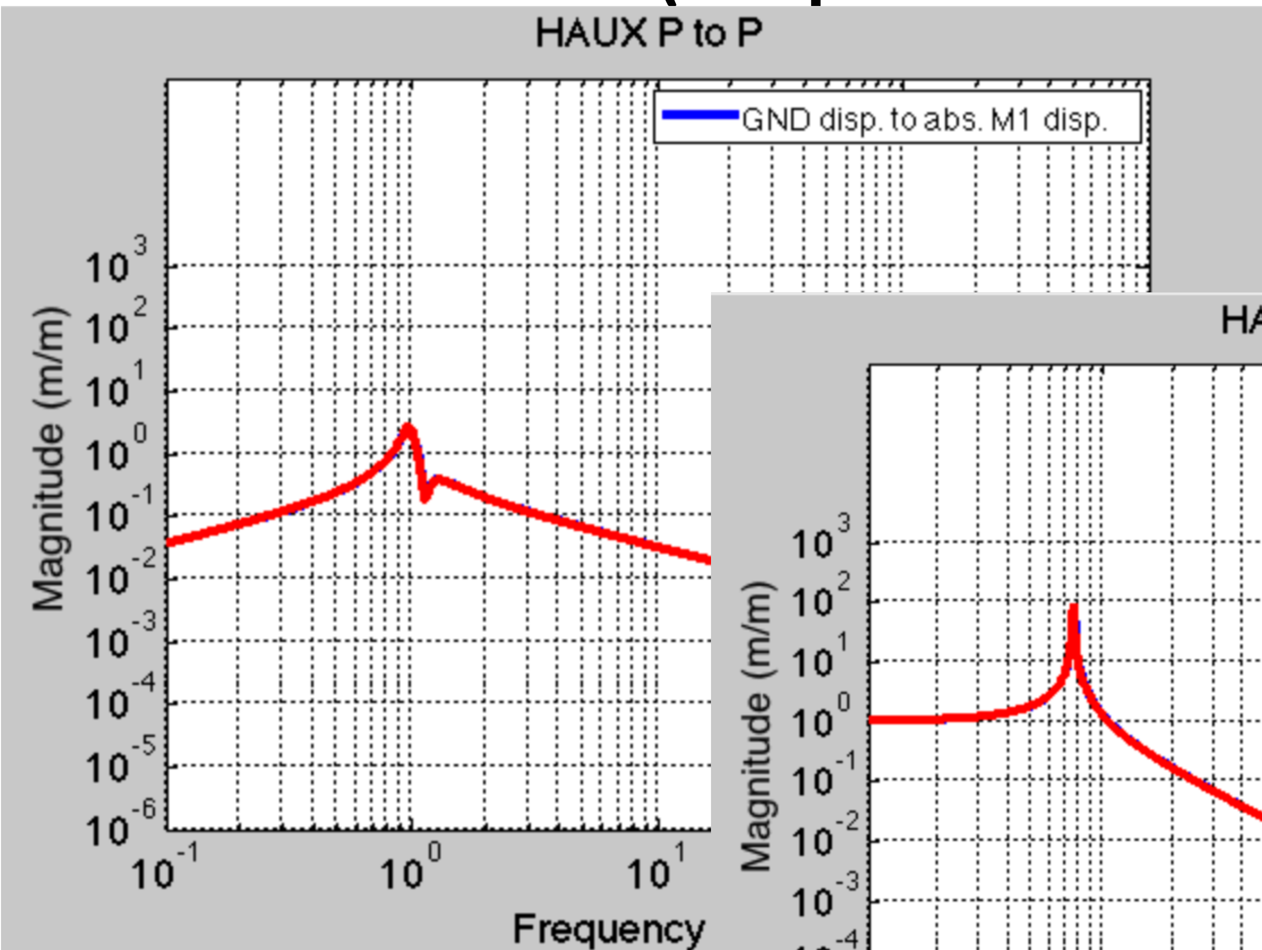
Results (L=x)

Perfect agreement between Mathematica and Matlab!

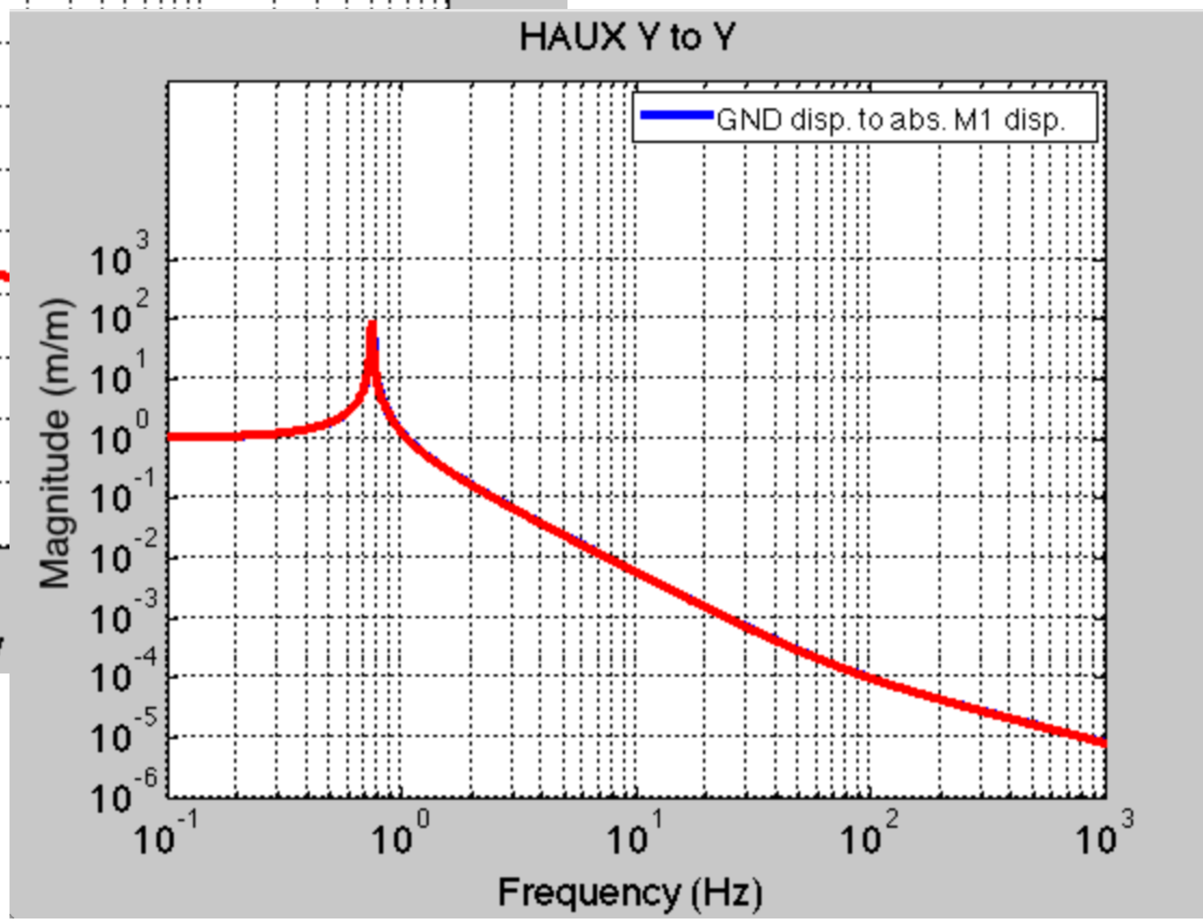


Note expected rolloff to 1/f at high frequency

Results (P=pitch and Y=yaw)



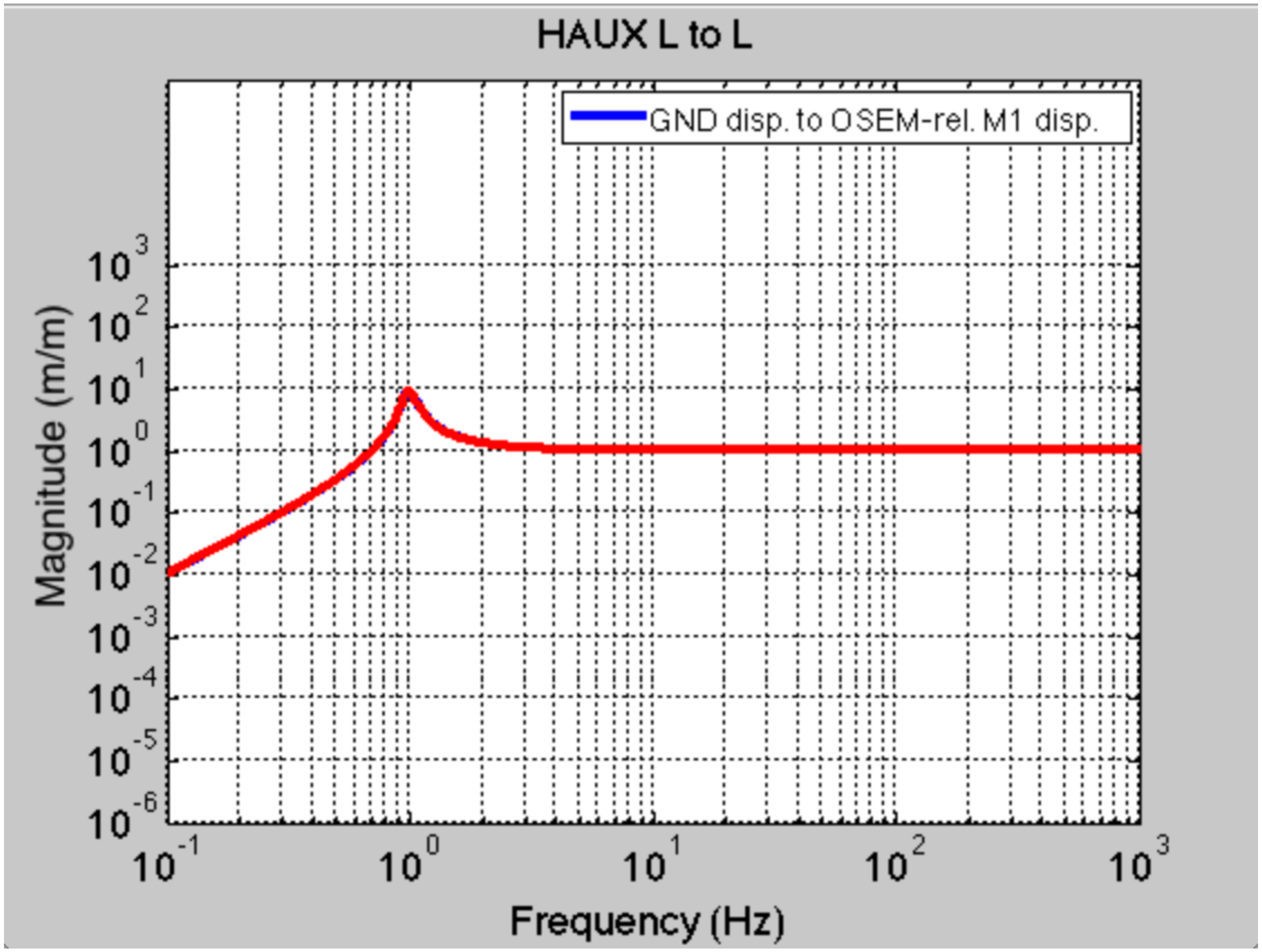
Also good agreement!



Bug reported for P in –v1 turned out to be sign error in LP term of E_B

Results (L as measured at OSEM)

New outputs allow easy plotting of displacement as measured at OSEMs



Note however, the OSEM coordinate outputs track the **s** input and ignore $\dot{\mathbf{s}}$ - make sure they're consistent!