Suspension Model Usage Notes

07/04/11 T. Sekiguchi

Installation

1. Make sure that file “visUtil.nb” and the matching .m file “visUtil.m” are in the parent directories.
2. To export state-space matrices, you need to create subdirectory called “matlab”. Be sure that such a subdirectory exists in your directory.

Contents

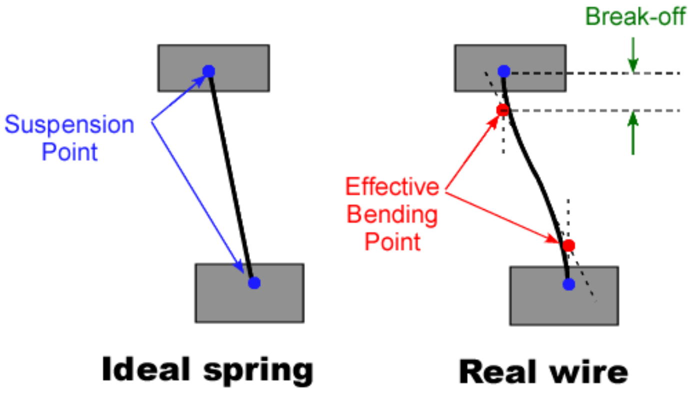
1. A notebook “toy110426.nb” contains definitions of a suspension model, setup and some codes to output 3D graphics, plot transfer functions, and etc.
2. The notebook is divided into the following sections:
   Description: Description about the model, version history
   Preliminaries: Some setup for the calculation (importing packages, defining a unit system)
   Constant Values: Define mass and MOI of bodies, length of wires, geometric parameters
   Variables: Pick up all the variables that define the model
   Energy Calculation: Calculate potential, kinetic and damping energies
   Find Equilibrium Point: Find the local minimum (working point) of the system
   Equation of Motions: Differentiate energy terms, create mass, stiffness and damping matrices
   Eigen Mode: Calculate the eigenmodes and eigenfrequencies of the system, visualize them
   Transfer Function Plot: Plot various transfer functions of the system
   State-Space Model: Create state-space matrices and export them in the MATLAB style

* Preliminaries

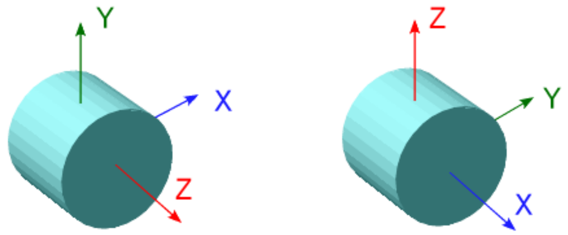
1. Be sure that there is a code to import the utility package in the parent directory (<<`"../visUtil.m"`).
2. MKS unit system is set as a default. If you want to use another unit system, edit the **Unite System** subsection.

* Constant Values

1. This section defines mass, MOI, shape of bodies, material, natural length, diameter of wires, spring constants, working direction of springs (GAS filters), damping coefficient of magnet dampers, geometric parameters defining orientation and configuration of wires.
2. In the suspension model, wires are considered to be ideal mass-less springs. In this condition, the bending elasticity of a wire is ignored. To take this into account, the wire length and the wire break-off are compensated in the subsubsection **Effective Bending Point Compensation**. For more details, find the document T010171-00-D (LIGO internal document).



3. The suspension points of a wire are defined in the local coordinate systems of the attached bodies. Local coordinates for all objects originate at their COM. Be careful that in my model, the following reference is used:
   X: Transversal, **Y**: Vertical, **Z**: Longitudinal (Beam axis)
   Pitch: Rotation around X, **Yaw**: Rotation around Y, **Roll**: Rotation around Z.
   This is the same as used in Virgo, but is different from the reference normally used by Japanese (**X**: Longitudinal, **Z**: Vertical).



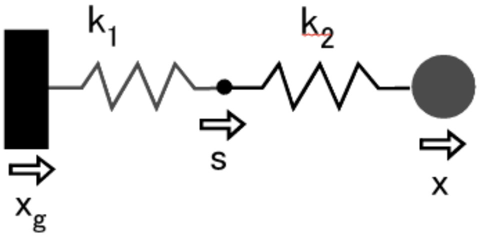
Virgo's Coordinate Japanese Coordinate

4. In the **Body Shape** subsection, the appearance of each body is defined. This is not important for the calculation of eigenmodes or transfer functions, but is used for the 3-D graphics to visualize eigenmodes of the system. To define the body shape, choose the type of the shape from zCylinder, Cuboid, OpenCuboid, yCylinder, TruncatedCone, DoubleCylinder, Doughnut, Circle, and then define the numerical values of the size.

* Variables

1. There are three types of variables: variables, parameters and floats. The lists of variables are called `allvars`, `allparams` and `allfloats`, respectively.
2. The `allvars` should be a list of the position and angle variables for all elements of the system that have mass/MOI associated with them. The `allparams` should be a list of positions and angles describing the state of the ground. The `allfloats` should be a list of positions and angles of connections where one elastic element is connected directly to another with no mass element between.

   e.g.) Oscillator with two mass-less springs



```
allvars={x}; allparams={xg}; allfloats={s};
```

* Energy Calculation

1. To calculate the potential, kinetic and damping energies of the system, lists of bodies, wires, springs and dampers are to be defined. To know how to define them, read the usage of the `bodylist`, `wirelist`, `springlist` and `damperlist`.
2. Potential energy is divided into four parts: wire stretching potential, wire torsional potential, spring potential, and gravity potential (+ other potential energy if you want). These energy terms can be obtained by functions such as `makewirepot` and etc.

3. The wire stretching potential reflects increase of the straight line distance between the end points.

$$P_{\mathrm{wireS}} = \frac{1}{2} k_{\mathrm{wire}} (l(\mathbf{x}) - l_0)^2$$

The user is responsible for supplying value for the elastic constant (k_{wire}) and the natural length (L_0) of the wire at the `wirelist`.

4. The function `b2s` provides the system coordinates $\{x_s, y_s, z_s\}$ for a point at body coordinates $\{x_b, y_b, z_b\}$ on the object defined by {x,y,z,pitch,yaw,roll}. As the suspension points of the wires are defined by the local coordinates of the attached bodies, it is necessary to convert body coordinates to the system coordinates.

$$\begin{pmatrix} x_s \\ y_s \\ z_s \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} \cos(roll) & -\sin(roll) & 0 \\ \sin(roll) & \cos(roll) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos(yaw) & 0 & \sin(yaw) \\ 0 & 1 & 0 \\ -\sin(yaw) & 0 & \cos(yaw) \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(pitch) & -\sin(pitch) \\ 0 & \sin(pitch) & \cos(pitch) \end{pmatrix} \begin{pmatrix} x_b \\ y_b \\ z_b \end{pmatrix}$$

5. The wire torsional potential is calculated by the following equation:

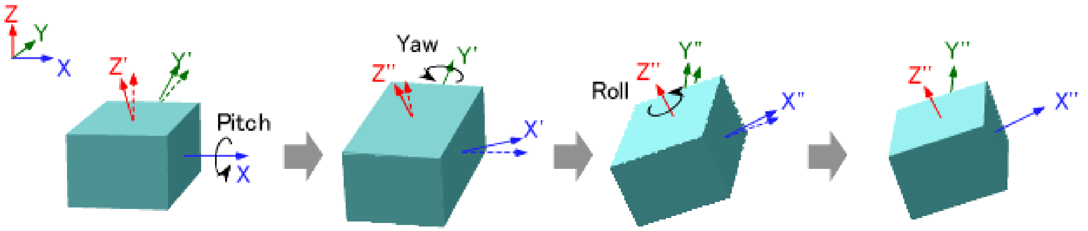
$$P_{\mathrm{wireT}} = \frac{1}{2} \frac{G I_{\mathrm{p}}}{l_0} \Delta\theta_{\mathrm{T}}^{\ 2} ,$$

where I_p is the polar moment of area, G is the shear modulus and θ_T is the twist of the wire.

6. A spring joins two points on different objects and apply restoring forces on them in 6 DOFs according to a tensor of elastic constants and a vector of pre-load forces:

$$P_{\mathrm{spring}} = \frac{1}{2} (\Delta x, \Delta y, \Delta z, \Delta Pitch, \Delta Yaw, \Delta Roll) \mathbf{K}_{\mathrm{spring}} \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \\ \Delta Pitch \\ \Delta Yaw \\ \Delta Roll \end{pmatrix} + \mathbf{f}_{\mathrm{preload}} \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \\ \Delta Pitch \\ \Delta Yaw \\ \Delta Roll \end{pmatrix}$$

7. The angular coordinates of bodies {pitch, yaw, roll} are defined as follows:



From these incremental pitch/yaw/roll representation, a differential angular displacement in the body coordinate is calculated by the following equation:

$$\begin{pmatrix} \Delta Pitch^B \\ \Delta Yaw^B \\ \Delta Roll^B \end{pmatrix} = \begin{pmatrix} \cos(Yaw)\cos(Pitch) & 0 & 0 \\ \cos(Yaw)\sin(Pitch) & \cos(Pitch) & 0 \\ -\sin(Yaw) & -\sin(Pitch) & 1 \end{pmatrix} \begin{pmatrix} \Delta Pitch \\ \Delta Yaw \\ \Delta Roll \end{pmatrix} \equiv \mathbf{\Omega}(Pitch, Yaw, Roll) \begin{pmatrix} \Delta Pitch \\ \Delta Yaw \\ \Delta Roll \end{pmatrix}$$

The kinetic energy about the angular motion of the body is calculated as follows:

$$K_{\mathrm{angle}} = \begin{pmatrix} \partial_t Pitch & \partial_t Yaw & \partial_t Roll \end{pmatrix} \mathbf{\Omega}(pitch, yaw, roll)^T \ \mathbf{I} \ \mathbf{\Omega}(pitch, yaw, roll) \begin{pmatrix} \partial_t Pitch \\ \partial_t Yaw \\ \partial_t Roll \end{pmatrix}$$

where I is the moment of inertia tensor.

* Find Equilibrium Point

1. Use `FindMinimum` function to find an equilibrium point of the system. Keep in mind that `FindMinimum` is a function to search for a "local" minimum point in a given potential, so that you need to choose appropriate starting points (initial values) in the **Variables** section.
2. The user can choose the calculation method and set the maximum number of iteration. To see how to define them, find the help of the `FindMinimum` function.

* Equations of Motion

1. Differentiate the potential energy of the system with respect to pairs of coordinates at equilibrium to create the stiffness matrix `matKxx`. In a similar way, differentiate the kinetic energy and damping energy with respect to coordinate velocities to create the mass matrix `matMxx` and damping matrix `matGxx`.

$$K_{ij}=\left.\frac{\partial E_{\text{Potential}}}{\partial x_i \partial x_j}\right|_{x=x_{(eq)}},\quad M_{ij}=\left.\frac{\partial E_{\text{Kinetic}}}{\partial \dot{x}_i \partial \dot{x}_j}\right|_{x=x_{(eq)}},G_{ij}=\left.\frac{\partial E_{\text{Damping}}}{\partial \dot{x}_i \partial \dot{x}_j}\right|_{x=x_{(eq)}}$$

2. The matrix `matKvv` is a differentiation of the potential energy with respect to pairs of variable coordinates. The matrix `matKpv` is a differentiation of the potential energy with respect to pairs of parameter coordinates and variable coordinates. In a similar way, `matKfv`, `matKfp`, `matKff`, `matMvv`, `matMpv` and `matGvv` are defined (`f` means float coordinates).
3. From these matrices, equations of motion of the system can be written as follows:

$$\left(s^2\mathbf{M}_{\text{vv}}+s\mathbf{G}_{\text{vv}}+\mathbf{K}_{\text{vv}}\right)\mathbf{x}+\left(s^2\mathbf{M}_{\text{pv}}+s\mathbf{G}_{\text{pv}}+\mathbf{K}_{\text{pv}}\right)\mathbf{x}_{\text{ground}}+\mathbf{K}_{\text{fv}}\mathbf{x}_{\text{float}}=0$$

As the float coordinates belong to massless bodies, the equations of motion about these coordinates are

$$\mathbf{K}_{\text{fv}}{}^T\mathbf{x}+\mathbf{K}_{\text{fp}}{}^T\mathbf{x}_{\text{ground}}+\mathbf{K}_{\text{ff}}\mathbf{x}_{\text{float}}=0$$

From these equations, the float coordinates are eliminated and you obtain the following "effective" spring constant matrices:

$$\left(s^2\mathbf{M}_{\text{vv}}+s\mathbf{G}_{\text{vv}}+\mathbf{K}_{\text{vv}}{}^{\text{(eff)}}\right)\mathbf{x}+\left(s^2\mathbf{M}_{\text{pv}}+s\mathbf{G}_{\text{pv}}+\mathbf{K}_{\text{pv}}{}^{\text{(eff)}}\right)\mathbf{x}_{\text{ground}}=0$$

$$\mathbf{K}_{\text{vv}}{}^{\text{(eff)}}=\mathbf{K}_{\text{vv}}-\mathbf{K}_{\text{fv}}\mathbf{K}_{\text{ff}}{}^{-1}\mathbf{K}_{\text{fv}}{}^T$$

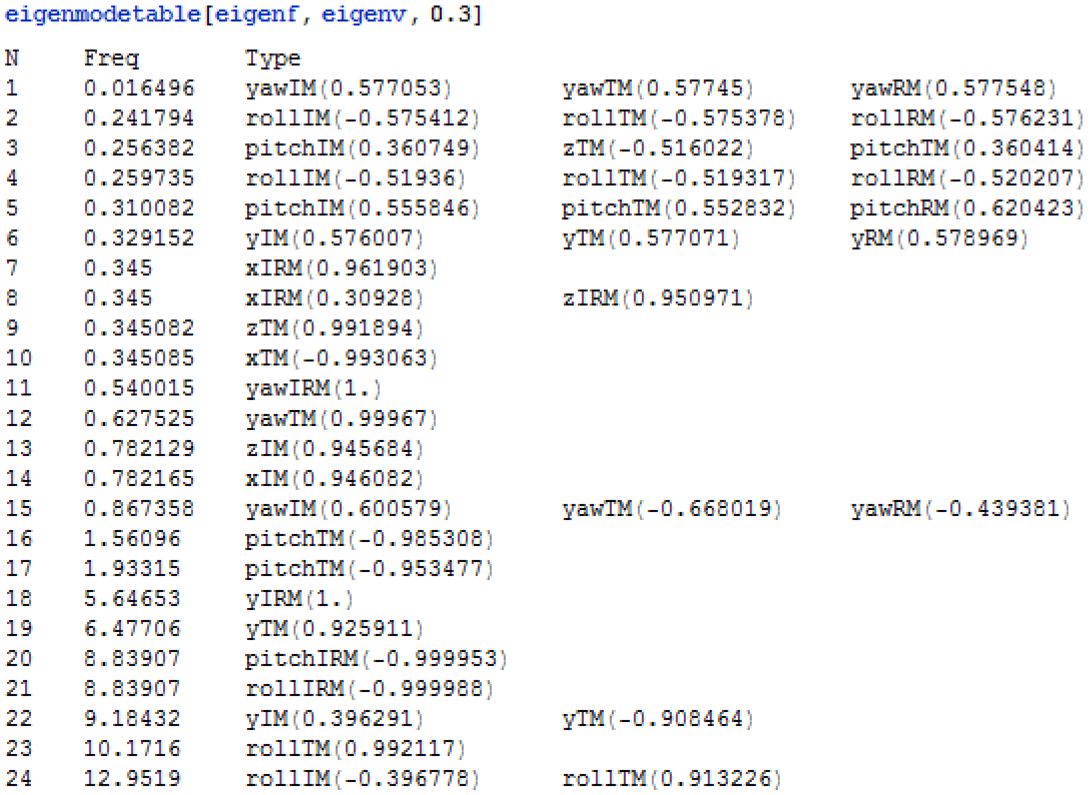
$$\mathbf{K}_{\text{pv}}{}^{\text{(eff)}}=\mathbf{K}_{\text{pv}}-\mathbf{K}_{\text{pv}}\mathbf{K}_{\text{ff}}{}^{-1}\mathbf{K}_{\text{fp}}{}^T$$

* Eigen Mode

1. In this section, we diagonalize the stiffness and mass matrices to obtain the eigenfrequencies f_i and eigenvectors $\mathbf{e}_i$ of the system. Eigenvalues and eigenvectors of the matrix can be obtained by the `Eigensystem` function.

$$(\mathbf{M}_{\text{vv}}{}^{-1}\mathbf{K}_{\text{vv}}{}^{\text{(eff)}})\mathbf{e}_i=(2\pi f_i)^2\mathbf{e}_i$$

2. The `eigentable` fuction generates a table of the eigenmodes like following:



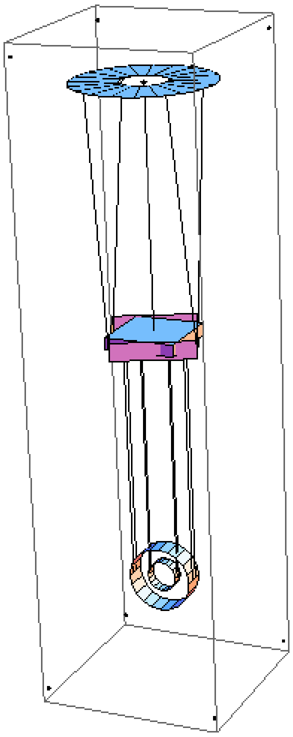
```
eigenmodetable[eigenf, eigenv, 0.3]

N    Freq       Type
1    0.016496   yawIM(0.577053)       yawTM(0.57745)       yawRM(0.577548)
2    0.241794   rollIM(-0.575412)     rollTM(-0.575378)    rollRM(-0.576231)
3    0.256382   pitchIM(0.360749)     zTM(-0.516022)       pitchTM(0.360414)
4    0.259735   rollIM(-0.51936)      rollTM(-0.519317)    rollRM(-0.520207)
5    0.310082   pitchIM(0.555846)     pitchTM(0.552832)    pitchRM(0.620423)
6    0.329152   yIM(0.576007)         yTM(0.577071)        yRM(0.578969)
7    0.345      xIRM(0.961903)
8    0.345      xIRM(0.30928)         zIRM(0.950971)
9    0.345082   zTM(0.991894)
10   0.345085   xTM(-0.993063)
11   0.540015   yawIRM(1.)
12   0.627525   yawTM(0.99967)
13   0.782129   zIM(0.945684)
14   0.782165   xIM(0.946082)
15   0.867358   yawIM(0.600579)       yawTM(-0.668019)     yawRM(-0.439381)
16   1.56096    pitchTM(-0.985308)
17   1.93315    pitchTM(-0.953477)
18   5.64653    yIRM(1.)
19   6.47706    yTM(0.925911)
20   8.83907    pitchIRM(-0.999953)
21   8.83907    rollIRM(-0.999988)
22   9.18432    yIM(0.396291)         yTM(-0.908464)
23   10.1716    rollTM(0.992117)
24   12.9519    rollIM(-0.396778)     rollTM(0.913226)
```

The table is sorted in the increasing numbers of eigenfrequencies. In the “Type” column, main variables that represent the eigenmode and their amplitudes are marked down.

3. The `eigenplot` function shows 3D graphics of the eigenmode shape. Put a number that is marked in the above list (`eigentable`) and amplitude, then the user can see the shape of the eigenmode in 3D graphics like as follows:

```
eigenplot[eigenv, 0.4, 11]
```



* Transfer Function Plot

1. There are three types of transfer functions that the user can plot. The `tfplot` calculates transfer functions from the ground displacement, to the body's displacement. The `tfplotf` calculates transfer functions from force/torque exerted to the input variables, to the output variables' displacement. The `tfplota` calculates transfer functions from the actuator force to the body's displacement. To use the `tfplota`, the user need to define the `actuatorlist`.
2. The transfer functions from the ground displacement, to the body's displacement are calculated as follows:

$$\mathbf{H}_{\text{ground}} = \left(s^2\mathbf{M}_{\text{vv}} + s\mathbf{G}_{\text{vv}} + \mathbf{K}_{\text{vv}}^{(\text{eff})}\right)^{-1}\left(s^2\mathbf{M}_{\text{pv}} + s\mathbf{G}_{\text{pv}} + \mathbf{K}_{\text{pv}}^{(\text{eff})}\right)$$

$$\mathbf{x} = \mathbf{H}_{\text{ground}}\mathbf{x}_{\text{ground}}$$

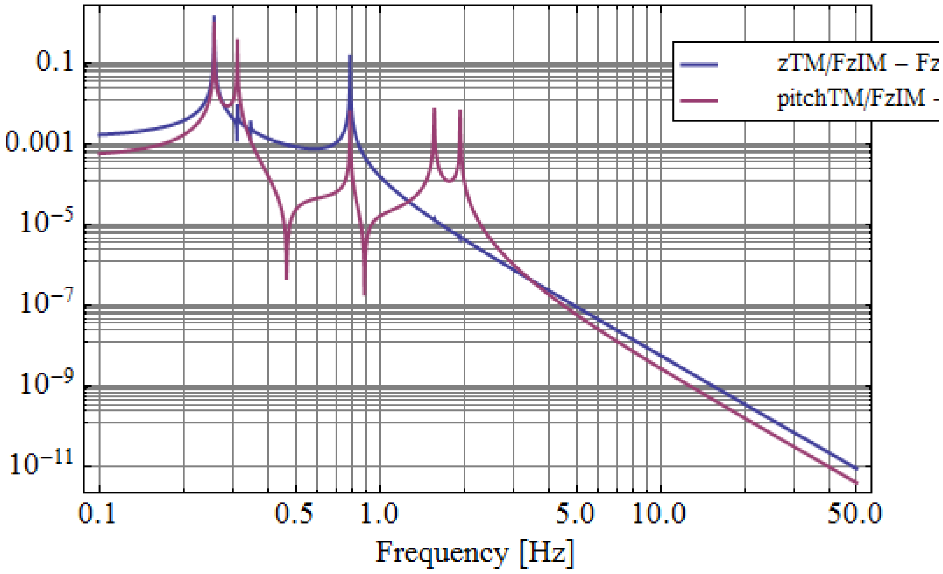
   The transfer functions from force exerted to the system, to the body's displacement are calculated as follows:

$$\mathbf{H}_{\text{force}} = \left(s^2\mathbf{M}_{\text{vv}} + s\mathbf{G}_{\text{vv}} + \mathbf{K}_{\text{vv}}^{(\text{eff})}\right)^{-1}$$

$$\mathbf{x} = \mathbf{H}_{\text{force}}\mathbf{f}$$

3. The followings are usage examples of transfer function plots. When using the `tfplotf` and `tfplota`, the input parameters can be a linear combination of the variables.

```
tfplotf[{zIM - zIRM}, {zTM, pitchTM}, freq]
```



* State-Space Model

1. For time domain simulation, the state-space formalism is more convenient. We assume a set of input and output parameters as follows:

$$\text{Input} = \begin{pmatrix} \mathbf{x}_{\text{ground}} \\ \mathbf{f} \end{pmatrix}, \qquad \text{Output} = \left(\mathbf{x} - \mathbf{x}_{\text{eq}}\right)$$

   From the matrices calculated in the **Equations of Motion** section, the state-space matrices are calculated as follows:

$$\mathbf{A}=\begin{pmatrix}\mathbf{0} & \mathbf{1}\\ -\mathbf{M}_{\mathrm{vv}}{}^{-1}\mathbf{K}_{\mathrm{vv}}{}^{(\mathrm{eff})} & -\mathbf{M}_{\mathrm{vv}}{}^{-1}\mathbf{G}_{\mathrm{vv}}\end{pmatrix} \qquad \mathbf{B}=\begin{pmatrix}\mathbf{0} & \mathbf{0}\\ -\mathbf{M}_{\mathrm{vv}}{}^{-1}\mathbf{K}_{\mathrm{pv}}{}^{(\mathrm{eff})} & \mathbf{M}_{\mathrm{vv}}{}^{-1}\end{pmatrix}$$

$$\mathbf{C}=\begin{pmatrix}\mathbf{1} & \mathbf{0}\end{pmatrix} \qquad \mathbf{D}=\begin{pmatrix}\mathbf{0} & \mathbf{0}\end{pmatrix}$$

2. Structure damping is approximated by viscous damping, since all the elements of the state-space matrices must be real and not be frequency dependent. When the imaginary part of the stiffness matrix is taken into account, the state-space A matrix is written as:

$$\mathbf{A}=\begin{pmatrix}\mathbf{0} & \mathbf{1}\\ \mathrm{Re}[-\mathbf{M}_{\mathrm{vv}}{}^{-1}\mathbf{K}_{\mathrm{vv}}{}^{(\mathrm{eff})}] & -\mathbf{M}_{\mathrm{vv}}{}^{-1}\mathbf{G}_{\mathrm{vv}}+\dfrac{\mathrm{Im}[-\mathbf{M}_{\mathrm{vv}}{}^{-1}\mathbf{K}_{\mathrm{vv}}{}^{(\mathrm{eff})}]}{2\pi f}\end{pmatrix}$$

   As the state-space matrices should not be frequency dependent, the frequency dependent part is approximated by a value with a single frequency:

$$\mathbf{A}=\begin{pmatrix}\mathbf{0} & \mathbf{1}\\ \mathrm{Re}[-\mathbf{M}_{\mathrm{vv}}{}^{-1}\mathbf{K}_{\mathrm{vv}}{}^{(\mathrm{eff})}] & -\mathbf{M}_{\mathrm{vv}}{}^{-1}\mathbf{G}_{\mathrm{vv}}+\dfrac{\mathrm{Im}[-\mathbf{M}_{\mathrm{vv}}{}^{-1}\mathbf{K}_{\mathrm{vv}}{}^{(\mathrm{eff})}]}{2\pi f_{\mathrm{damping}}}\end{pmatrix}$$

3. The `matlabexport` and `matlabexportappend` functions export the matrices in the format used in MATLAB.