KAGRA Type-C suspension simulation

This JupyterNotebook generates transfer functions of Type-C suspensions of KAGRA.

The simulation code was originally developed by K. Arai for TAMA suspensions.

Preparation

Install Wolfram Engine

This notebook is written in Wolfram language (not Python).

In order to run the notebook, you need to install Wolfram Engine from here:
https://www.wolfram.com/engine/index.ja.php?source=footer

Then install WolframLanguageForJupyter from here:
https://github.com/WolframResearch/WolframLanguageForJupyter

Please follow the instruction there.

Coordinate system definition

Load the simulation kernel

Load the core simulation code.

In [365]:
```
Get[FileNameJoin[{"dp2D_kernel.3.0.m"}]]
```

Test Mass Type

There are two variations in Type-C: TM type and PO type. TM type is used for MCe, IMMT1, IMMT2, OMMT1 and OMMT2. PO type is used for MCi, MCo and OSTM1.

In this section, we will compute the TFs of TM type suspensions.

Load Geometry

In [366]:
```
<<"TM_Type_geometry.m"
```

Find the equilibrium points

In [367]:
```
findEquivXY;
findEquivYZ;
findEquivZX;
```

```
1: {0, -63.705, 0, 0, 63.5298, 0}
2: {0, -0.00899875, 0, 0, 0.00899875, 0}
```

```
3: {0, -1.9042 10^-10, 0, 0, 1.9047 10^-10, 0}
```

Set the EQOM matrices

In [370]:

```
setMat
```

Eigen frequencies

In [371]:

```
Map[MatrixForm, {eigenFreqxy, eigenFreqyz, eigenFreqzx}]
```

Out[371]:

$$\left\{\begin{pmatrix} 0.951544 & 4.53665 \\ 1.47641 & 2.2259 \\ 1.4783 & 3.49672 \\ 1.68502 & 0.582449 \\ 1.93977 & 0.300882 \\ 1.94388 & 0.308963 \\ 3.60058 & 2.50066 \\ 3.62733 & 3.17731 \\ 20.5036 & 17.3268 \end{pmatrix}, \begin{pmatrix} 0.950993 & 4.55488 \\ 1.93931 & 0.300536 \\ 3.5806 & 2.51412 \\ 5.18673 & 3.99282 \\ 6.86533 & 1.27776 \\ 7.11209 & 5.34938 \\ 10.6844 & 8.97122 \\ 23.6514 & 16.6021 \\ 30.88 & 51.3536 \end{pmatrix}, \begin{pmatrix} 0.951448 & 4.53758 \\ 1.65554 & 1.02925 \\ 1.94012 & 0.300623 \\ 3.59292 & 2.50926 \\ 5.18673 & 3.99282 \\ 6.86533 & 1.27776 \\ 7.94257 & 5.07256 \\ 14.794 & 12.9182 \\ 23.6514 & 16.6021 \end{pmatrix}\right\}$$

Calculate vibration isolation ratio

In [372]:

```
calcVIRxy;
calcVIRyz;
calcVIRzx;
```

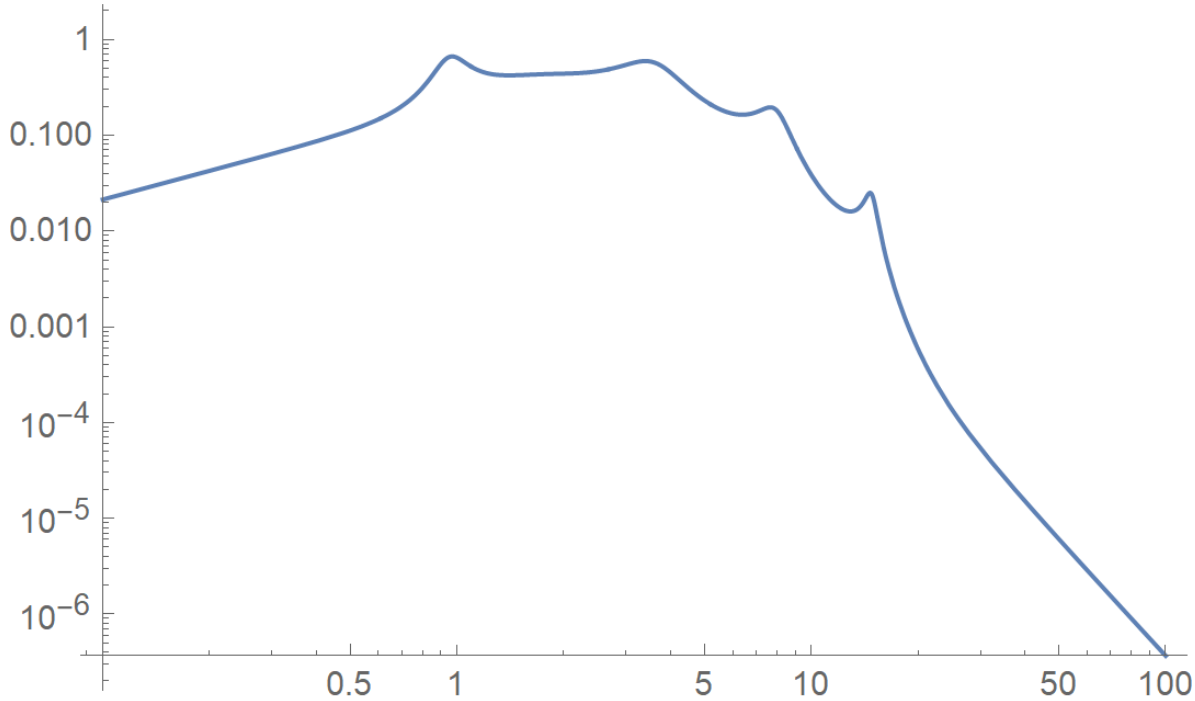
```
iso{X|Y|Zeta}{X|Y|Zeta}{IFM}xy have been set.
iso{Y|Z|Xi}{Y|Z|Xi}{IFM}yz have been set.
iso{Z|X|Eta}{Z|X|Eeta}{IFM}zx have been set.
```

Plot vibration isolation ratio for Pitch

In [375]:

```
LogLogPlot[Abs[isoXEtaFzx], {f, 0.1, 100}]
```

Out[375]:



Calculate actuation transfer functions

In [377]:

```
calcActxy;
calcActyz;
calcActzx;
```

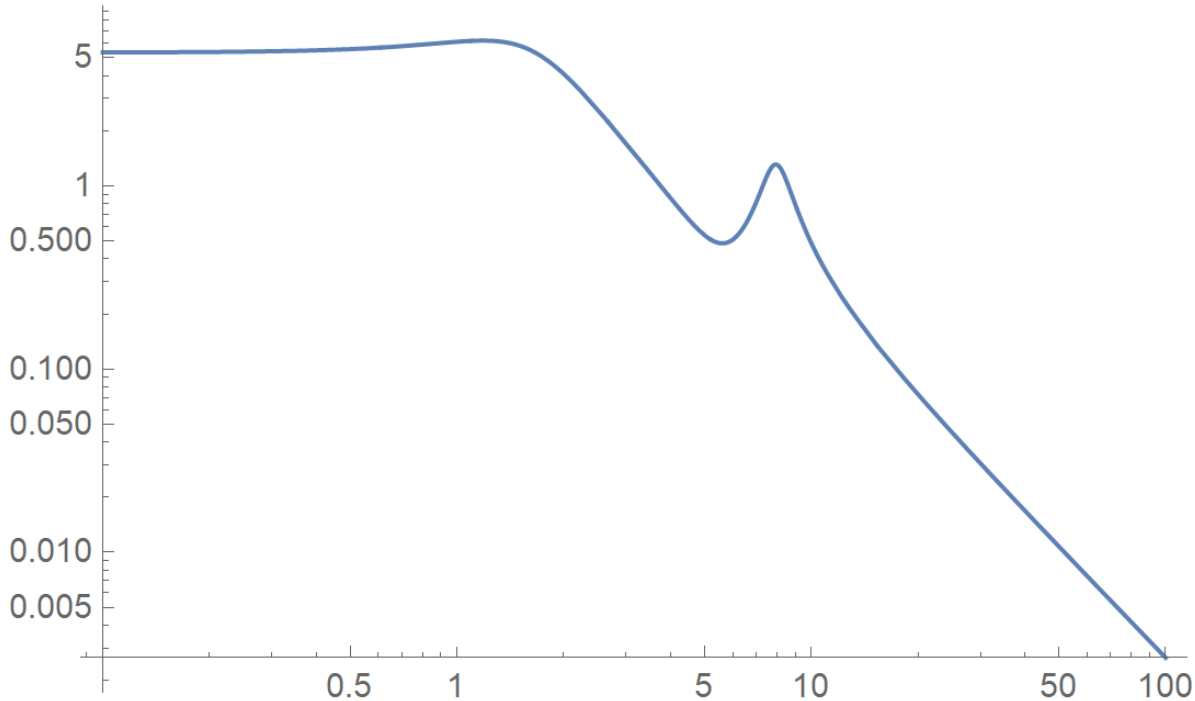
```
act{X|Y|Zeta}{X|Y|Zeta}{IFM}xy have been set.
act{Y|Z|Xi}{Y|Z|Xi}{IFM}yz have been set.
act{Z|X|Eta}{Z|X|Eeta}{IFM}zx have been set.
```

Actuation TF: Pitch -> Pitch

In [380]:

```
LogLogPlot[Abs[actEtaEtaFzx], {f,0.1,100}]
```

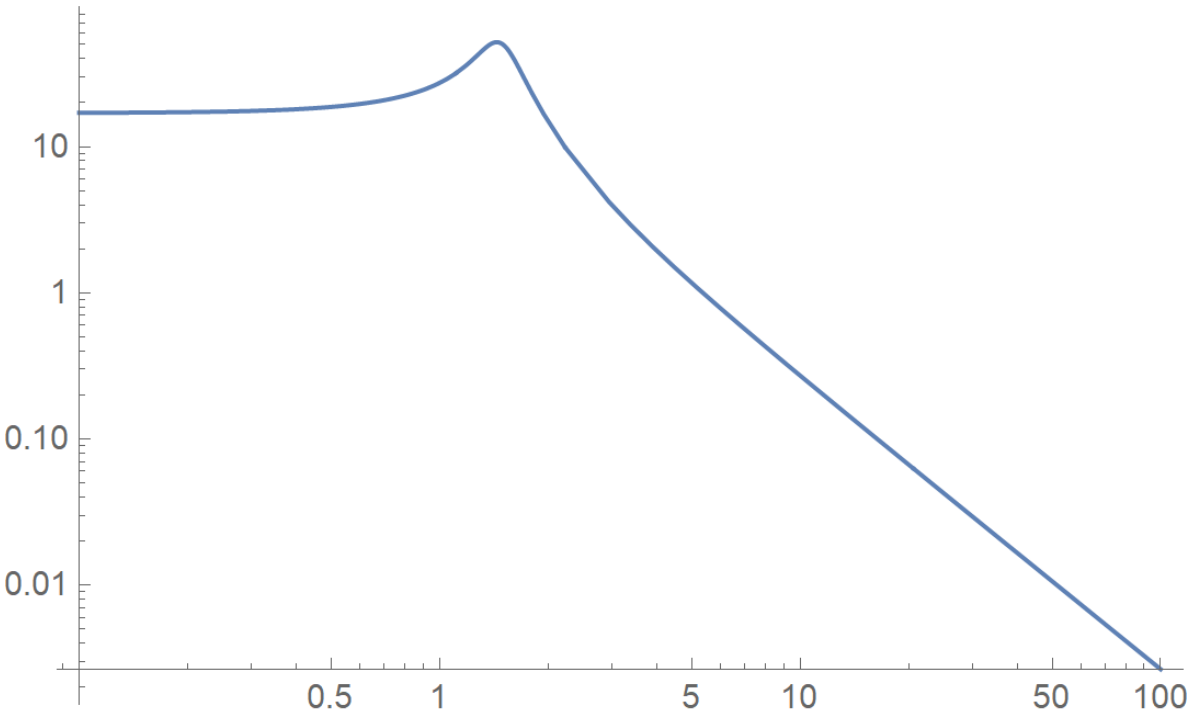
Out[380]:



Actuation TF: Yaw -> Yaw

In [381]:
```
LogLogPlot[Abs[actZetaZetaFxy], {f,0.1,100}]
```

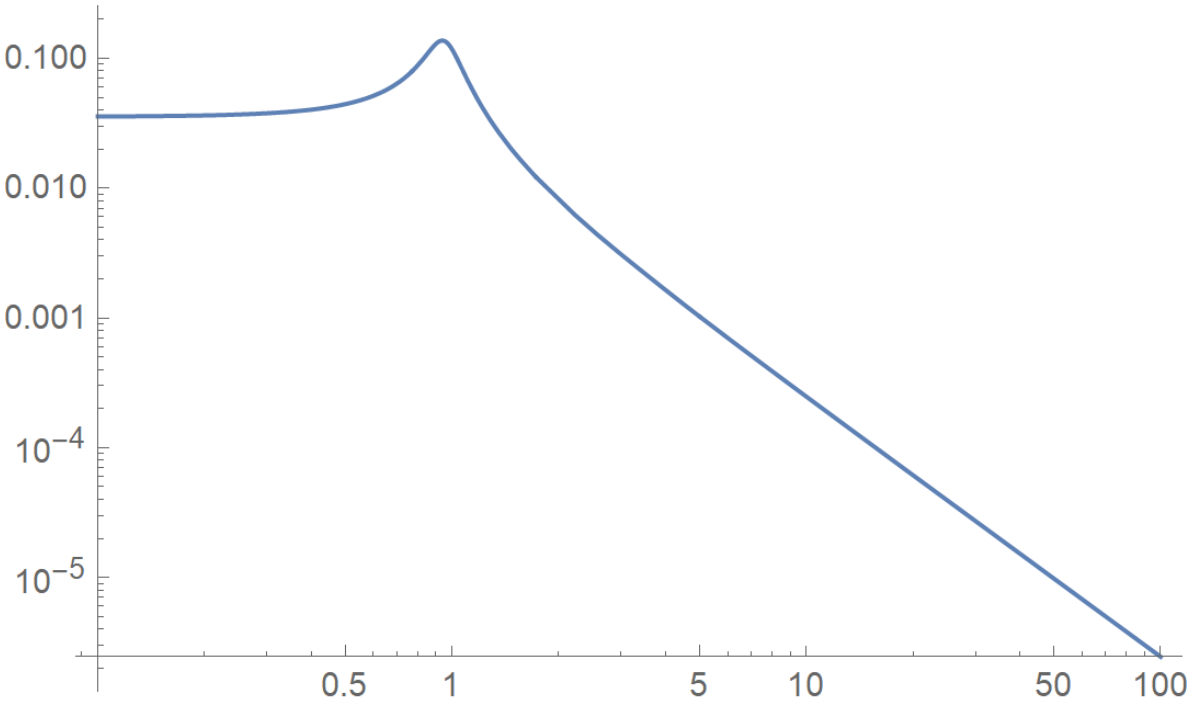
Out[381]:



Actuation TF: L -> L

In [382]:
```
LogLogPlot[Abs[actXXFxy], {f,0.1,100}]
```

Out[382]:



Save data

In [383]:
```
fisoXXF[f_]=Abs[isoXXFzx];
fisoYYF[f_]=Abs[isoYYFyz];
fisoZZF[f_]=Abs[isoZZFzx];
fisoXEtaF[f_]=Abs[isoXEtaFzx];
fisoZEtaF[f_]=Abs[isoZEtaFzx];
```

```
writeIgor["Type-C_TM_Isolation.dat",
{"#freq","XX","YY","ZZ","X->Pitch"},
{freq, fisoXXF[freq],fisoYYF[freq],fisoZZF[freq],fisoXEtaF[freq]}
  ];


factXXF[f_]=Abs[actXXFzx];
factXXFAng[f_]=Arg[actXXFzx]/Pi*180;

factEtaEtaF[f_]=Abs[actEtaEtaFzx];
factEtaEtaFAng[f_]=Arg[actEtaEtaFzx]/Pi*180;
factZetaZetaF[f_]=Abs[actZetaZetaFxy];
factZetaZetaFAng[f_]=Arg[actZetaZetaFxy]/Pi*180;

writeIgor["Type-C_TM_Act.dat",
{"#freq","Abs:Len","Phase:Len","Abs:Pit","Phase:Pit","Abs:Yaw","Phase:Yaw"},
{freq, factXXF[freq],factXXFAng[freq],factEtaEtaF[freq],factEtaEtaFAng[freq],factZetaZ
  ];
```

Pick-off Type

In this section, we will compute the TFs of PO type suspensions.

Load Geometry

In [396]:

```
<<"PO_Type_geometry.m"
```

Find the equilibrium points

In [397]:

```
findEquivXY;
findEquivYZ;
findEquivZX;
```

```
1: {0, 18.3526, 0, 0, -21.3772, 0}
2: {0, -0.00862611, 0, 0, 0.00862611, 0}
                    -9                  -9
3: {0, -1.37646 10  , 0, 0, 1.37648 10  , 0}
```

Set the EQOM matrices

In [400]:

```
setMat
```

Eigen frequencies

In [401]:

```
Map[MatrixForm,{eigenFreqxy,eigenFreqyz,eigenFreqzx}]
```

Out[401]:

$$\left\{\begin{pmatrix} 0.96447 & 4.87219 \\ 1.28209 & 0.403662 \\ 1.55938 & 1.17266 \\ 1.67453 & 0.322864 \\ 2.17403 & 2.21149 \\ 3.99642 & 1.19615 \\ 4.02243 & 2.8971 \\ 6.12656 & 23.9612 \\ 11.8006 & 11.9933 \end{pmatrix}, \begin{pmatrix} 0.957211 & 5.00847 \\ 1.664 & 0.324476 \\ 4.01987 & 2.84166 \\ 6.54159 & 15.174 \\ 6.92227 & 1.42326 \\ 7.18528 & 4.14944 \\ 13.6283 & 2.49809 \\ 32.3155 & 21.8628 \\ 37.3849 & 299.515 \end{pmatrix}, \begin{pmatrix} 0.964389 & 4.87216 \\ 1.67453 & 0.322863 \\ 2.00268 & 2.51254 \\ 4.02243 & 2.8971 \\ 6.54159 & 15.174 \\ 6.92227 & 1.42326 \\ 10.0261 & 11.0494 \\ 11.0922 & 34.0537 \\ 32.3155 & 21.8628 \end{pmatrix}\right\}$$

Calculate vibration isolation ratio

In [402]:

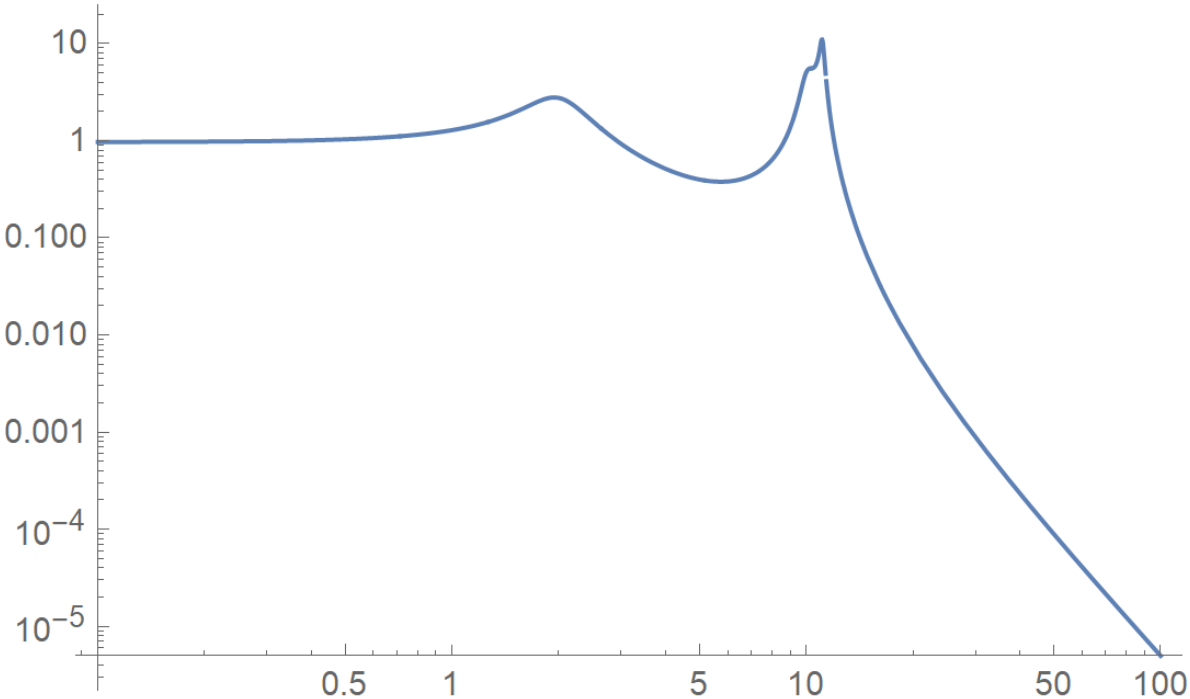
```
calcVIRxy;
calcVIRyz;
calcVIRzx;
```

```
iso{X|Y|Zeta}{X|Y|Zeta}{IFM}xy have been set.
iso{Y|Z|Xi}{Y|Z|Xi}{IFM}yz have been set.
iso{Z|X|Eta}{Z|X|Eeta}{IFM}zx have been set.
```

In [405]:

```
LogLogPlot[Abs[isoEtaEtaFzx], {f,0.1,100}]
```

Out[405]:



Calculate actuation transfer functions

In [406]:

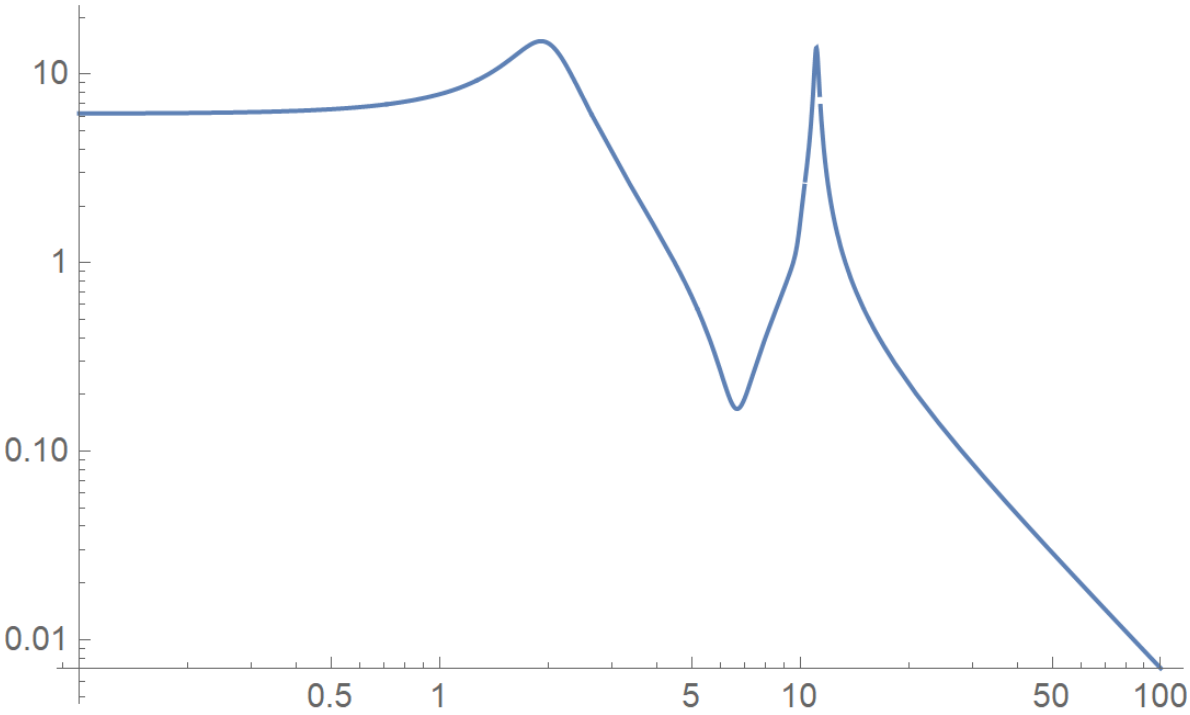
```
calcActxy;
calcActyz;
calcActzx;
```

```
act{X|Y|Zeta}{X|Y|Zeta}{IFM}xy have been set.
act{Y|Z|Xi}{Y|Z|Xi}{IFM}yz have been set.
act{Z|X|Eta}{Z|X|Eeta}{IFM}zx have been set.
```

Actuation TF: Pitch -> Pitch

In [409]:
```
LogLogPlot[Abs[actEtaEtaFzx], {f,0.1,100}]
```

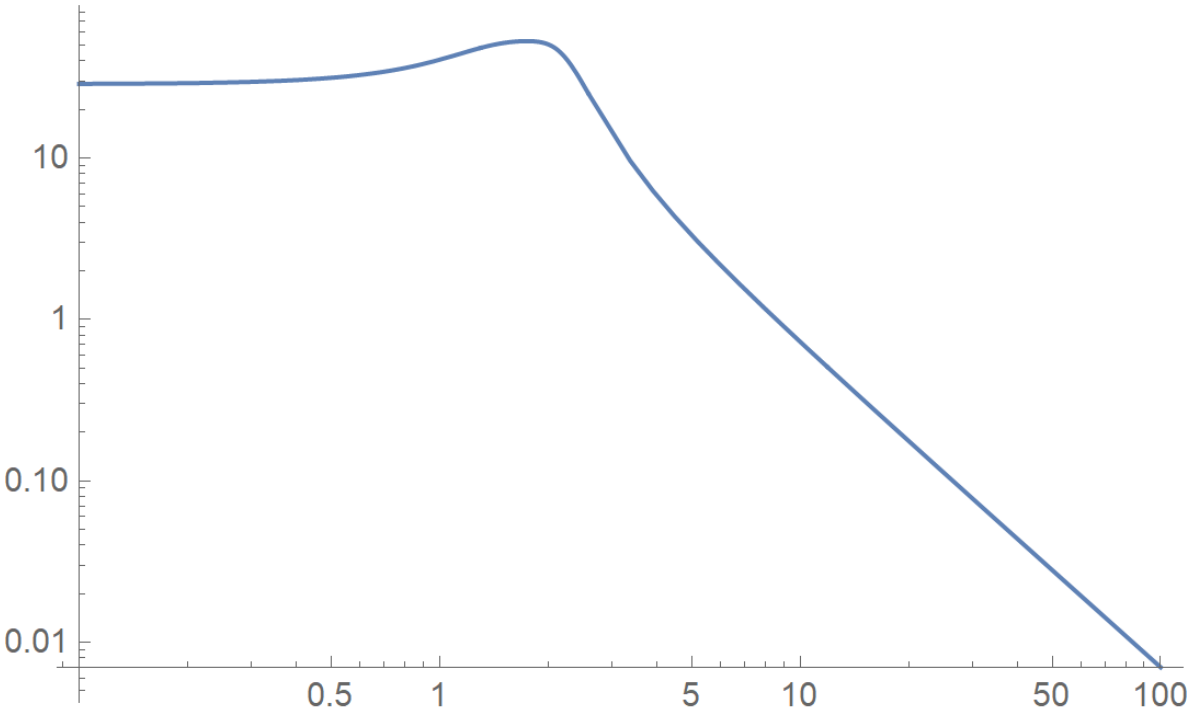
Out[409]:



Actuation TF: Yaw -> Yaw

In [410]:
```
LogLogPlot[Abs[actZetaZetaFxy], {f,0.1,100}]
```

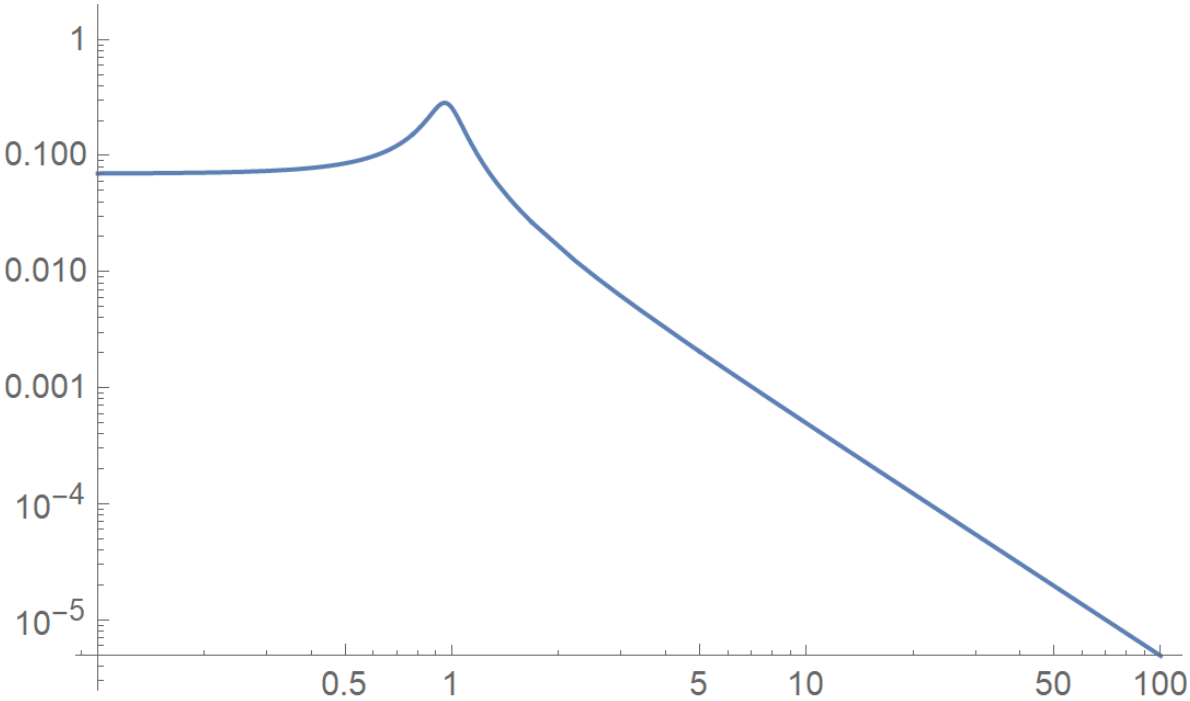
Out[410]:



Actuation TF: L -> L

In [411]:
```
LogLogPlot[Abs[actXXFxy], {f,0.1,100}]
```

Out[411]:



In [426]:

```
vecActXxy
```

Out[426]: {0, 0, 0, 1, 0, 0, 0, 0, 0}

Save data

In [412]:

```
fisoXXF[f_]=Abs[isoXXFzx];
fisoYYF[f_]=Abs[isoYYFyz];
fisoZZF[f_]=Abs[isoZZFzx];
fisoXEtaF[f_]=Abs[isoXEtaFzx];

writeIgor["Type-C_PO_Isolation.dat",
{"#freq","XX","YY","ZZ","X->Pitch"},
{freq, fisoXXF[freq],fisoYYF[freq],fisoZZF[freq],fisoXEtaF[freq]}
  ];

factXXF[f_]=Abs[actXXFzx];
factXXFAng[f_]=Arg[actXXFzx]/Pi*180;

factEtaEtaF[f_]=Abs[actEtaEtaFzx];
factEtaEtaFAng[f_]=Arg[actEtaEtaFzx]/Pi*180;
factZetaZetaF[f_]=Abs[actZetaZetaFxy];
factZetaZetaFAng[f_]=Arg[actZetaZetaFxy]/Pi*180;

writeIgor["Type-C_PO_Act.dat",
{"#freq","Abs:Len","Phase:Len","Abs:Pit","Phase:Pit","Abs:Yaw","Phase:Yaw"},
{freq, factXXF[freq],factXXFAng[freq],factEtaEtaF[freq],factEtaEtaFAng[freq],factZetaZ
  ];
```

In [ ]: