Features in the Quad State Space Model

Contents

Send questions or comments to Edgard (edgard [at] stanford.edu)
and/or Brett (shapirob [at] stanford.edu)

Feature Summary

- **Built in features** – always included in the model
  - Relative SUS/cage sensors and actuators – new as of 1 Jan 2016
  - Suspension point reaction forces – new as of 1 Jan 2016

- **Optional features** – your choice to include or not
  - Two chain model – includes both a main and reaction chain
  - Violin modes
  - Damping
    - Import live damping from sites
    - Import damping from a prior GPS time from sites
    - Include optical lever damping (PUM and UIM actuation)
    - Load damping from a saved filter file or command space variable
    - PUM Pitch damping from PUM OSEMs.
  - UIM length force input filter (accounts for unmodeled dynamics).
  - SUS point displacement to TOP drive feedforward.

- **Possible future features** – there is always room for improvement
  - Global control
  - Radiation pressure

Note: Graphical Simulink representations of the model layouts are cited in step 1 of the '3 steps to edit the model features' section. These will provide more detail for some of the descriptions below.

Built in features:

- always included in the model -

Feature	Description
Relative SUS/cage sensors and actuators Added 1 Jan 2016	- The model incorporates the relative nature of the sensors and actuators. E.g. the damping OSEMSs see both the SUS and cage and drive both the SUS and cage. - Outputs exist to examine both true SUS displacement and the relative displacement seen by the sensors. Relative outputs at just the top for a single chain, all stages for two-chain models. - In the case of two-chain models, independent drive inputs exist for each chain at all stages, as well as differential inputs for the global control actuators.
Suspension point reaction forces Added 1 Jan 2016	Outputs exist for the reaction forces exerted by the top suspension wires at the suspension point. The damping reaction forces are also projected here. This permits the SUS model to be dynamically combined with the ISI model.

Optional features:

- your choice to include or not -

Feature	Description
Two chains	A model with both a reaction chain and main chain. The chains share seismic inputs and global control sensor/actuator signals. They also have independent force inputs and displacement outputs at all stages. Each chain has its own suspension point reaction force outputs.
Violin modes	You can compile the model with violin modes at any, or all, wires of your choosing. You specify the number of modes you want at each stage. The model simulates left and right wires so that the modes couple to length, yaw, and pitch. Violin modes are only available for the main chain. In the case of two-chain models, only the main chain receives the modes. You can specify measured mode frequencies and Qs by declaring them in the parameter file, e.g. h2etmy.m. The format for this is to add these variables, with frequency in Hz: fibers: fibermode_freq, fibermode_Q; uim-pum wires: uimpummode_freq, uimpummode_Q; top-uim wires: topuimmode_freq, topuimmode_Q; top wires: topmode_freq, topmode_Q. The way the model handles these is that any measured values in the parameter file simply replace the modeled values. Thus, if there are gaps in the data, e.g. measurements for modes 1 and 3, but not 2; modeled values fill in these gaps.

Optional features:

- your choice to include or not -

Feature	Description
Import live damping filters from sites	The damping loops currently running on a given suspension at a given site can be included in the model. A 5 min delay is built in to allow for latency. The can include top mass damping alone, or top mass with oplev damping. Some setup is required to make this work on your local computer.
Import damping filters from sites from a prior GPS time	Just like the live damping option, except the damping is loaded from a selected prior GPS time. The same setup is required.
Include custom filters from the command line or from a saved file	Make your own damping filters for the model. These are read in via a .mat file or from a command space variable. This must be a struct with fields L.c,T.c,V.c,Y.c,P.c,R.c, L2OL_P.c, L2OL_Y.c. For a saved file, the struct must be named calibFilter for historical reasons. From the command line the struct can be any name.

Optional features:

- your choice to include or not -

Feature	Description
Include PUM Pitch damping From PUM OSEMs	Applies a damping filter from the PUM (L2) Pitch OSEMs to the PUM Pitch actuation.
Include UIM length force input filter	This is a placeholder to account for the mysterious dynamics observed in the UIM length to test mass length transfer function (GSWG Log 11197) The hope is to replace this with dynamics we understand
Include a feedforward filter from SUS point displacement to TOP mass drive	Make a feedforward filter to try to compensate for the forces in the top mass due to suspension point displacement.

Note: Any combination of the optional features categories can be included in the model. E.g. you can make a model with two chains, violin modes, and damping; or a single chain with violin modes but no damping; etc.

Possible future features:

- there's always room for improvement -

No options yet exist for:

- **Global Control** - length and angular loops
- the inclusion of **Radiation Pressure** effects

These can be added as needed.

Where in the SVN does the model live?

The model files are found in

```
…/SusSVN/sus/trunk/QUAD/Common/MatlabTools/QuadModel_Production/
```

The model is compiled with the function

```
generate_QUAD_Model_Production.m
```

The supporting files called by this script are:

a) Simulink layout files:

```
generate_QUAD_SingleChainUndamped_Simulink.slx
generate_QUAD_SingleChainDamped_Simulink.slx
generate_QUAD_BothChainsUndamped_Simulink.slx
generate_QUAD_BothChainsDamped_Simulink.slx
```

b) Import damping filters from the sites:

```
M0LiveDampingFilters.mdl -> imports main chain damping from sites
R0LiveDampingFilters.mdl -> imports reaction chain damping from sites
L1L2_OplevLiveDampingFilters.mdl -> imports oplev damping from sites
```

Where in the SVN does the model live?

c) Adding violin modes:

```
makequad_with_modal_fibers.m  -> PUM-TST violin modes
makequad_with_modal_uimpum_wires -> UIM-PUM violin modes
makequad_with_modal_topuim_wires -> TOP-UIM violin modes
makequad_with_modal_top_wires.m -> SUS-TOP violin modes
```

d) Simulink diagrams and user interface management:

```
In_Out_Parser.m -> parses the input and output indices from simulink
QUAD_Model_input_options_template.m -> Template to call the model
ExtractUserOptions -> Manages the backwards compatible call to the script
UndampedChain.m -> Creates a class that compiles the undamped chain model
```

3 steps to edit the model features

1. **If you are changing the signal flow, edit the appropriate Simulink .slx layout file**

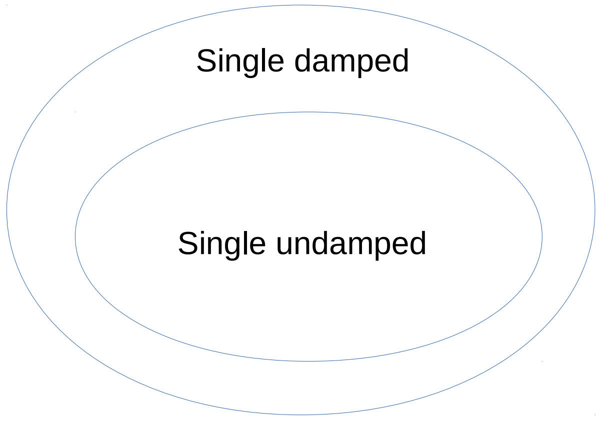
There are 4 possible layout files for the model depending on which features are called:

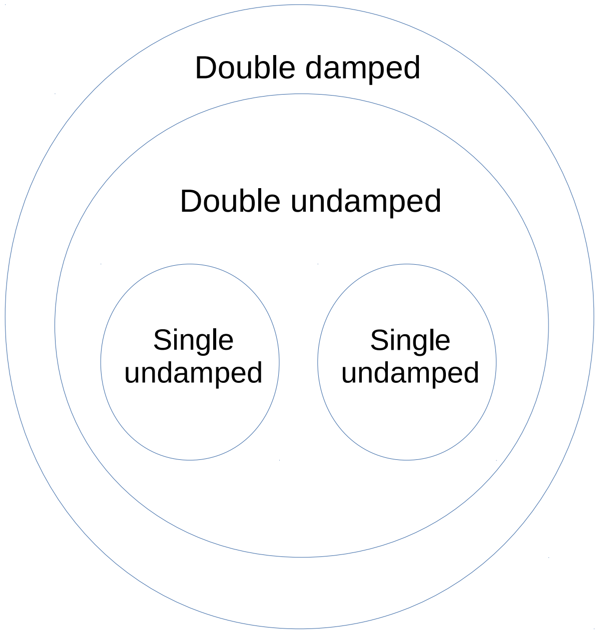
1) A single undamped chain: `generate_QUAD_SingleChainUndamped_Simulink.slx`
2) A single damped chain: `generate_QUAD_SingleChainDamped_Simulink.slx`
3) Two undamped chains: `generate_QUAD_BothChainsUndamped_Simulink.slx`
4) Two damped chains: `generate_QUAD_BothChainsDamped_Simulink.slx`

If new features are added to the single undamped chain, handle them by modifying the `UndampedChain.m` script. The goal is to keep the code as modular as possible.

3 steps to edit the model features

1b. The models are built in a hierarchical way. More complicated models are built around simpler ones. If one of the ‘inner’ models gets modified the changes propagate to the ‘outer’ models:





By modifying the single undamped chain, we modify every single model. Sometimes the outer diagrams have to be modified to account for changes in the inner ones. *(see step 3 for more information)*

Filters and features for the single undamped chain must be included and handled through the **UndampedChain.m** class script.

3 steps to edit the model features

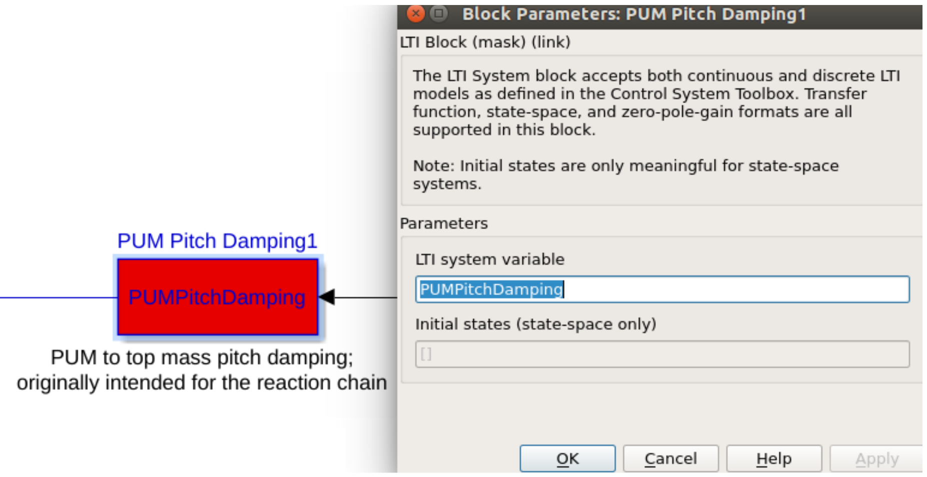
1c. If you are adding new filters that might be imported from the sites then you must also add these to the import list in the appropriate .mdl file, in addition to the layout file. These are separate from the layout because the user might supply their own filter designs.
Import files currently exist only for top mass and optical lever damping. Make new files as needed.

• main chain damping filters for import:	`M0LiveDampingFilters.mdl`
• reaction chain damping filters for import:	`R0LiveDampingFilters.mdl`
• Oplev damping filters for import:	`L1L2_OplevLiveDampingFilters.mdl`

3 steps to edit the model features

2. If new filters or variables are added to the layout .slx file, they must be passed to the layout by
`generate_QUAD_Model_Production.m`
Or
`UndampedChain.m` *(if it is the single undamped chain)*

Each variable appearing in a simulink diagram has to be passed to the workspace with its corresponding name in order to be compiled.


PUM Pitch Damping1
PUMPitchDamping
PUM to top mass pitch damping; originally intended for the reaction chain
Block Parameters: PUM Pitch Damping1
LTI Block (mask) (link)

The LTI System block accepts both continuous and discrete LTI models as defined in the Control System Toolbox. Transfer function, state-space, and zero-pole-gain formats are all supported in this block.

Note: Initial states are only meaningful for state-space systems.

Parameters

LTI system variable: PUMPitchDamping

Initial states (state-space only): []

Pass the corresponding filter from inside the script by using the line:

```
assignin('base','PUMPitchDamping',options.PUMPitchDamping)
```

In this case the filter comes directly from the user input struct

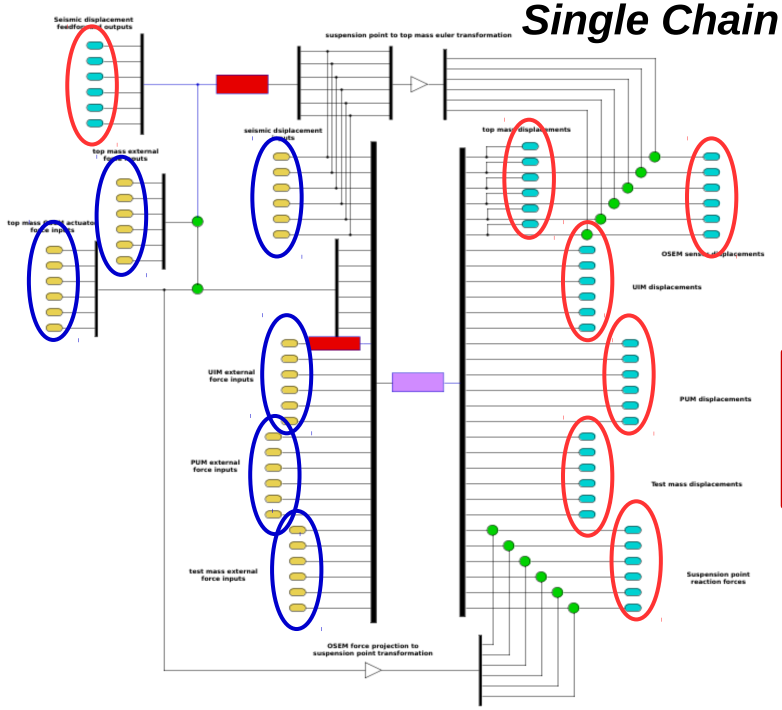
3 steps to edit the model features

3. If the inputs and outputs are modified, make the appropriate changes in the ‘outer’ simulink diagrams:

While the inputs and outputs are parsed directly by the script, the hierarchical implementation requires to manually make sure that the inputs and outputs of the ‘inner’ models merge well with the ones on the ‘outer’ models wrapped around them.

Single Chain undamped model

The single undamped chain model has 36 numbered inputs


Seismic displacement feedforward outputs
suspension point to top mass euler transformation
seismic dsiplacement inputs
top mass displacements
top mass external force inputs
top mass OSEM actuator force inputs
OSEM sensor displacements
UIM displacements
UIM external force inputs
PUM external force inputs
PUM displacements
Test mass displacements
test mass external force inputs
Suspension point reaction forces
OSEM force projection to suspension point transformation

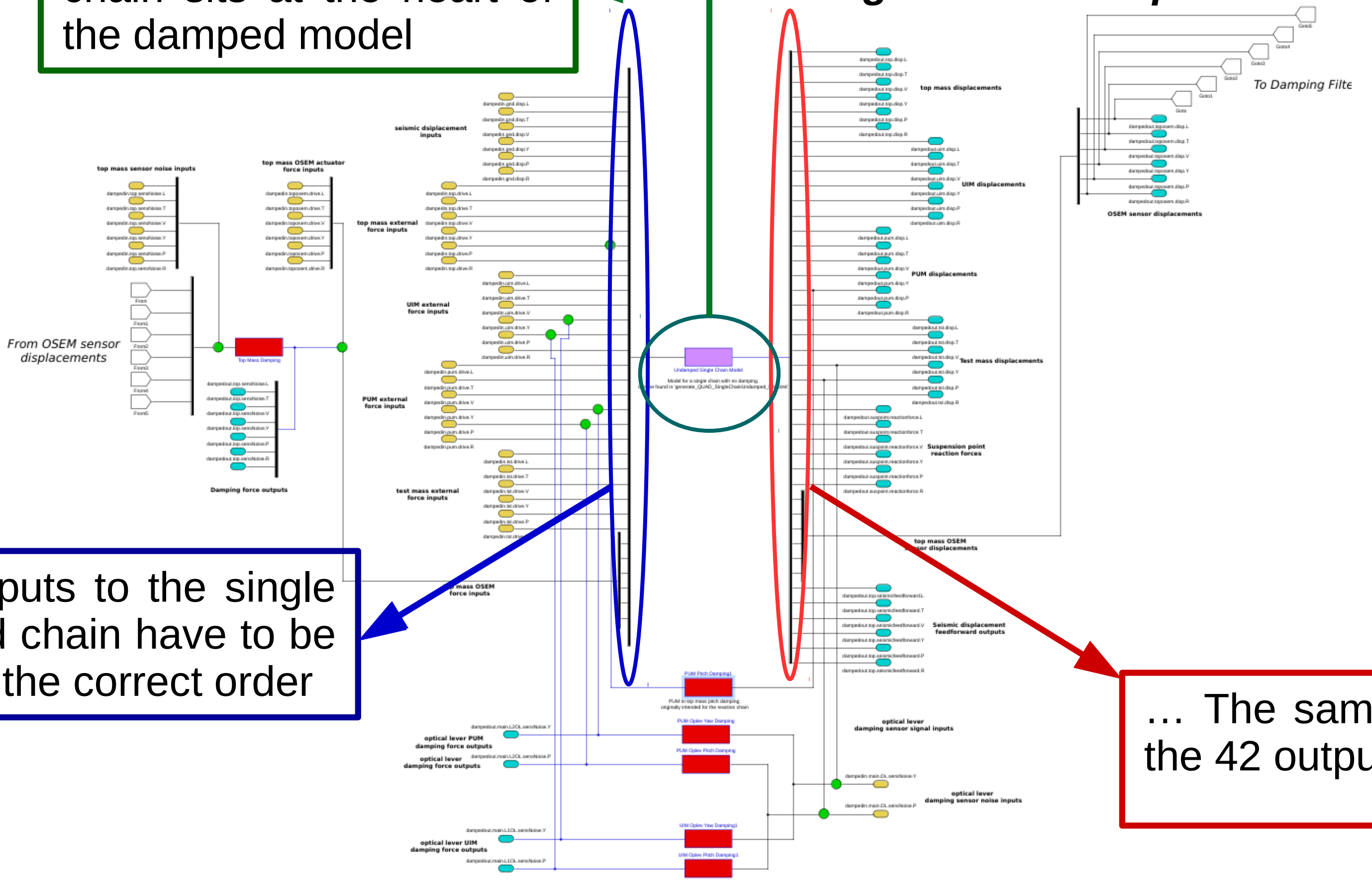
… And 42 numbered outputs

3 steps to edit the model features

3. If the inputs and outputs are modified, make the appropriate changes in the ‘outer’ simulink diagrams:

The single undamped chain sits at the heart of the damped model

Single Chain damped model



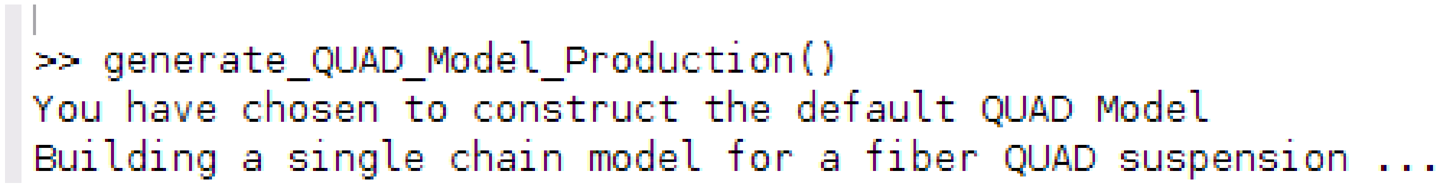
To Damping Filte
From OSEM sensor displacements
top mass sensor noise inputs
top mass OSEM actuator force inputs
seismic dsiplacement inputs
top mass external force inputs
UIM external force inputs
PUM external force inputs
test mass external force inputs
Damping force outputs
top mass displacements
UIM displacements
PUM displacements
Test mass displacements
OSEM sensor displacements
Suspension point reaction forces
Seismic displacement feedforward outputs
optical lever damping sensor signal inputs
optical lever damping sensor noise inputs
optical lever PUM damping force outputs
optical lever damping force outputs
optical lever UIM damping force outputs
Top Mass Damping
Undamped Single Chain Model

The 36 inputs to the single undamped chain have to be fed to it in the correct order

… The same applies for the 42 outputs.

How to call the model

1. **The model can be called with no inputs, which gives the default build:**



```
>> generate_QUAD_Model_Production()
You have chosen to construct the default QUAD Model
Building a single chain model for a fiber QUAD suspension ...
```

This call creates the default quadruple suspension model:

- Single chain fiber build (No reaction chain).
- No feedforward filters or feedback loops are applied.
- Does not include any of the UIM Mysterious Dynamics.
- No violin modes are applied.

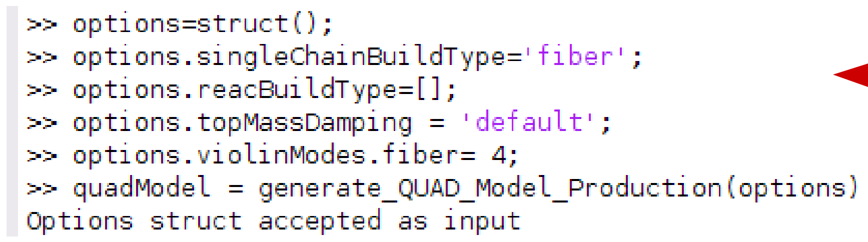
The Options struct call modifies the default settings with the user defined options.

(See Next Slide)

How to call the model

2. The model runs by using an ‘options’ struct to modify the default behavior:

Detailed instructions for the model call, and a template for managing the features can be found in: `QUAD_Model_input_options_template.m`.



```
>> options=struct();
>> options.singleChainBuildType='fiber';
>> options.reacBuildType=[];
>> options.topMassDamping = 'default';
>> options.violinModes.fiber= 4;
>> quadModel = generate_QUAD_Model_Production(options)
Options struct accepted as input
```

Create an options struct and fill the fields with the options you want

Call the script using this struct as input

Options:
- Single chain fiber QUAD suspension
- The default top mass damping filters are applied
- The first 4 violin modes between test mass and PUM are included

Any option not set manually is going to get set to the function’s default behavior.
(see the previous slide)

For more information check
`QUAD_Model_input_options_template.m`.

How to call the model

3. Backwards compatible call:
(New features cannot be accessed this way)

Detailed instructions for this call are commented into the header of `generate_QUAD_Model_Production.m`.

Here is an example of how to call the model with the old supported features:

Violin modes on

The function precomputes frequency responses over this frequency vector

Local SVN directory

```
quadModel = generate_QUAD_Model_Production(freq_vec,'fiber_with_erm',SvnDir,0,{'top','L2OL'},'liveh1etmy',
0,[],1,[10 10 5 5])
```

1st ten violin modes on fiber and PUM wires, 1st 5 at UIM and top wires

No plots
(not supported)

Import live damping from H1ETMY

Modal damping excluded
(not supported)

Two chains: main with fibers, and a reaction ERM

Top mass and oplev damping

Model Output:

The model output is a matlab struct containing the information about the state space model for the options the user requested. The main output subfields are:

1. **ss (dampedss):**
   State space model relating the inputs and outputs of the QUAD suspension. **‘ss’** corresponds to the undamped version of the system, **‘dampedss’** is the model if the damping loops are turned on. These can be used to extract transfer functions and other information for the QUAD suspension.
2. **in / out (dampedin/dampedout):**
   These subfields organize the numbering of the inputs and outputs for the state space **‘ss’** (resp. **‘dampedss’**). They contain subfield names indicating the specific input or output and they can be used to easily interface with the state space model.

   As an example we will show how to get a ***Transfer Function*** *bode plot by using these structs in the next slide:*

Model Output:

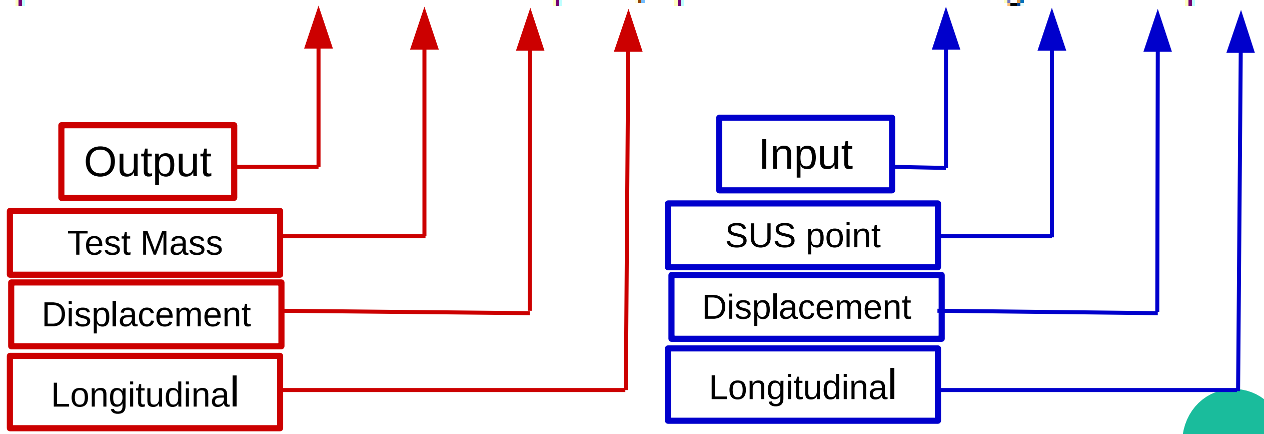
Getting a **Transfer Function** from output:

a) Run the model script

```
>> quadModel = generate_QUAD_Model_Production();
You have chosen to construct the default QUAD Model
Building a single chain model for a fiber QUAD suspension ...
     Using QUAD model:ssmake4pv2eMB5f_fiber
     Using Params file: quadopt_fiber
Calculating the frequency response of the open loop model ...
   Finished open loop in 0.014798 seconds.
```

b) Use **bode** or **bode2** with the **ss** (*resp.* **dampedss**) field of the model:

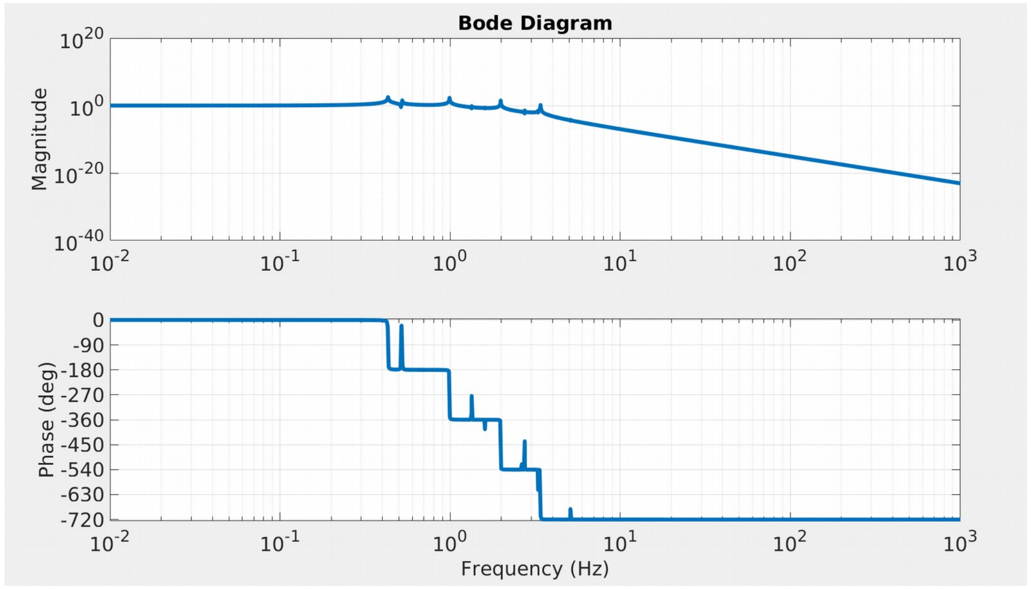
```
>> bode(quadModel.ss(quadModel.out.tst.disp.L,quadModel.in.gnd.disp.L))
```



Model Output:

Getting a **Transfer Function** from output:

c) See the output:



d) If you have a model with damping loops active, you can compare the performance with and without them:

```
>> bode2(quadModel.ss(quadModel.out.tst.disp.L,quadModel.in.gnd.disp.L))
```

Undamped

```
>> bode2(quadModel.dampedss(quadModel.dampedout.tst.disp.L,quadModel.dampedin.gnd.disp.L))
```

Damped

Function output from example model call

```
Building a model for a fiber_with_erm QUAD suspension ...
     Using QUAD model:ssmake4pv2eMB5f_fiber
     Using Params file: quadopt_fiber
     Using QUAD model:ssmake4pv2eMB4f_erm
     Using Params file: quadopt_erm
Adding the first 10 fiber violin modes with viscous damping to the model...
Adding the first 10 UIM-PUM wire violin modes with viscous damping to the model...
Adding the first 5 Top-UIM wire violin modes with viscous damping to the model...
Adding the first 5 Top wire violin modes with viscous damping to the model...
Combining main and reaction chain models...
Calculating the frequency response of the open loop model ...
   Finished open loop in 0.07709 seconds.
Closing the loop ...
reading filters for H1ETMY from GPS time 1140933834

loading damping electronics gain of 1.17. This is harcoded in the function with the variable
Damping_electronics_gain!

6 LiveParts found
    M0LiveDampingFilters/M0_DAMP_L :: 4 channels
    M0LiveDampingFilters/M0_DAMP_P :: 4 channels
    M0LiveDampingFilters/M0_DAMP_R :: 4 channels
    M0LiveDampingFilters/M0_DAMP_T :: 4 channels
    M0LiveDampingFilters/M0_DAMP_V :: 4 channels
    M0LiveDampingFilters/M0_DAMP_Y :: 4 channels
Connecting to NDS server nds2.ligo-wa.caltech.edu
Fetching 24 channels, start GPS 1140933834, duration 1 sec
Downloading filter file H1SUSETMY.txt for GPS 1140933835
2 LiveParts found
    L2OplevLiveDampingFilters/L2_OLDAMP_P :: 4 channels
    L2OplevLiveDampingFilters/L2_OLDAMP_Y :: 4 channels
Connecting to NDS server nds2.ligo-wa.caltech.edu
Fetching 8 channels, start GPS 1140933834, duration 1 sec
Reusing results of downloadFilterFile from a previous run (see "help cacheFunction" for details)
6 LiveParts found
    R0LiveDampingFilters/R0_DAMP_L :: 4 channels
    R0LiveDampingFilters/R0_DAMP_P :: 4 channels
    R0LiveDampingFilters/R0_DAMP_R :: 4 channels
    R0LiveDampingFilters/R0_DAMP_T :: 4 channels
    R0LiveDampingFilters/R0_DAMP_V :: 4 channels
    R0LiveDampingFilters/R0_DAMP_Y :: 4 channels
Connecting to NDS server nds2.ligo-wa.caltech.edu
Fetching 24 channels, start GPS 1140933834, duration 1 sec
Reusing results of downloadFilterFile from a previous run (see "help cacheFunction" for details)

Calculating the frequency response of the closed loop model ...
   Finished closed loop in 0.1231 seconds.

quadModel =

             dampFilters: [1x1 struct]
     mainchain_modelName: 'ssmake4pv2eMB5f_fiber'
    mainchain_paramsName: 'quadopt_fiber'
     reacchain_modelName: 'ssmake4pv2eMB4f_erm'
    reacchain_paramsName: 'quadopt_erm'
           hasFiberModes: '1 through 10 with viscous damping'
    hasUIMPUM_Wire_Modes: '1 through 10 with viscous damping'
    hasTopUIM_Wire_Modes: '1 through 5 with viscous damping'
       hasTop_Wire_Modes: '1 through 5 with viscous damping'
                      ss: [81x75 ss]
    mainchain_pendParams: [1x1 struct]
    reacchain_pendParams: [1x1 struct]
                      in: [1x1 struct]
                     out: [1x1 struct]
                       f: [81x75x2 double]
                dampedin: [1x1 struct]
               dampedout: [1x1 struct]
         maindampFilters: [1x1 struct]
           OLdampFilters: [1x1 struct]
         reacdampFilters: [1x1 struct]
                dampedss: [95x89 ss]
                 dampedf: [95x89x2 double]
```