vfo Documentation

Release 0.0.9

Anurag Upadhyay

May 22, 2023

CONTENTS

Important: Current Version is 0.0.11

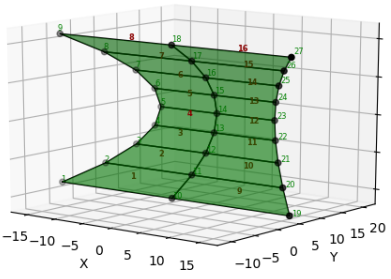
ANNOUNCEMENT: The plotting backend for vfo is being switched to a vtk based plotting tool PyVista starting vfo 0.0.6 for smooth interaction in large OpenSees model. Some of the commands are changing and the older commands will not work with the new updated version. Please read this document carefully.

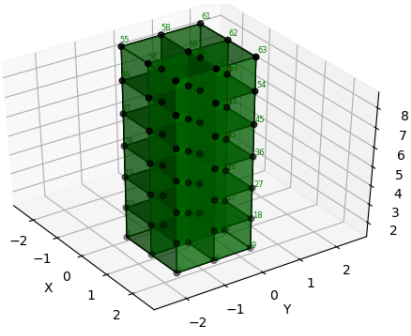
vfo (Visualization for OpenSees) is a Python package to make your life better by helping you visualize your OpenSees models, Python or Tcl. It utilizes PyVista library to plot 2D and 3D models in a dedicated interactive window. You can use click-and-hold to change the view angle and zoom the plot. The model image can be saved with the desired orientation directly from the interactive plot window.

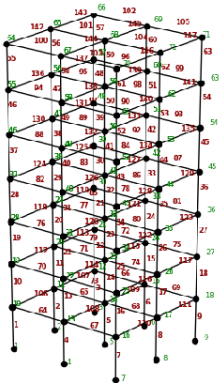
Animation: To save the animation movie as .mp4 file, FFmpeg codecs are required.

Following elements are supported:

- 2D and 3D Beam-Column Elements
- 2D and 3D Quad Elements
- 2D and 3D Tri Elements
- 8 Node Brick Elements
- 4 Node Tetrahedron Elements







Installation

```
python -m pip install vfo

python -m pip install --user vfo
```

To upgrade the package installation

```
python -m pip install --upgrade vfo

python -m pip install --user --upgrade vfo
```

The following two commands are needed to visualize the model, as shown below:

```
# import vfo rendering module
import vfo.vfo as vfo

# render the model after defining all the nodes and elements
vfo.plot_model()

# plot mode shape
vfo.plot_modeshape(modenumber=3)
```

Following are commands and development guide related to model visualization:

1. *Installation*
2. *Create Output Database*
3. *saveFiberData2D command*
4. *plot_model command*
5. *plot_modeshape command*
6. *plot_deformedshape command*
7. *animate_deformedshape command*
8. *plot_fiberResponse2D command*
9. *animate_fiberResponse2D command*
10. *Plotting OpenSees Tcl Output*

CHAPTER

ONE

INSTALLATION

- To install

```
python -m pip install vfo

python -m pip install --user vfo
```

- To upgrade

```
python -m pip install --upgrade vfo

python -m pip install --user --upgrade vfo
```

- To import

```
import vfo.vfo as vfo
```

CHAPTER

TWO

CREATE OUTPUT DATABASE

`vfo.createODB(`*model="ModelName"*, *<loadcase="LoadCaseName">*, *<Nmodes=0>*, *<deltaT=0.0>*`)`

This command creates an Output Database for the active model with an option to save a specific load case output. The command **must** be called while the model is built, but before the main analysis is run. An output database with name ModelName_ODB is created in the current directory with node and element information in it. See the example below.

Input arguments are as follows,

`model` (str)	Name of the model the user wants to save database with.
`loadcase` (str)	Name of the subfolder to save load case output data.(Optional)
`Nmodes` (int)	Number of modes to be saved for visualization.(Optional)
`deltaT` (float)	Timesteps at which output to be saved. By default, the output is saved at each analysis step. (optional)

Here is a simple example:

```
vfo.createODB(model="TwoSpan_Bridge", Nmodes=3)
```

The above command will create,

- a folder named **TwoSpan_Bridge_ODB** containing the information of the nodes and elements to visualize the structure in future without using a OpenSeesPy model file.
- a sub-folder named **ModeShapes** containing information on modeshapes and modal periods.
- no output from any loadcase will be saved saved in this case even if the analysis is run in the script since there is no argument for `loadcase` argument is provided.

Here is another example command to be used when recording data from a load case.

```
vfo.createODB(model="TwoSpan_Bridge", loadcase="Dynamic_GM1", Nmodes=3, deltaT=0.5)
```

The above command should be used right before running a load case analysis in the OpenSeesPy script and will create,

- a folder named **TwoSpan_Bridge_ODB** containing the information on the nodes and elements to visualize the structure in future without using a OpenSeesPy model file.
- a sub-folder named **Dynamic_GM1** containing the information on node displacement data to plot deformed shape.
- a sub-folder named **ModeShapes** containing information on modeshapes and modal periods.
- the node displacement data will be saved at closest time-steps at each 0.05 sec interval.

CHAPTER

THREE

SAVEFIBERDATA2D COMMAND

`vfo.`**`saveFiberData2D`**(*ModelName*, *LoadCaseName*, *eleNumber*, *sectionNumber*, *deltaT=0.0*)

This command saves the fiber output of the specified section in a non-linear beam-column element to a sub-folder named "LoadCaseName" inside the output database folder "ModelName_ODB". This output data can be visualized using `plot_fiberResponse2D()` and `animate_fiberResponse2D()` commands.

`ModelName` (str)	Name of the model the user wants to save database with.
`LoadCaseName` (str)	Name of the subfolder to save load case output data.
`eleNumber` (int)	Tag of the element with a fiber section assigned.
`sectionNumber` (float)	Tag of the section where the fiber response is to be recorded.
`deltaT` (list)	Timesteps at which output to be saved. Default is 0.0. (optional)

Here is a simple example:

```
vfo.saveFiberData2D("TwoSpan_Bridge", "Pushover", 101, 2)
```

The above command will:

- create a folder named **TwoSpan_Bridge_ODB** and a sub-folder named **Pushover** if they are not already there.
- save fiber response of the section with a tag 2 of the element with a tag 101 in the Pushover folder.

CHAPTER

FOUR

PLOT_MODEL COMMAND

vfo.**plot_model**(*model='none'*, *show_nodes='no'*, *show_nodetags='no'*, *show_eletags='no'*, *font_size=10*, *setview='3D'*, *elementgroups=None*, *line_width=1*, *filename=None*)

Once the model is built, it can be visualized using this command. By default Node and element tags are not displayed. No analysis is required in order to visualize the model.

To visualize an OpenSees model from an existing database (using `createODB()` command), the optional argument model="ModelName" should be used. The command will read the model data from folder **ModelName_ODB**.

`model` (str)	name of the model output database as used in createODB() function. If no name is provided, the function tries to get data from the active model. Default is "none". (optional)
`show_nodes` (str)	Renders nodes as spheres if "yes". Default is "no". (optional)
`show_nodetags` (str)	Displays node tags if "yes". Default is "no". (optional)
`show_eletags` (str)	Displays element tags if "yes". Default is "no". (optional)
`font_size` (int)	Size of tag font. Default is 10. (optional)
`setview` (str)	sets the camera view to predefined angles. Valid entries are "xy","yz","xz","3D", or a list with [x,y,z] unit vector. Default is "3D". (optional)
`elementgroups` (list)	List of lists of elements of groups and respective colors. See example below. (optional)
`line_width` (float)	Line width of the rendered elements. (optional)
`filename` (str)	Filename to save an image of the modeshape. (optional)

Input arguments to diaplay node and element tags can be used in any combination.

`vfo.plot_model()`
Displays the model using data from the active OpenSeesPy model with no node and element tags on it.

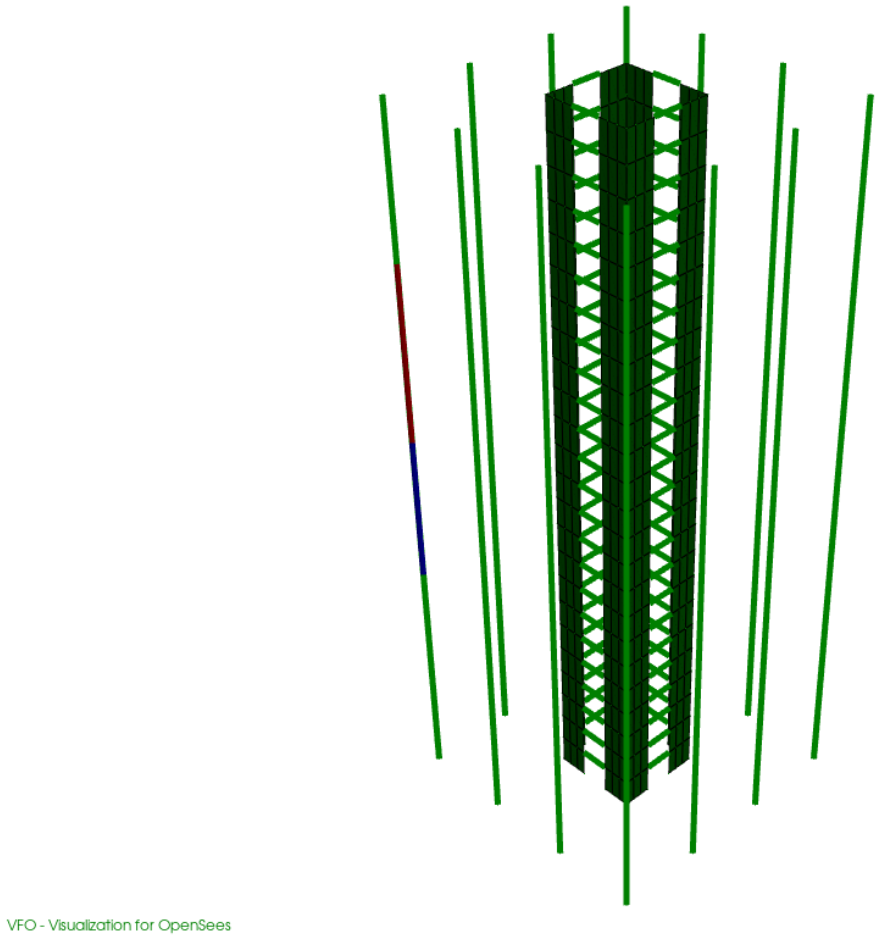
`vfo.plot_model(show_nodetags="yes")`
Displays the model using data from the active OpenSeesPy model with only node tags on it.

`vfo.plot_model(show_eletags="yes")`
Displays the model using data from the active OpenSeesPy model with only element tags on it.

`vfo.plot_model(show_nodetags="yes",show_eletags="yes")`
Displays the model using data from the active OpenSeesPy model with both node and element tags on it.

`vfo.plot_model(model="ModelName",show_nodetags="yes")`
Displays the model using data from an existing database "ModelName_ODB" with only node tags on it.

`vfo.plot_model(elementgroups=[[[1,2,3],[10,11]],["red","blue"]])`
Displays the model using data from the active model with elements [1,2,3] in "red" and elements [10,11] in "blue" colors.



CHAPTER

FIVE

PLOT_MODESHAPE COMMAND

vfo.**plot_modeshape**(*model='none'*, *modenumber=1*, *scale=10*, *setview='3D'*, *contour='none'*, *contourlimits=None*, *line_width=1*, *filename=None*)

Any modeshape can be visualized using this command. There is no need to perform an eigen analysis exclusively since the command runs an eigen analysis internally.

`model` (str)	name of the model output database as used in createODB() function. If no name is provided, the function tries to get data from the active model. Default is "none". (optional)
`modenumber` (int)	Mode number to visualize. For example: plot_modeshape(modenumber=3).
`scale` (float)	Scale factor for to display mode shape. (optional, default is 10)
`setview` (str)	Sets the camera view to predefined angles. Valid entries are "xy","yz","xz","3D", or a list with [x,y,z] unit vector. Default is "3D". (optional)
`contour` (str)	Contours of displacement in x, y, or z. Default is "none". (optional)
`contourlimits` (list)	A list of minimum and maximum limits of the displacement contours. (optional)
`line_width` (float)	Line width of the rendered elements. (optional)
`filename` (str)	Filename to save an image of the modeshape. (optional)

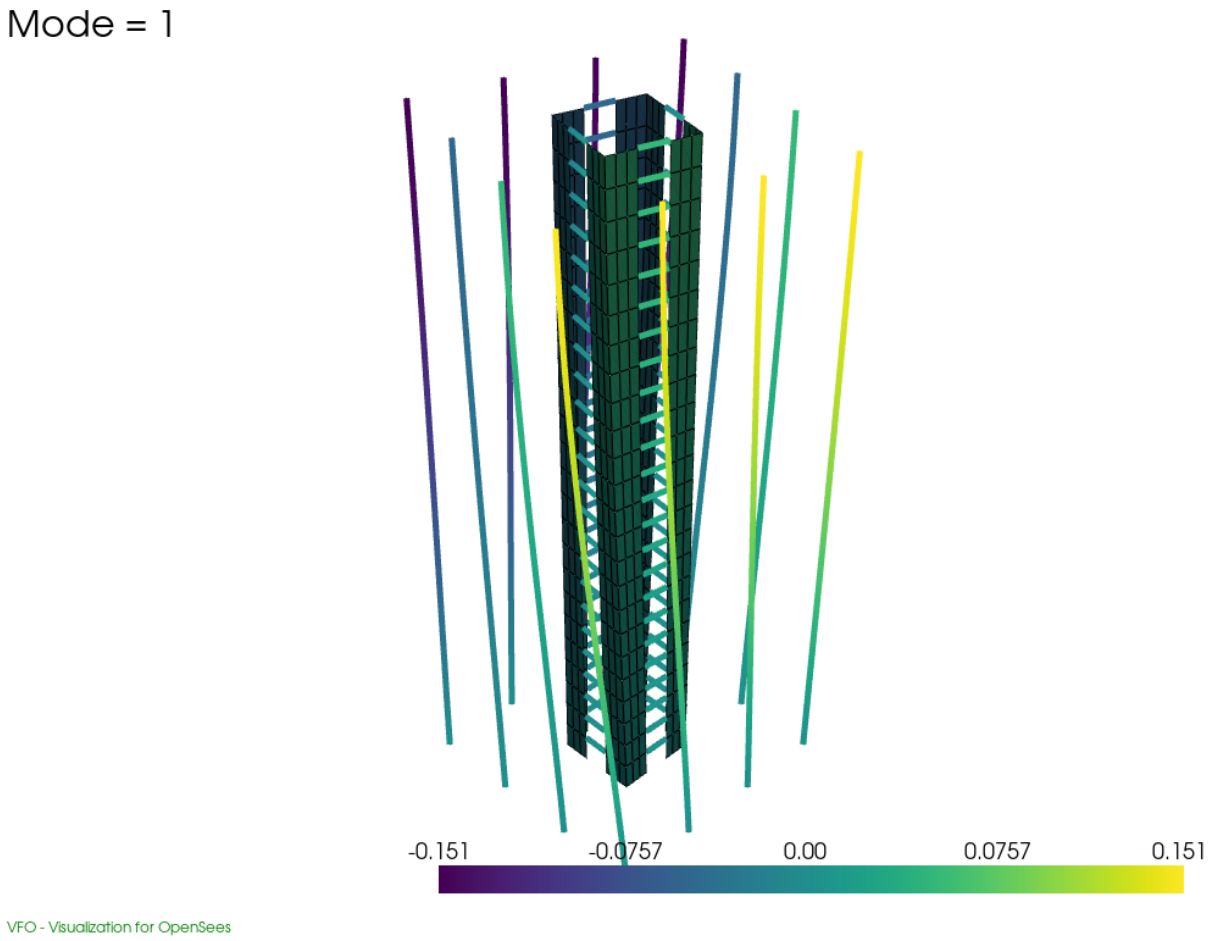
Example:

`vfo.plot_modeshape(modenumber=1, scale=300)`

Displays the 3rd modeshape using data from the active OpenSeesPy model with a scale factor of 300. This command should be called from an OpenSeesPy model script once the model is completed. The model should successfully work for Eigen value analysis. There is no need to create a database with *createODB()* command to use this option.

`vfo.plot_modeshape(modenumber=1, scale=300, model="3D_Building")`

Displays the 3rd modeshape using data from *3D_Building_ODB* database with a scale factor of 300. This command can be called from cmd, Ipython notebook or any Python IDE. An output database using *createODB()* has to be present in the current directory to use this command.


Mode = 1
-0.151
-0.0757
0.00
0.0757
0.151
VFO - Visualization for OpenSees

CHAPTER

SIX

PLOT_DEFORMEDSHAPE COMMAND

vfo.**plot_deformedshape**(*model='ModelName'*, *loadcase='LoadCaseName'*, *scale=10*, *tstep=-1*, *overlap='no'*, *contour='none'*, *setview='3D'*, *line_width=1*, *contourlimits=None*, *filename=None*)

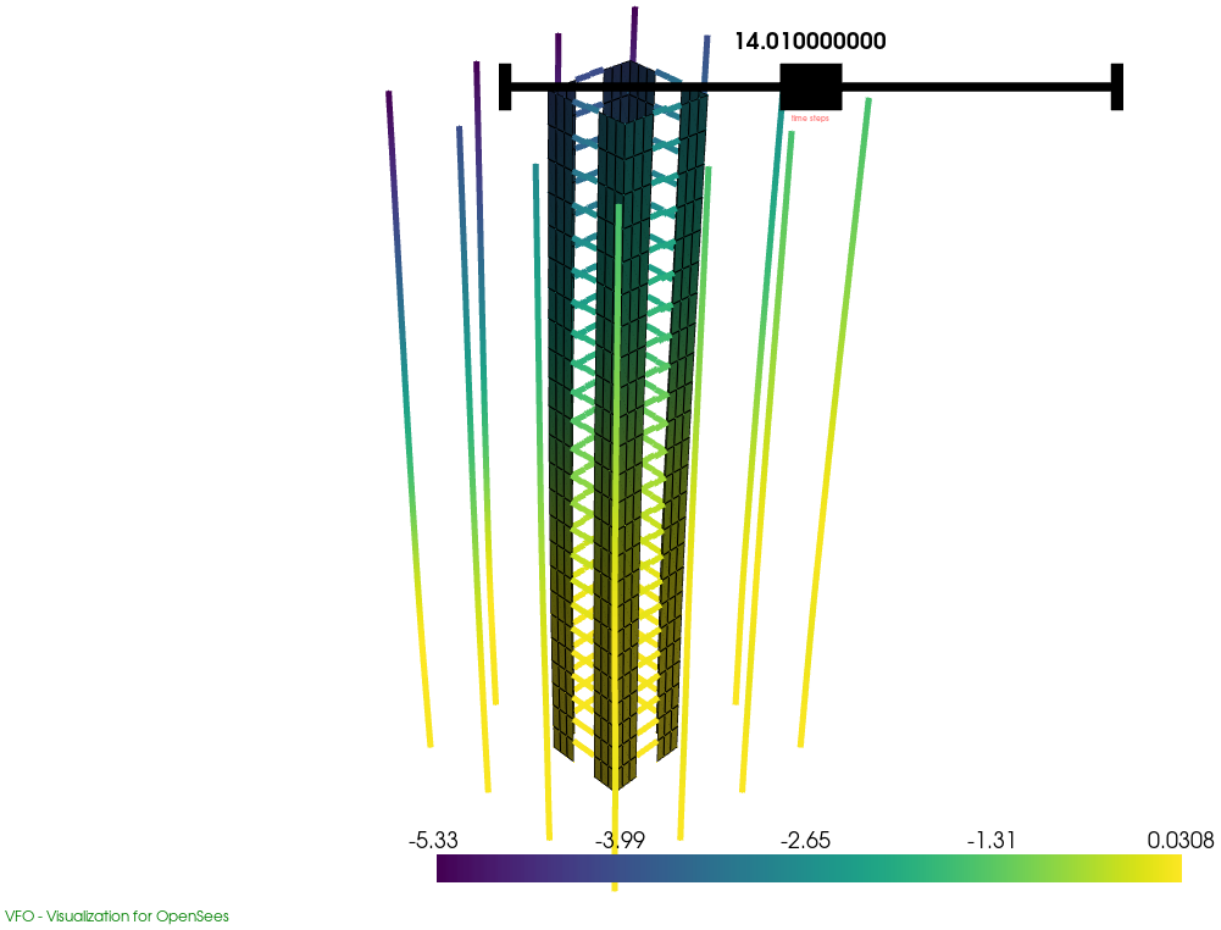
Displays the deformed shape of the structure by reading data from a saved output database.

`model` (str)	Name of the model to read data from output database, created with *createODB()* command.
`loadcase` (str)	Name of the subfolder with load case output data.
`scale` (float)	Scale factor for to display mode shape. (optional, default is 10)
`tstep` (float)	Approximate time of the analysis at which deformed shape is to be displaced. (optional, default is the last step)
`setview` (str)	sets the camera view to predefined angles. Valid entries are “xy”,”yz”,”xz”,”3D”, or a list with [x,y,z] unit vector. Default is “3D”. (optional)
`line_width` (float)	Line width of the rendered elements. (optional)
`contour` (str)	Contours of displacement in x, y, or z. Default is “none”. (optional)
`contourlimits` (list)	A list of minimum and maximum limits of the displacement contours. (optional)
`filename` (str)	Filename to save an image of the modeshape. (optional)

Examples:

vfo.plot_deformedshape(model="3D_Building", loadcase="Dynamic_GM1")
Displays the deformedshape of structure by reading data from *3D_Building_ODB* with a sub-folder *Dynamic_GM1* at the last analysis step (default) with a default scale factor of 10.

vfo.plot_deformedshape(model="3D_Building", loadcase="Dynamic_GM1", tstep=24.0, scale=50)
Displays the deformedshape of structure by reading data from *3D_Building_ODB* with a sub-folder *Dynamic_GM1* at the analysis time closest to 24.0 sec with a scale factor of 50.


14.010000000
time steps
-5.33
-3.99
-2.65
-1.31
0.0308
VFO - Visualization for OpenSees

CHAPTER

SEVEN

ANIMATE_DEFORMEDSHAPE COMMAND

vfo.animate_deformedshape(model="ModelName",loadcase="LoadCaseName",scale=10,speedup=1, contour="none",setview="3D",line_width=1,node_for_th=None,node_dof=1, moviename=None,gifname=None)

Displays an animation of the deformed structure by reading data from a saved output database. The animation object should be stored as an variable in order for the animation to run.

When saving the animation as a GIF file or .mp4 movie, the on-screen animation window may seem flickering based on the computer hardware.

`model` (str)	Name of the model to read data from output database, created with *createODB()* command.
`loadcase` (str)	Name of the subfolder with load case output data.
`scale` (int)	Scale factor for to display mode shape. (optional, default is 10)
`speedup` (int)	A factor to speedup the animation. (optional, The default is 1)
`setview` (str)	sets the camera view to predefined angles. Valid entries are “xy”,”yz”,”xz”,”3D”, or a list with [x,y,z] unit vector. Default is “3D”. (optional)
`contour` (str)	Contours of displacement in x, y, or z. Default is “none”. (optional)
`line_width` (float)	Line width of the rendered elements. (optional)
`node_for_th` (int)	Node ID to display displacement time-history. (optional)
`node_dof` (int)	Degree-of-freedom to display displacement time-history of node_for_th. Default is 1. (optional)
`moviename` (str)	Filename to save animation as a movie in .mp4 format. (optional)
`gifname` (str)	Filename to save animation as a movie in .gif format. (optional)

Examples:

```
ani = vfo.animate_deformedshape(model="3D_Building", loadcase="Dynamic_GM1")
```

The above command animates the deformedshape of structure by reading data from *3D_Building_ODB* with a subfolder *Dynamic_GM1* at default time-step speed.

```
ani = vfo.animate_deformedshape(model="3D_Building", loadcase="Dynamic_GM1", speedup=4,␣
↪scale=50, gifname="Building_Dynamic")
```

The above command animates the deformedshape of structure by reading data from *3D_Building_ODB* with a subfolder *Dynamic_GM1* at a speed 4x the default saved steps using a scale factor of 50. The animation movie will be saved as *Building_Dynamic.mp4* in the current folder.

CHAPTER

EIGHT

PLOT_FIBERRESPONSE2D COMMAND

vfo.**plot_fiberResponse2D**(*ModelName*, *LoadCaseName*, *element*, *section*, *LocalAxis='y'*, *InputType='stress'*, *tstep=-1*)

Plots the fibre stress or strain distribution along the local Z or Y-axis of a 2D fiber section. The local axis appears on the x axis, while the fiber response appears on the y axis.

`ModelName` (str)	Name of the model to read data from output database, created with *createODB()* command.
`LoadCaseName` (str)	Name of the subfolder with load case output data.
`element` (int)	Tag of the element where the section to be plotted is located.
`section` (int)	Tag of the section to be plotted.
`LocalAxis` (str)	Local axis of the section, based on a user defined axes transformation. (optional, default is "Y")
`InputType` (str)	Type of the fiber response to be plotted, `"stress"` or `"strain"`. (optional, default is "stress")
`tstep` (float)	Approximate time of the analysis at which fiber response is to be plotted. The program will find the closed time step to the input value.(optional, default is the last step).

Examples:

`vfo.plot_fiberResponse2D("TwoSpan_Bridge", "Dynamic_GM1", 101, 2)`

Plots the fiber stress (default) distribution of section 2 in element 101 of structure by reading data from *TwoSpan_Bridge_ODB* with a sub-folder *Dynamic_GM1* at the last analysis step (default).

CHAPTER

NINE

ANIMATE_FIBERRESPONSE2D COMMAND

vfo.animate_fiberResponse2D(Model, LoadCase, element, section, LocalAxis='y', InputType='stress', skipStart=0, skipEnd=0, rFactor=1, outputFrames=0, fps=24, Xbound=[], Ybound=[])

Animates fibre stress or strain distribution along the local Z or Y-axis of a 2D fiber section. The local axis appears on the x axis, while the fiber response appears on the y axis. The animation object should be stored as an variable in order for the animation to run.

`ModelName` (str)	Name of the model to read data from output database, created with *createODB()* command.
`LoadCaseName` (str)	Name of the subfolder with load case output data.
`element` (int)	Tag of the element where the section to be plotted is located.
`section` (int)	Tag of the section to be plotted.
`LocalAxis` (str)	Local axis of the section to appear on the x axis. (optional, default is "Y")
`InputType` (str)	Type of the fiber response to be plotted, possible values are `"stress"` or `"strain"`. (optional, default is "stress")
`skipStart` (int)	If specified, this many datapoints will be skipped from the data start. (optional, default is 0)
`skipEnd` (int)	If specified, this many datapoints will be skipped from the data end. (optional, default is 0)
`rFactor` (int)	If specified, only every "x" frames will be output by this factor. For example, if x=2, every other frame will be used in the animation. (optional, default is 0)
`outputFrames` (int)	The number of frames to be included after all other reductions. (optional, default is 0)
`fps` (str)	Number of animation frames to be displayed per second. (optional, default is 24)
`Xbound` (list)	``[xmin, xmax]` The domain of the chart. (optional, default is 1.1 the max and min values)
`Ybound` (float)	``[ymin, ymax]` The domain of the chart. (optional, default is 1.1 the max and min values)

Examples:

ani = vfo.animate_fiberResponse2D("TwoSpan_Bridge", "Dynamic_GM1", 101, 2)

Animates the fiber stress (default) distribution of section 2 in element 101 of structure by reading data from *TwoSpan_Bridge_ODB* with a sub-folder *Dynamic_GM1* at the last analysis step (default).

CHAPTER

TEN

PLOTTING OPENSEES TCL OUTPUT

OpenSees Tcl users can also take advantage of the plotting functions of *vfo_for_tcl* library. In order to do that, a Tcl script `vfo_for_tcl.tcl` to create an output database is used. First the user need to source *vfo_for_tcl.tcl* into the OpenSees tcl model file and then call the procedure to create an output database. This procedure does what `createODB()` does in vfo. Once the output database is created, users can call plot_model(), plot_modeshape(), plot_deformedshape() and animate_deformedshape() commands. Compatibility with other commands will be added in the next release.

Download the Tcl script here `vfo_for_tcl.tcl`.

`createODB "ModelName" "LoadCaseName" Nmodes`

`ModelName` (str)	Name of the model the user wants to save database with. Folder name will be *ModelName_ODB*
`LoadCaseName` (str)	`"none"` or `"LoadCaseName"`. Name of the subfolder to save load case output data.
`Nmodes` (int)	`0` or `Nmodes (int)`. Number of modes to be saved for visualization.

Note: To record modeshape data, this procedure utilizes an internal Eigenvalue analysis. Make sure your model is well defined to avoid errors.

Example: Here is a minimal example of how to use vfo_for_tcl.tcl.

```
# source the script in the beginning of the Tcl script.
source vfo_for_tcl.tcl

# create model here.
# define nodes, elements etc.
# Once the model definition is finished, call the procedure to record the first 3␣
↪modeshapes.
# When recording modeshapes, use "none" for the loadCaseName.

createODB "3DBuilding" "none"  3

# The above command will save all the data in a folder named "3DBuilding_ODB" and ...
# ... a sub-folder "Modeshapes".

# Now to record data from a dynamic loadcase, assign a name for load case folder and ...
# ... the number 0 to Nmodes to avoid performing Eigenvalue analysis again.

createODB "3DBuilding" "Dynamic" 0

# The above command will save the node displacement data to a sub-folder "Dynamic" in ...
# ... the "3DBuilding_ODB" folder.
```

Now open a python terminal or Jupyter notebook and type the following. Make sure you install the latest version of OpenSeesPy first. Or, put the following lines in a Python script and run.

```
import vfo.vfo as vfo

# render the model with node and element tags on it
vfo.plot_model(model="3D_Building",show_nodetags="yes",show_eletags="yes")

# plot mode shape 2 with a scale factor of 100
vfo.plot_modeshape(modenumber=1, scale=300, model="3D_Building")

# animate the deformed shape for dynaic analysis and save it as a 3DBuilding.mp4 file.
vfo.animate_deformedshape(model="3D_Building", loadcase="Dynamic_GM1", gifname="Building_
↪Dynamic")
```

All figures are interactive and can be saved as a .png file from the plot window.

INDEX

B

V