
ModelOrderReduction Documentation

Release 1.0

Defrost Team

Jul 13, 2018

CONTENTS

1	Install	1
1.1	Requirement	1
1.2	Install & Get Sarted	2
2	Tutorials	5
2.1	Reduction Process Tutoriel	5
3	Examples	13
3.1	Diamond	13
3.2	Starfish	15
3.3	Sofia	18
4	Tools	21
4.1	mor.animation	21
4.2	mor.script	22
4.3	mor.wrapper	29
5	Components	33
5.1	Forcefields	33
5.2	Loaders	49
5.3	Mappings	49
6	Indices and tables	53
	Python Module Index	55
	Index	57

INSTALL

1.1 Requirement

This work is a plugin of [SOFA](#) which a simulation software. For the moment we haven't got any pre-made SOFA version with our work so the first thing you will need to do is compile SOFA

1.1.1 Step 1: Compile SOFA

For that follow these [instruction](#) (here is for linux but instruction exist for MAC & Windows also) Before compiling it you have to check in CMake-gui the option : ***SOFA_WITH_EXPERIMENTAL_FEATURES*** (will be removed in the future)

1.1.2 Step 2: Install Required dependencies

Ubuntu

- Python 2.7.X
- Cheetah

```
sudo apt-get update
sudo apt-get install python-cheetah
```

- PyQt

```
sudo apt-get install python-pyqt5
```

- SOFA Plugin Dependencies

(The best way to add plugin to SOFA is explained here <https://www.sofa-framework.org/community/doc/using-sofa/build-a-plugin/>)

STLIB with branch *stdlib_wrapper*

Plugin easing the way to write SOFA scene in python

```
git clone -b stdlib_wrapper https://github.com/SofaDefrost/STLIB.git
```

optional:

SoftRobots with branch *Documentation*

The different examples present in the plugin are based on scene requiring SoftRobots

```
git clone -b Documentation https://github.com/SofaDefrost/SoftRobots.git
```

- Sofa Launcher

We use a tool of SOFA named **sofa-launcher** allowing us to gain a lot of calculation time thanks to parallel execution of multiple SOFA scene. For that you have to add to your *.bashrc* in the end the following line:

```
export PYTHONPATH=/PathToYourSofaSrcFolder/tools/sofa-launcher
```

OSX

TODO

Windows

TODO

Optionnal

To learn how to reduce your own model we have done a tutorial which will make you learn step by step the process. For this interactive tutorial we use a [python notebook](#).

So to be able to do it please install jupyter

the easiest way:

```
pip install jupyter
```

1.2 Install & Get Sarted

1.2.1 Download

Now that you have a compiled & working SOFA, we can add our plugin.

- Create a directory where you will put it
- Move into it and clone our last working version:

```
git clone https://github.com/SofaDefrost/ModelOrderReduction
```

1.2.2 Install

- Add it to SOFA
 - With CMake-gui add to the key **SOFA_EXTERNAL_DIRECTORIES** the path to your folder
 - Then Configure, Generate and verify that it has been added (normally a message will pass in the console of CMake-gui)
- Re-compile SOFA

1.2.3 Launch test

To confirm all the previous steps and verify that the plugin is working properly you can launch the *test_component.py* SOFA scene situated in:

```
/ModelOrderReduction/tools
```

This example show that after the reduction of a model (here the 2 examples Diamond robot, Starfish robot), you can re-use it easily as a python object with different arguments allowing positionning of the model in the SOFA scene.

2.1 Reduction Process Tutoriel

Note: The following tutorial comes from a python-notebook. If you want to make the tutorial interactively go directly to:

```
/ModelOrderReduction/tools
```

then, if you have installed jupyter like explained in the requirement, open a terminal there and launch a session:

```
jupyter notebook
```

It will open in your web-Browser a tab displaying the current files in the directory. Normally you should have one called **modelOrderReduction.ipynb**

You can click on it and follow the tutorial

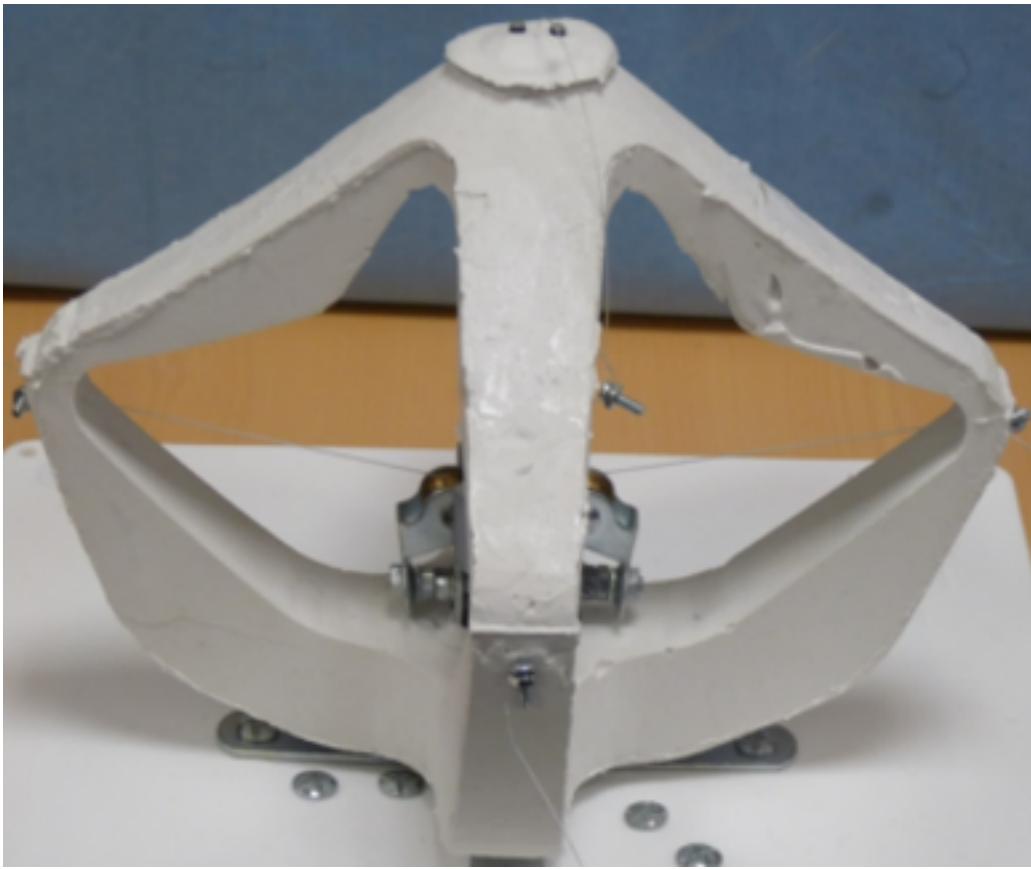
2.1.1 Model Order Reduction Example

Introduction

In this python notebook exemple we will see with 2 real examples how to reduce a model from one of your sofa scene thanks to the **Model Order Reduction** plugin done by the INRIA research team **Defrost**.

the two examples will be :

- **The Diamond**



- The Starfish



After these example presentation we can now proceed to the reduction. First we have to prepare it by setting a bunch of parameters while explaining there purpose (here the parameters will be set twice, one for the diamond and one for the starfish so you will be able to switch easily between each example)

User Parameters

here some import that will be usefull for this python notebook

```
# Import
import os
import sys

sys.path.append(os.getcwd()+'../../python') # TO CHANGE

from plotly.offline import init_notebook_mode, iplot
import plotly.graph_objs as go
init_notebook_mode(connected=True)

# MOR IMPORT
import mor.script.morUtilityFunctions as ui
from mor.script import ReduceModel
from mor.script import ObjToAnimate
```

The first important things to set are the different path where we will work:

- The scene you want to work on
- The folder containing its mesh
- The folder where you want the results to be put in

```
# Important path
originalScene = ui.openFileName('Select the SOFA scene you want to reduce')
meshDir = ui.openDirName('Select the directory containing the mesh of your scene')
outputDir = ui.openDirName('Select the directory tha will contain all the results')
```

Now The different parameters for the reduction

nodesToReduce

- *ie : list containing the model path you want to reduce.

For example if you want to reduce child2 which is in the sofa graph scene a child of child1 which is a child of root you would give this path : **'/child1/child2'** *

```
nodesToReduce_DIAMOND = [ '/modelNode' ]
nodesToReduce_STARFISH =[ ('/model', '/model/modelSubTopo') ]
```

listObjToAnimate

- *ie : contain a list of object from the class ObjToAnimate.

A ObjToAnimate will define an object to “animate” during the shaking, this animation is the variation of a particular value of this object with a certain increment on a with a minumum/maximum value to attain. There are 3 important parameter to this object :

- location : sofa node name in which we will work
- animFct : the animation function we will use (by default defaultShaking)
- objName : the object name we want to animate (by default None)

For exemple here we want to animate the node named “nord”, but we won’t specify either the animFct and objName so the default animation function will be used and be apply on the first default object it will find. The default function will need 2 additionnal parameters :

- increment
 - ie : *By which value we increment for each animation during one step*
- maxPull
 - ie : *The maximum value of each animation*

nord = ObjToAnimate(“nord”, incr=5,incrPeriod=10,rangeOfAction=40)

```
# animation parameters

### DIAMOND ROBOT PARAM
nord = ObjToAnimate("nord", incr=5,incrPeriod=10,rangeOfAction=40)
sud = ObjToAnimate("sud",incr=5,incrPeriod=10,rangeOfAction=40)
est = ObjToAnimate("est",incr=5,incrPeriod=10,rangeOfAction=40)
ouest = ObjToAnimate("ouest",incr=5,incrPeriod=10,rangeOfAction=40)
listObjToAnimate_DIAMOND = [nord,ouest,sud,est]

### STARFISH ROBOT PARAM
centerCavity = ObjToAnimate("centerCavity",incr=350,incrPeriod=2,rangeOfAction=3500)
rearLeftCavity = ObjToAnimate("rearLeftCavity",incr=200,incrPeriod=2,
                           ↴rangeOfAction=2000)
rearRightCavity = ObjToAnimate("rearRightCavity",incr=200,incrPeriod=2,
                           ↴rangeOfAction=2000)
frontLeftCavity = ObjToAnimate("frontLeftCavity",incr=200,incrPeriod=2,
                           ↴rangeOfAction=2000)
frontRightCavity = ObjToAnimate("frontRightCavity",incr=200,incrPeriod=2,
                           ↴rangeOfAction=2000)
listObjToAnimate_STARFISH = [centerCavity,rearLeftCavity,rearRightCavity,
                           ↴frontLeftCavity,frontRightCavity]
```

Modes parameters

- addRigidBodyModes
- tolModes
 - ie : *With which tolerance we want to select our modes (add explanation of the selection process)*

```
addRigidBodyModes_DIAMOND = [0,0,0]
addRigidBodyModes_STARFISH = [1,1,1]

tolModes = 0.001
```

- tolGIE
 - ie : *blabla ...*

```
# Tolerance
tolGIE = 0.05
```

And to finish some optionnal parameters

```
# Optionnal
verbose = False

packageName = 'test'
addToLib = False
```

We can now execute one of the reduction we choose with all these parameters

Execution

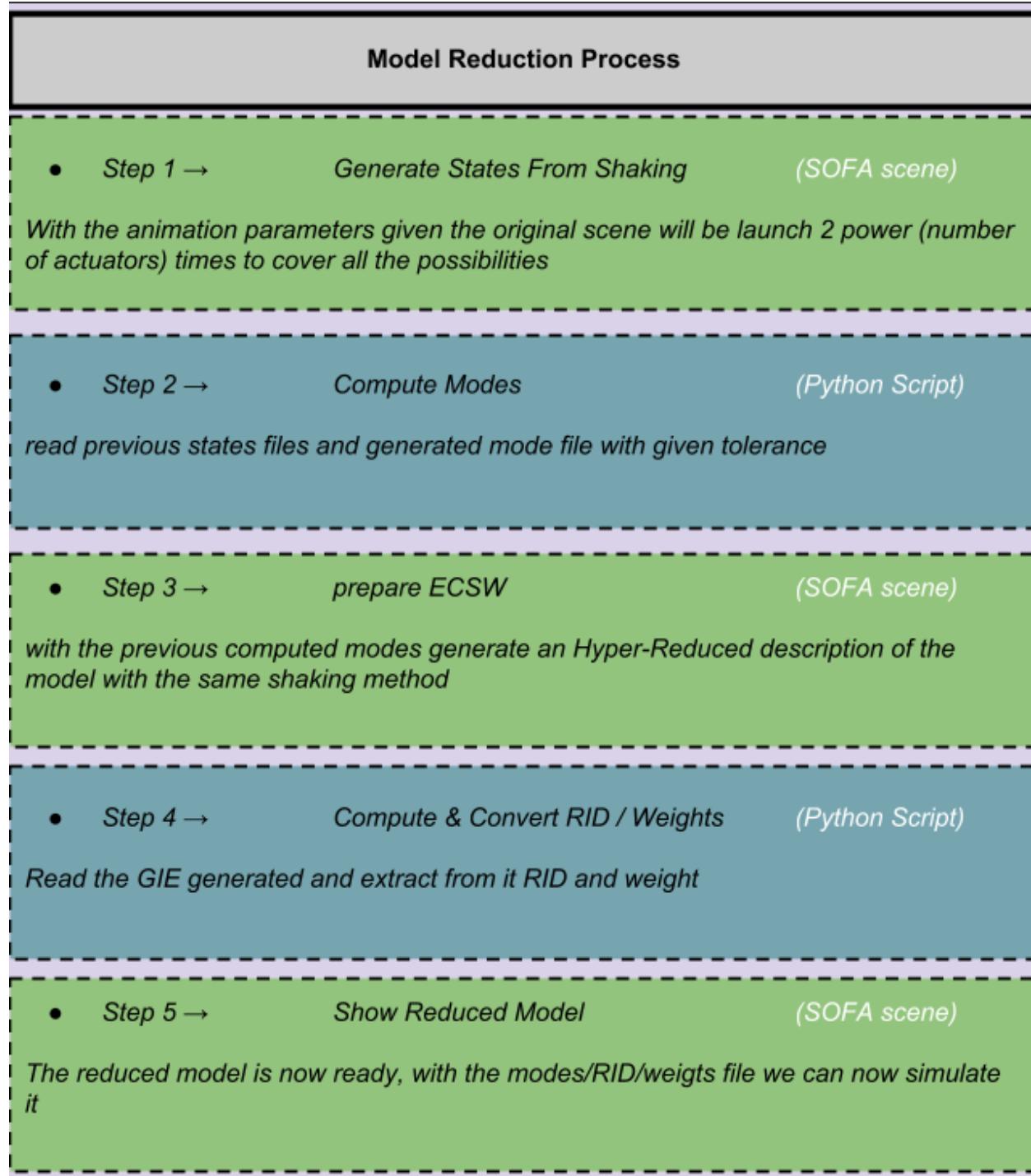
Initialization

The execution is done with an object from the class ReduceModel. we initialize it with all the previous argument either for the Diamond or Starfish example

```
# Initialization of our script
nodesToReduce = nodesToReduce_DIAMOND # nodesToReduce_STARFISH
listObjToAnimate = listObjToAnimate_DIAMOND # listObjToAnimate_STARFISH
addRigidBodyModes = addRigidBodyModes_DIAMOND # addRigidBodyModes_STARFISH

reduceMyModel = ReduceModel(
    originalScene,
    nodesToReduce,
    listObjToAnimate,
    tolModes, tolGIE,
    outputDir,
    meshDir,
    packageName = packageName,
    addToLib = addToLib,
    verbose = verbose,
    addRigidBodyModes = addRigidBodyModes)
```

We can finally perform the actual reduction. here is a schematic to resume the differents steps we will perform :



Phase 1

We modify the original scene to do the first step of MOR :

- We add animation to each actuators we want for our model
- And add a writeState component to save the shaking resulting states

```
reduceMyModel.phase1()
```

Phase 2

With the previous result we combine all the generated state files into one to be able to extract from it the different mode

```
reduceMyModel.phase2()
```

```
# Plot result
with open(reduceMyModel.packageBuilder.debugDir+'Sdata.txt') as f:
    content = f.readlines()

content = [x.strip() for x in content]

data = [go.Bar(x=range(1, len(content)+1),
               y=content)]

iplot(data, filename='jupyter/basic_bar')
```

```
print("Maximum number of Modes : ")
reduceMyModel.reductionParam.nbrOfModes
```

Phase 3

We launch again a set of sofa scene with the sofa launcher with the same previous arguments but with a different scene
This scene take the previous one and add the model order reduction component:

- HyperReducedFEMForceField
- MappedMatrixForceFieldAndMass
- ModelOrderReductionMapping and produce an Hyper Reduced description of the model

```
reduceMyModel.phase3()
```

Phase 4

Final step : we gather again all the results of the previous scenes into one and then compute the RID and Weights with it. Additionnally we also compute the Active Nodes

```
reducedScene = reduceMyModel.phase4()
```

End of example you can now go test the results in the folder you have designed at the beginning of this tutorial

To go Further

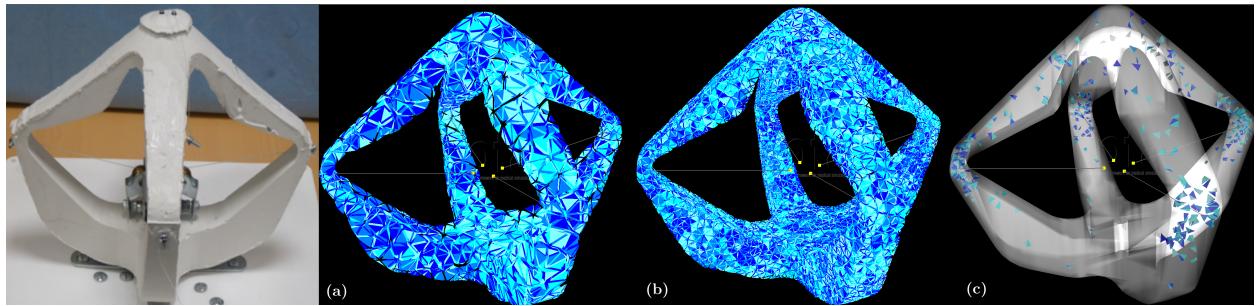
Here are some link to additionnal information about the plugin and a more detailed description of its operation

[link to Olivier paper](#)

[link to website doc](#)

EXAMPLES

3.1 Diamond



3.1.1 Presentation

The diamond robot is a demonstrator of our team showing control of actuators by SOFA simulation. There are multiples papers which have been written using it, here a [link](#) to them, we also use it for our model order reduction paper ([link](#)).

Brief description :

the robot is entirely made of soft silicone and is actuated by four cables controlled by step motors located at its center. Pulling on the cables has the effect of lifting the effector located on top of the robot. The “game” with this robot is to control the position of the effector by pulling on the cables.

Little video of presentation showing it in action

Why reduce it :

Previously the robot was controlled through real-time finite element simulation based on a mesh of 1628 nodes and 4147 tetrahedra. That size of mesh was manageable in real-time on a standard desktop computer. The simulation made using this underlying mesh was accurate enough to control the robot, only considering the displacement of the effector point, located on the top of the robot and with a limited range on the pulling of the cable actuators.

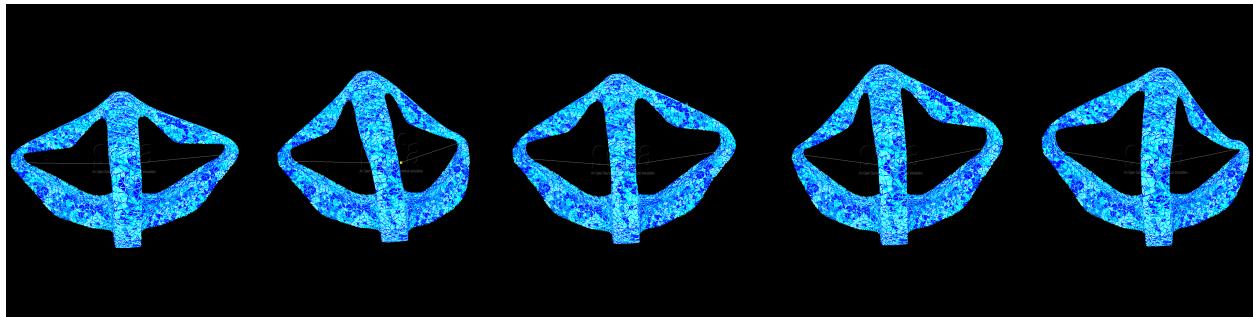
However, this does not show that the actual position of each of the four arms of the robot was accurately predicted for example. When considering an application where the robot arms may enter in contact with the environment, an accurate prediction of their position becomes relevant.

To have this accuracy we need a much more finer mesh which will demand some intensive calculations and in the process we will lose the real-time simulation of it. So here comes our plugin to resolve this issue.

3.1.2 Reduction Parameters

To reduce this robot we will use the `defaultShaking(link!)` function to shake it because we just need for actuators to perform simple incrementation along there working interval (here `[0 .. 40]` with an increment of 5)

After that with a reasonable tolerance (here `0.001`) we will select different modes, here some possible modes selected :

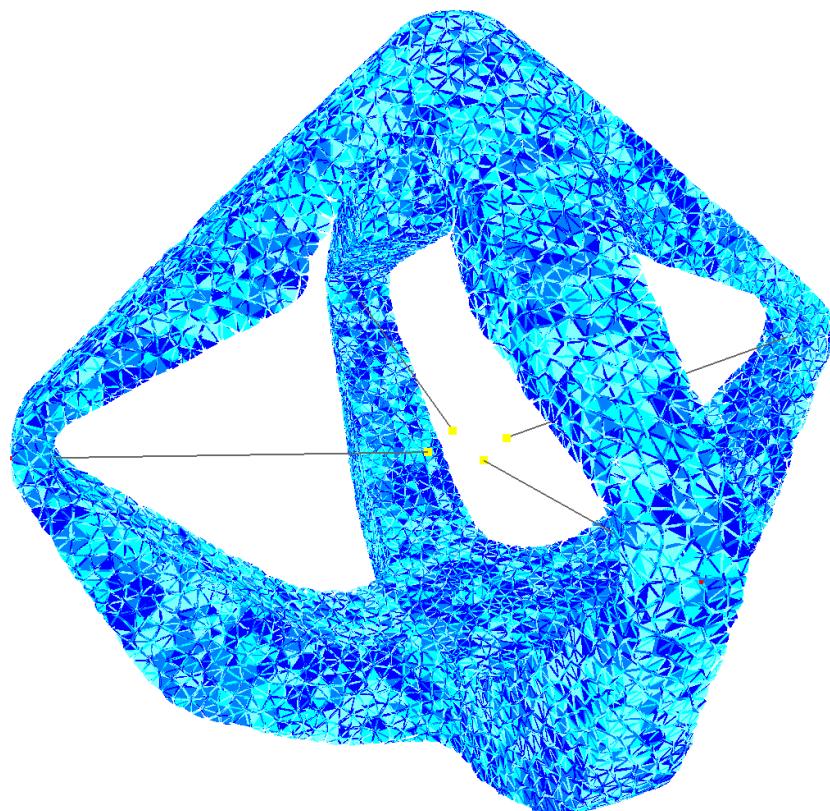


With these different parameters we will after perform the reduction like explained [here](#)

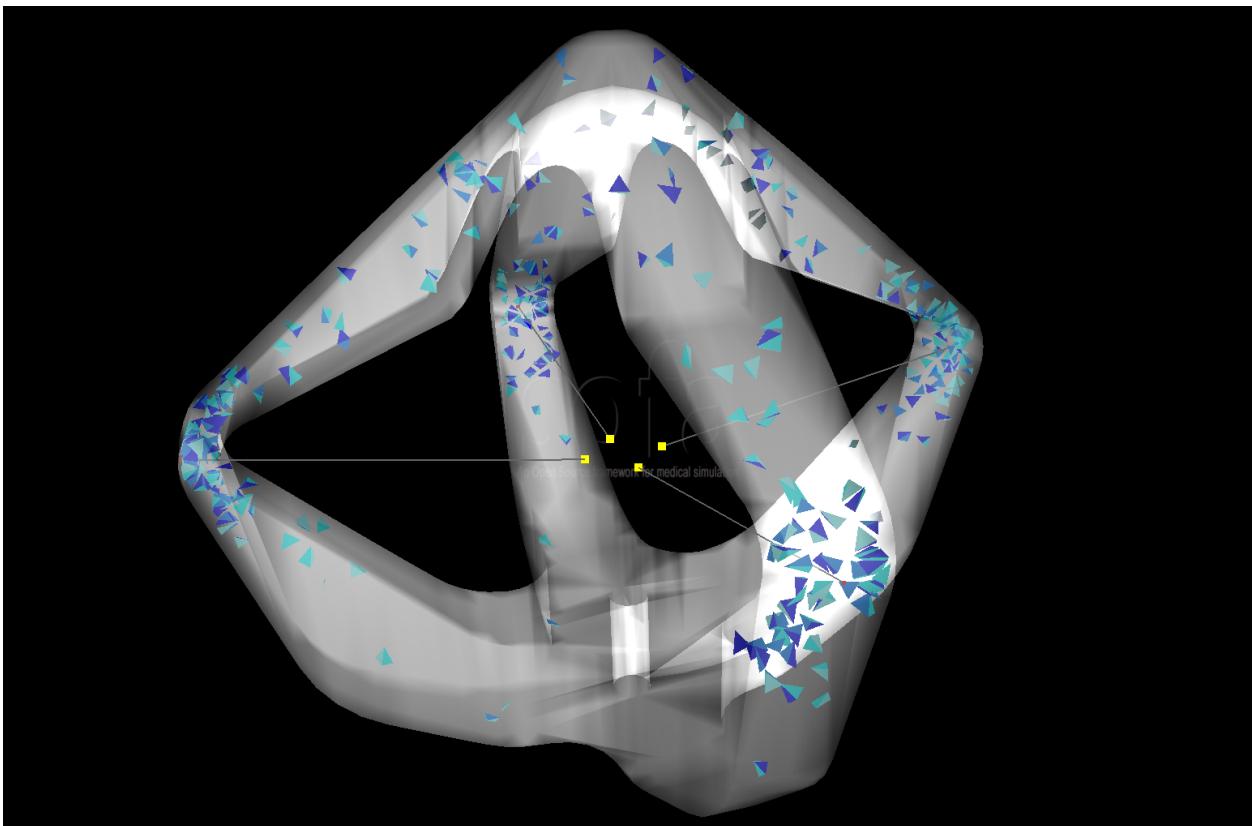
3.1.3 Results

example results with a fine mesh:

Before

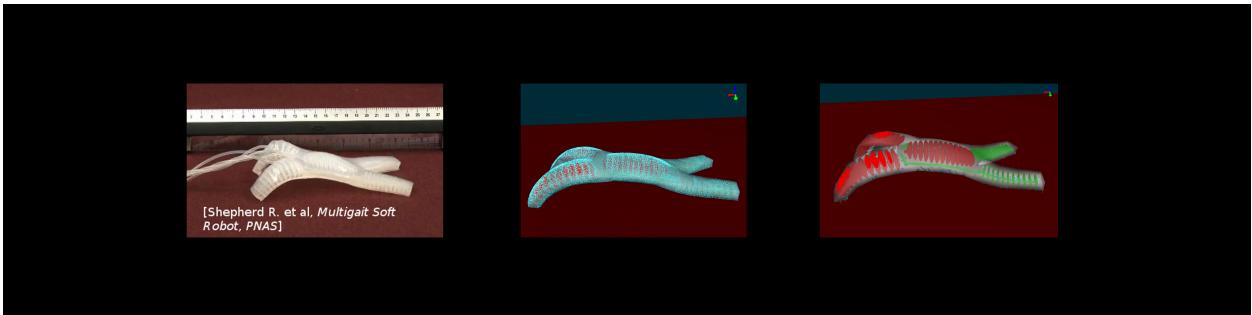


After



For more details about the results, displacement error comparison, test with different mesh and other, you can read the paper affiliated with this plugin here([link!](#))

3.2 Starfish



3.2.1 Presentation

the Sarfish robot or more precisely the multigait soft robot come from the work described here : [paper link](#)

Brief description :

This robot is made of two layers: one thick layer of soft silicone containing the cavities, and one stiffer and thinner layer of Polydimethylsiloxane (PDMS) that can bend easily but does not elongate. The robot is actuated by five air cavities that can be actuated independently. The effect of inflating each cavity is to create a motion of bending. Then, by actuating with various sequences each cavities, the robot can move along the floor.

Little video of presentation showing it in action :

Why reduce it :

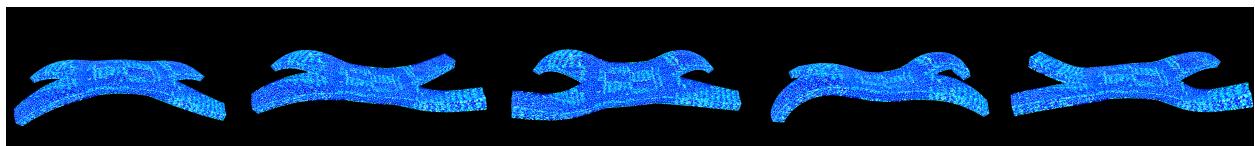
The simulation of this crawling robot has to be really precise in order to simulate properly the different deformations and the contact with the floor has shown in the previous video.

This needs of precision results with heavy calculations when the simulation is running preventing the fluidity of it, by reducing it we will be able to resolve this issue and also show that we the reduce model can move and handle contact in comparison with the previous example *Diamond robot* that was fixed.

3.2.2 Reduction Parameters

To reduce this robot we will use the `defaultShaking(link!)` function to shake it because we just need for actuators to perform simple incrementation along there working interval (here `[0 .. 2000 or 3500]` with an increment of `200 or 350`)

After that with a reasonable tolerance (here `0.001`) we will select different modes, here some possible modes selected :

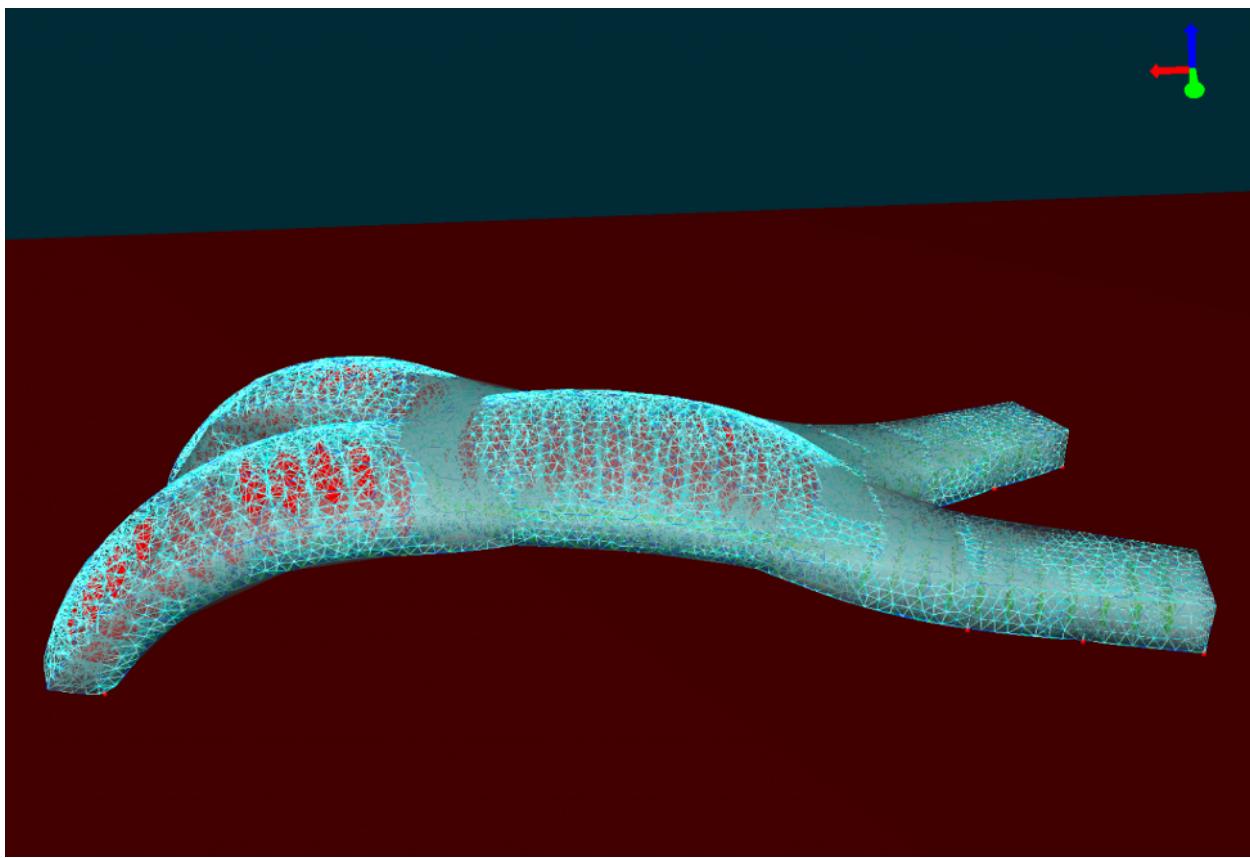


With these different parameters we will after perform the reduction like explained [here](#)

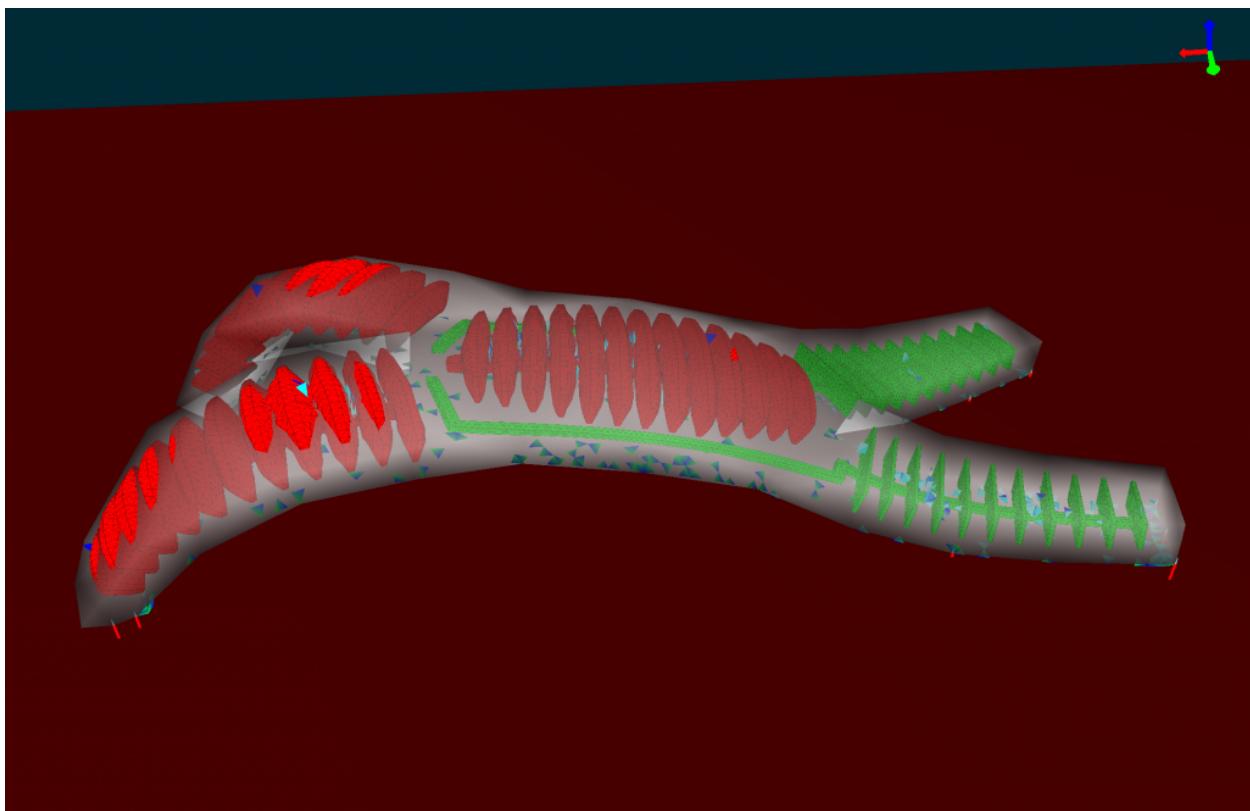
3.2.3 Results

example results with a fine mesh:

Before

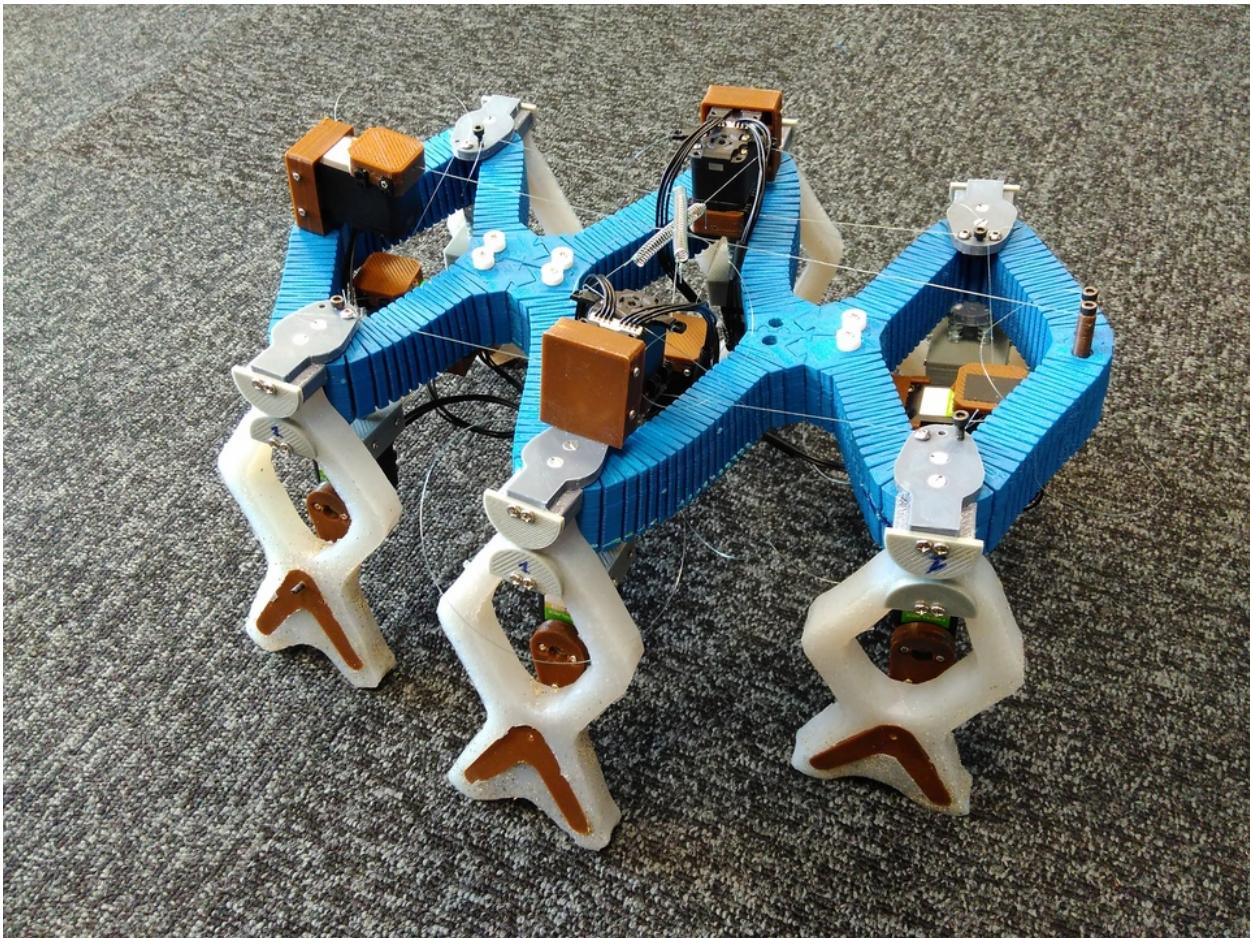


After



For more details about the results, displacement error comparison, test with different mesh and other, you can read the paper affiliated with this plugin here([link!](#))

3.3 Sofia



3.3.1 Presentation

Brief description :

presentation video of the simulation showing it in action:

video of the realisation based on the previous simulation:

Why reduce it :

To show that we can easily reduce parts of a soft robot and re-use it in the full robot. Here we only reduce the leg of our robot not its core.

3.3.2 Reduction Parameters

To reduce this robot we had to create a new special function to shake. To create the rotation movement we see on the different previous videos we rotate a point that will be followed by the model creating the rotation.

[here](#) how it was implemented

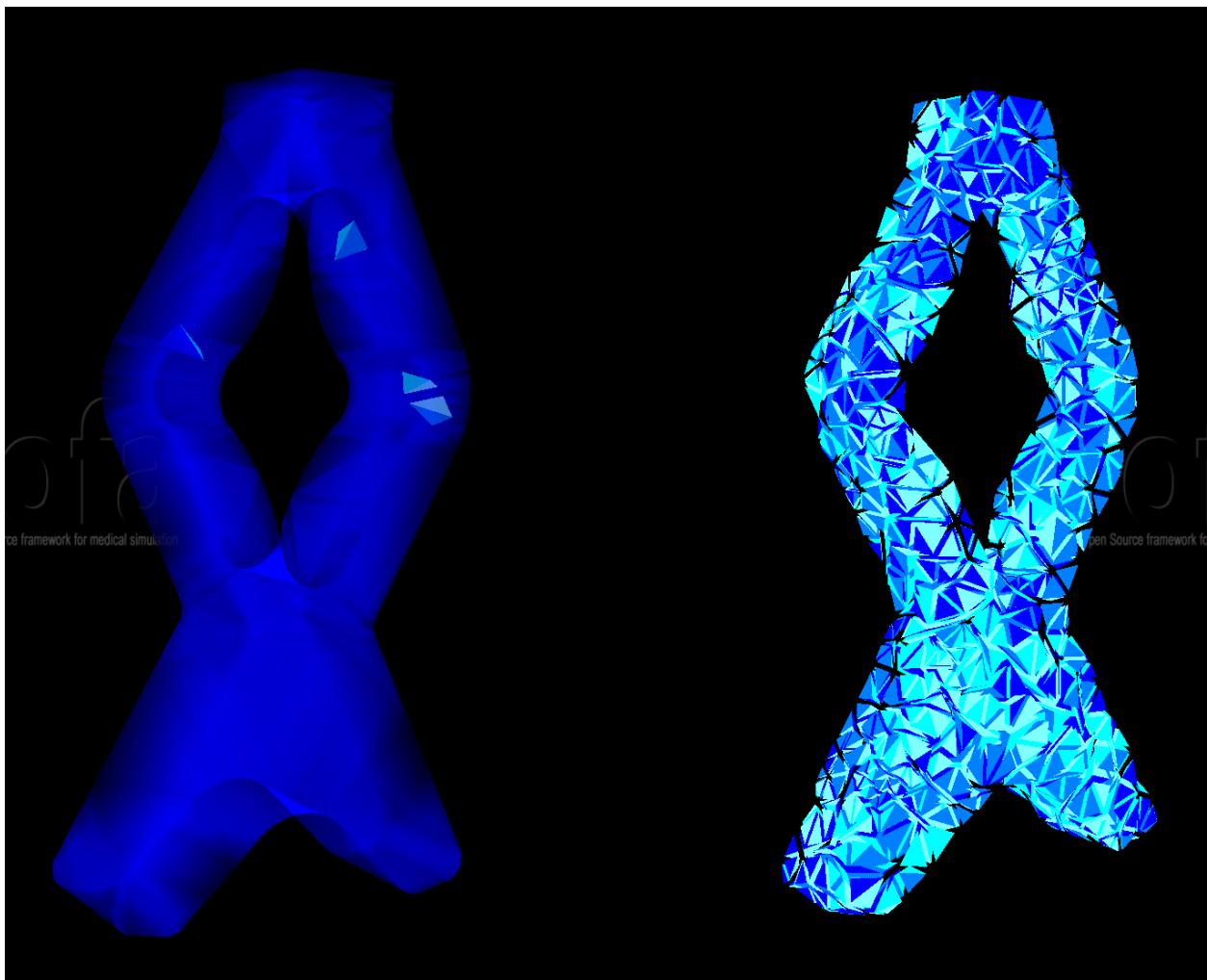
We have only one actuator here, so our *listObjToAnimate* contains only one object:

```
ObjToAnimate("actuator", "shakingSofia", 'MechanicalObject', incr=0.05, incrPeriod=3,
             rangeOfAction=6.4, dataToWorkOn="position", angle=0, rodRadius=0.7)
```

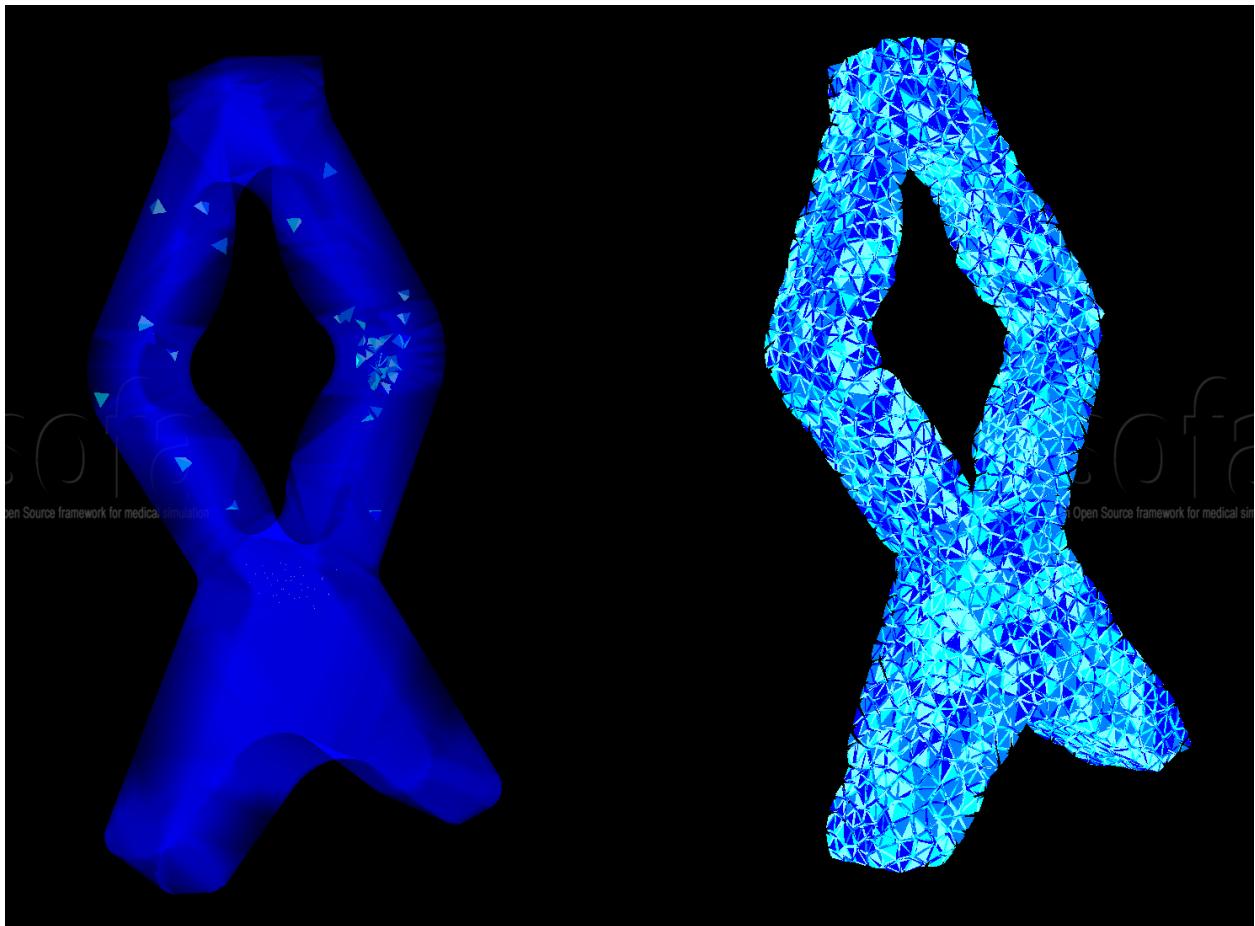
With these different parameters we will after perform the reduction like explained [here](#)

3.3.3 Results

With coarse mesh



With fine mesh



CHAPTER FOUR

TOOLS

<i>mor.animation</i>	Set of predefined function to shake our model during the reduction
<i>mor.script</i>	Set of Function/Class to perform Model Reduction
<i>mor.wrapper</i>	Set of class and functions to modify the SOFA scene during its construction or save it to make a reusable component

4.1 mor.animation

Set of predefined function to shake our model during the reduction

Each function has to have 3 mandatory arguments:

argument	type	definition
objToAnimate	ObjToAniamte	the obj containing all the information/arguments about the animation
dt	sc	Time sept of the Sofa scene
factor	float	<p>Argument given by the Animation class from STLlib.</p> <p>It indicate where we are in the animation sequence:</p> <ul style="list-style-type: none">• 0.0 —> beginning of sequence• 1.0 —> end of sequence <p>It is calculated as follow:</p> <pre>factor = (currentTime - startTime) / duration</pre>

Content:

<code>mor.animation.defaultShaking(objToAnimate, increase a value of a Sofa object until it reach its maximum ...)</code>	
<code>mor.animation.shakingSofia(objToAnimate, dt, ...)</code>	was made specifically to shake the Sofia Leg and can be an example as an custom shaking animation
<code>mor.animation.shakingInverse(objToAnimate, TODO ...)</code>	

4.1.1 mor.animation.defaultShaking

`mor.animation.defaultShaking(objToAnimate, dt, factor, **param)`
increase a value of a Sofa object until it reach its maximum

For this objToAnimate.params arguments which is a dic will need 3 keys:

Keys:

argument	type	definition
dataToWorkOn	str	Name of the Sofa datafield we will work on by default it will be set to <i>value</i>
incrPeriod	float	Period between each increment
incr	float	Value of each increment
rangeOfAction	float	Until which value the data will increase

4.1.2 mor.animation.shakingSofia

`mor.animation.shakingSofia(objToAnimate, dt, factor, **param)`
 was made specifically to shake the Sofia Leg and can be an example as an custom shaking animation

It take a position, rotate it and update it in the component

For this shaking, objToAnimate.params arguments will need 3 keys:

Keys:

argument	type	definition
dataToWorkOn	str	Name of the Sofa datafield we will work on here it will be <i>position</i>
incrPeriod	float	Period between each increment
incr	float	Value of each increment
rangeOfAction	float	Until which value the data will increase
angle	float	Initial angle value in radian
rodRadius	float	Radius Length of the circle

4.1.3 mor.animation.shakingInverse

`mor.animation.shakingInverse(objToAnimate, dt, factor, **param)`
 TODO

4.2 mor.script

Set of Function/Class to perform Model Reduction

Content:

<code>mor.script.reduceModel</code>	Set of class simplifying and allowing to perform ModelReduction
<code>mor.script.morUtilityFunctions</code>	Utilities functions used mainly by the reduceModel classes

4.2.1 mor.script.reduceModel

Set of class simplifying and allowing to perform ModelReduction

Content:

<code>mor.script.reduceModel.ObjToAnimate(location)</code>	ObjToAnimate is a class allowing us to store in 1 object all the information about a specific animation
<code>mor.script.reduceModel.ReductionAnimations(...)</code>	Contain all the parameters & functions related to the animation of the reduction
<code>mor.script.reduceModel.PackageBuilder(...[, ...])</code>	Contain all the parameters & functions related to building the package
<code>mor.script.reduceModel.ReductionParam(...)</code>	Contain all the parameters related to the reduction
<code>mor.script.reduceModel.ReduceModel(...[, ...])</code>	Main class that will perform the reduction

mor.script.reduceModel.ObjToAnimate

```
class mor.script.reduceModel.ObjToAnimate(location, animFct='defaultShaking', objName=None, node=None, obj=None, duration=-1, **params)
```

ObjToAnimate is a class allowing us to store in 1 object all the information about a specific animation

Args

argument	type	definition
location	Str	Name of the Sofa Node where our obj to animate is
animFct	Str	<p>Name of our function we want to use to animate.</p> <p>During execution of the Sofa Scene, it will import the module mor.animation.animFct where your function has to be located in order to be used</p>
objName	Str	Name of our Sofa obj
node	Sofa.Node	pointer to Sofa node in which we are working on (will be set during execution)
obj	Sofa.Obj	pointer to Sofa obj working on (will be set during execution)
duration	sc	Total time in second of the animation (put by default to -1 & will be calculated & set later in the execution)
**params	undefined	You can put in addation whatever parameters you will need for your specific animation function

Example :

```
ObjToAnimate("nord","defaultShaking", incr=5,incrPeriod=10,rangeOfAction=40)
```

mor.script.reduceModel.ReductionAnimations

```
class mor.script.reduceModel.ReductionAnimations (listObjToAnimate)
Contain all the parameters & functions related to the animation of the reduction

setNbIteration (nbIterations=None)
generateListOfPhase (nbPossibility, nbActuator)
```

mor.script.reduceModel.PackageBuilder

```
class mor.script.reduceModel.PackageBuilder (outputDir, meshDir, toKeep, package-Name=None, addToLib=False)
Contain all the parameters & functions related to building the package

copy (src, dest)
```

```

checkExistance (dir)
checkNodeNbr (modeFileName)
cleanStateFile (periodSaveGIE, stateFileName)
copyFileIntoAnother (fileToCopy, fileToPasteInto)
copyAndCleanState (results, periodSaveGIE, stateFileName, gie=None)
finalizePackage (result)
addToLib ()

```

mor.script.reduceModel.ReductionParam

```

class mor.script.reduceModel.ReductionParam (tolModes, tolGIE, addRigidBodyModes,
                                              dataDir)
Contain all the parameters related to the reduction

setNbTrainingSet (rangeOfAction, incr, nbPossibility)

addParamWrapper (nodeToReduce, prepareECSW=True, subTopo=None, paramForcefield=None,
                  paramMappedMatrixMapping=None, paramMORMapping=None)

setFilesName ()

```

mor.script.reduceModel.ReduceModel

```

class mor.script.reduceModel.ReduceModel (originalScene, nodesToReduce, listObjToAnimate,
                                           tolModes, tolGIE, outputDir, meshDir,
                                           packageName=None, toKeep=None, addToLib=False,
                                           verbose=False, addRigidBodyModes=False, nbrCPU=4)

```

Main class that will perform the reduction

argument	type	definition
originalScene	str	absolute path to original scene
nodesToReduce	list(str)	list of paths to models to reduce
listObjToAnimate	ObjToAnimate	list containing all the ObjToAnimate that will be used to shake our model
tolModes	float	tolerance applied to choose the modes
tolGIE	float	tolerance applied to calculated GIE
outputDir	str	absolute path to output directory in which all results will be stored
meshDir	str	absolute path to the directory containing the different mesh used by the Sofa scene in reduction
packageName	str	Which name will have the final component & package if the option is activated
toKeep	str	Indicate which Sofa.node to keep for our reduced component. by default will keep all the nodes used for animation
addToLib	Bool	If True will add in the python library of this plugin the finalized reduced component
verbose	Bool	display more or less verbose
addRigidBodyModes	list(int)	List of 3 of 0/1 that will allow translation along [x,y,z] axis of our reduced model
nbrCPU	int	Number of CPU we will use to generate/calculate the reduced model

setListSofaScene (*phasesToExecute=None*)

argument	type	definition
phases-ToExecute	list(int)	Allow to choose which phase to execute for the reduction by default will select all the phases

performReduction (*phasesToExecute=None, nbrOfModes=None*)

Perform all the steps of the reduction in one function

if you are sure of all the parameters this way is recommended to gain time

argument	type	definition
phasesToExecute	list(int)	Allow to choose which phase to execute for the reduction by default will select all the phase
nbrOfModes	int	Number of modes you want to keep by default will keep them all

phase1 (*phasesToExecute=None*)

The step will launch in parallel multiple Sofa scene (nbrCPU by nbrCPU number of scene) until it has run all the scene in the sequence. For that it will use the sofa launcher utility

argument	type	definition
phases-ToExecute	list(int)	Allow to choose which phase to execute for the reduction by default will select all the phase

What does it do to each scene:

- add animation to each actuators we want for our model in the predefined sequence
- add a writeState component to save the shaking resulting states
- take all the resulting states files and combines them in one file put in the debug dir with a debug scene

phase2 ()

With the previous result we compute the modes

it will set nbrOfModes to its maximum, but it can be changed has argument to the next step

phase3 (*phasesToExecute=None, nbrOfModes=None*)

The step will launch in parallel multiple Sofa scene (nbrCPU by nbrCPU number of scene) until it has run all the scene in the sequence. For that it will use the sofa launcher utility

argument	type	definition
phasesToExecute	list(int)	Allow to choose which phase to execute for the reduction by default will select all the phase
nbrOfModes	int	Number of modes you want to keep by default will keep them all

What does it do to each scene:

- **take the previous one and add the model order reduction component:**
 - HyperReducedFEMForceField
 - MappedMatrixForceFieldAndMas
 - ModelOrderReductionMapping
- produce an Hyper Reduced description of the model
- produce files listing the different element to keep
- produce also a state file of all the resulting states files & put in the debug

phase4 (nbrOfModes=None)

Final step :

- compute the RID and Weights
- compute the Active Nodes
- finalize the package
- add it to the plugin library if option activated

4.2.2 mor.script.morUtilityFunctions

Utilities functions used mainly by the **reduceModel** classes

```
mor.script.morUtilityFunctions.copy(src, dest)
mor.script.morUtilityFunctions.openDirName(infoStr)
mor.script.morUtilityFunctions.openFileName(infoStr)
mor.script.morUtilityFunctions.distance3D(vec1, vec2)
mor.script.morUtilityFunctions.update_progress(progress)
mor.script.morUtilityFunctions.errDif(G, xi, b)
mor.script.morUtilityFunctions.etaTild(Gtilde, b)
mor.script.morUtilityFunctions.selectECSW(G, b, tau, verbose)
```

```

mor.script.morUtilityFunctions.readStateFilesAndComputeModes (stateFilePath, tol,
                                                               modesFileName,
                                                               addRigidBody-
                                                               Modes=None,
                                                               verbose=False)

mor.script.morUtilityFunctions.readGieFileAndComputeRIDandWeights (gieFilename,
                                                               RIDFile-
                                                               Name,
                                                               weights-
                                                               FileName,
                                                               tol, ver-
                                                              bose=False)

mor.script.morUtilityFunctions.convertRIDinActiveNodes (RIDFileName, connectivity-
                                                               FileName, listActiveNodes-
                                                               FileName, verbose=False)

```

4.3 mor.wrapper

Set of class and functions to modify the SOFA scene during its construction or save it to make a reusable component

Content:

<code>mor.wrapper.morWrapper</code>	Wrapper used for MOR
<code>mor.wrapper.writeScene</code>	Set of functions to create a reusable SOFA component out of a SOFA scene

4.3.1 mor.wrapper.morWrapper

<code>mor.wrapper.MORWrapper</code>	MagicMock is a subclass of Mock with default implementations of most of the magic methods.
-------------------------------------	--

mor.wrapper.MORWrapper

`mor.wrapper.MORWrapper`

4.3.2 mor.wrapper.writeScene

<code>mor.wrapper.writeScene.</code>	Write a templated Header to the file <i>packageName</i>
<code>writeHeader(packageName)</code>	
<code>mor.wrapper.writeScene.</code>	With 2 lists describing the 2 Sofa.Node containing the
<code>writeGraphScene(...)</code>	components for our reduced model,
<code>mor.wrapper.writeScene.</code>	Write a templated Footer to the file <i>packageName</i>
<code>writeFooter(...)</code>	
<code>mor.wrapper.writeScene.</code>	According to the case it will add transla-
<code>buildArgStr(arg,[...])</code>	tion,rotation,scale arguments

mor.wrapper.writeScene.writeHeader

```
mor.wrapper.writeScene.writeHeader(packageName)
    Write a templated Header to the file packageName
```

Arg:

argument	type	definition
packageName	str	Name of the file were we will write (without any extension)

mor.wrapper.writeScene.writeGraphScene

```
mor.wrapper.writeScene.writeGraphScene(packageName, nodeName, myMORMModel, my-
                                         Model)
```

With 2 lists describing the 2 Sofa.Node containing the components for our reduced model,

this function will write each component with there initial parameters and clean or add parameters
in order to have in the end a reduced model component reusable as a function with arguments as :

```
def MyReducedModel ( attachedTo=None,
                      name="MyReducedModel",
                      rotation=[0.0, 0.0, 0.0],
                      translation=[0.0, 0.0, 0.0],
                      scale=[1.0, 1.0, 1.0],
                      surfaceMeshFileName=False,
                      surfaceColor=[1.0, 1.0, 1.0],
                      poissonRatio=None,
                      youngModulus=None,
                      totalMass=None)
```

Args:

argument	type	definition
packageName	str	Name of the file were we will write (without any extension)
nodeName	str	Name of the Sofa.Node we reduce
myMORMModel	list	list of tuple (solver_type , param_solver)
myModel	OrderedDict	Ordered dic containing has key Sofa.Node.name & has var a tuple of (Sofa_componant_type , param_solver)

mor.wrapper.writeScene.writeFooter

```
mor.wrapper.writeScene.writeFooter(packageName, nodeName, modelTransform)
    Write a templated Footer to the file packageName
```

Args:

argument	type	definition
packageName	str	Name of the file were we will write (without any extension)

mor.wrapper.writeScene.buildArgStrmor.wrapper.writeScene.**buildArgStr**(arg, translation=None)

According to the case it will add translation,rotation,scale arguments

Args:

argument	type	definition
arg	dic	Contains all argument of a Sofa Component
translation	float	Containis the initial translation of the model this will allow us to calculate a new position of an object depending of our reduced model by substracting our model relative origin make the TRS in the absolute origin and replace it in our model relative origin

COMPONENTS

5.1 Forcefields

5.1.1 HyperReducedTetrahedronFEMForceField

Defines

```
SIMPLEFEM_COLORMAP
SOFAHYPERREDUCEDTETRAHEDRONFEMFORCEFIELD_COLORMAP
namespace sofa

namespace sofacomponent

namespace sofa::componentforcefield

template <class DataTypes>
class sofa::component::forcefieldHyperReducedTetrahedronFEMForceField
Inherits from core::behavior::ForceField< DataTypes >, BaseRotationFinder
```

Per element (tetrahedron) data

```
typedef defaulttype::VecNoInit<12, Real> Displacement
typedef defaulttype::Mat<6, 6, Real> MaterialStiffness
typedef defaulttype::Mat<12, 6, Real> StrainDisplacement
typedef defaulttype::MatNoInit<3, 3, Real> Transformation
typedef defaulttype::Mat<12, 12, Real> StiffnessMatrix
typedef defaulttype::VecNoInit<6, Real> VoigtTensor
defaulttype::MatNoInit<3, 3, Real> R0
```

Full system matrix assembly support

```
typedef std::pair<int, Real> Col_Value
typedef helper::vector<Col_Value> CompressedValue
```

```
typedef helper::vector<CompressedValue> CompressedMatrix  
CompressedMatrix _stiffnesses
```

Plasticity such as “Interactive Virtual Materials”, Muller & Gross, GI 2004

```
Data<Real> _plasticMaxThreshold  
Data<Real> _plasticYieldThreshold  
Data<Real> _plasticCreep
```

Public Types

```
enum [anonymous]::component::forcefield __anonymous0  
Values:  
sofa::component::forcefieldSMALL = 0  
sofa::component::forcefieldLARGE = 1  
sofa::component::forcefieldPOLAR = 2  
sofa::component::forcefieldSVD = 3  
typedef core::behavior::ForceField<DataTypes> InheritForceField  
typedef DataTypes::VecCoord VecCoord  
typedef DataTypes::VecDeriv VecDeriv  
typedef DataTypes::VecReal VecReal  
typedef VecCoord Vector  
typedef DataTypes::Coord Coord  
typedef DataTypes::Deriv Deriv  
typedef Coord::value_type Real  
typedef core::objectmodel::Data<VecDeriv> DataVecDeriv  
typedef core::objectmodel::Data<VecCoord> DataVecCoord  
typedef core::topology::BaseMeshTopology::index_type Index  
typedef core::topology::BaseMeshTopology::Tetra Element  
typedef core::topology::BaseMeshTopology::SeqTetrahedra VecElement  
typedef core::topology::BaseMeshTopology::Tetrahedron Tetrahedron
```

Public Functions

```
SOFA_CLASS2 (SOFA_TEMPLATE) HyperReducedTetrahedronFEMForceField,  
           DataTypes  
           ,  
           SOFA_TEMPLATEcore::behavior::ForceField,  
           core::behavior::BaseRotationFinder  
           DataTypes,  
           Real getRestVolume ()  
           void getRotation (Transformation &R, unsigned int nodeIdx)
```

```

void getRotations (VecReal &vecR)
void getRotations (defaulttype::BaseMatrix *rotations, int offset = 0)
void setPoissonRatio (Real val)
void setYoungModulus (Real val)
void setComputeGlobalMatrix (bool val)
Transformation getActualTetraRotation (unsigned int index)
Transformation getInitialTetraRotation (unsigned int index)
void setMethod (std::string methodName)
void setMethod (int val)
void setUpdteStiffnessMatrix (bool val)
virtual void reset ()
virtual void init ()
virtual void reinit ()
virtual void addForce (const core::MechanicalParams *mparams, DataVecDeriv
&d_f, const DataVecCoord &d_x, const DataVecDeriv
&d_v)
virtual void addDForce (const core::MechanicalParams *mparams, DataVecDeriv
&d_df, const DataVecDeriv &d_dx)
virtual SReal getPotentialEnergy (const core::MechanicalParams *, const
DataVecCoord &x) const
virtual void addKToMatrix (sofa::defaulttype::BaseMatrix *m, SReal kFactor, un-
signed int &offset)
virtual void addKToMatrix (const core::MechanicalParams *, const
sofa::core::behavior::MultiMatrixAccessor *)
virtual void addSubKToMatrix (sofa::defaulttype::BaseMatrix *mat, const
helper::vector<unsigned> &subMatrixIndex, SReal k,
unsigned int &offset)
void draw (const core::visual::VisualParams *vparams)
void getElementStiffnessMatrix (Real *stiffness, unsigned int nodeIdx)
void getElementStiffnessMatrix (Real *stiffness, Tetrahedron &te)
virtual void computeMaterialStiffness (MaterialStiffness &materialMatrix, In-
dex &a, Index &b, Index &c, Index &d)

```

Public Members

```

Data<VecCoord> _initialPoints
int method
Data<std::string> f_method
Data<Real> _poissonRatio
Data<VecReal> _youngModulus
Data<VecReal> _localStiffnessFactor

```

```
Data<bool> _updateStiffnessMatrix
Data<bool> _assembling
Data<sofa::helper::OptionsGroup> _gatherPt
Data<sofa::helper::OptionsGroup> _gatherBsize
Data<bool> drawHeterogeneousTetra
Data<bool> drawAsEdges
Data<bool> d_prepareECSW
Data<unsigned int> d_nbModes
Data<unsigned int> d_nbTrainingSet
Data<unsigned int> d_periodSaveGIE
Data<bool> d_performECSW
Data<std::string> d_modesPath
Data<std::string> d_RIDPath
Data<std::string> d_weightsPath
Real minYoung
Real maxYoung
helper::vector<Real> elemLambda
helper::vector<Real> elemMu
helper::vector<Mat44> elemShapeFun
Real prevMaxStress
Data<int> _computeVonMisesStress
Data<helper::vector<Real>> _vonMisesPerElement
Data<helper::vector<Real>> _vonMisesPerNode
Data<helper::vector<defaulttype::Vec4f>> _vonMisesStressColors
helper::ColorMap m_VonMisesColorMap
Data<std::string> _showStressColorMap
Data<float> _showStressAlpha
Data<bool> _showVonMisesStressPerNode
Data<bool> isToPrint
Data<bool> _updateStiffness
helper::vector<defaulttype::Vec<6, Real>> elemDisplacements
bool updateVonMisesStress
```

Protected Types

```
typedef helper::vector<MaterialStiffness> VecMaterialStiffness
typedef helper::vector<StrainDisplacement> VecStrainDisplacement
typedef defaulttype::Mat<4, 4, Real> Mat44
typedef defaulttype::Mat<3, 3, Real> Mat33
typedef defaulttype::Mat<4, 3, Real> Mat43
```

Protected Functions

```
HyperReducedTetrahedronFEMForceField()
virtual ~HyperReducedTetrahedronFEMForceField()
void computeStrainDisplacement (StrainDisplacement &J, Coord a, Coord b, Coord c, Coord d)
Real pseudo_determinant_for_coef (const defaulttype::Mat<2, 3, Real> &M)
void computeStiffnessMatrix (StiffnessMatrix &S, StiffnessMatrix &SR, const MaterialStiffness &K, const StrainDisplacement &J, const Transformation &Rot)
virtual void computeMaterialStiffness (int i, Index &a, Index &b, Index &c, Index &d)
void computeForce (Displacement &F, const Displacement &Depl, VoigtTensor &plasticStrain, const MaterialStiffness &K, const StrainDisplacement &J)
void computeForce (Displacement &F, const Displacement &Depl, const MaterialStiffness &K, const StrainDisplacement &J, SReal fact)
void initSmall (int i, Index &a, Index &b, Index &c, Index &d)
void accumulateForceSmall (Vector &f, const Vector &p, typename VecElement::const_iterator elementIt, Index elementIndex)
void applyStiffnessSmall (Vector &f, const Vector &x, int i = 0, Index a = 0, Index b = 1, Index c = 2, Index d = 3, SReal fact = 1.0)
void initLarge (int i, Index &a, Index &b, Index &c, Index &d)
void computeRotationLarge (Transformation &r, const Vector &p, const Index &a, const Index &b, const Index &c)
void accumulateForceLarge (Vector &f, const Vector &p, typename VecElement::const_iterator elementIt, Index elementIndex)
void initPolar (int i, Index &a, Index &b, Index &c, Index &d)
void accumulateForcePolar (Vector &f, const Vector &p, typename VecElement::const_iterator elementIt, Index elementIndex)
void initSVD (int i, Index &a, Index &b, Index &c, Index &d)
void accumulateForceSVD (Vector &f, const Vector &p, typename VecElement::const_iterator elementIt, Index elementIndex)
void applyStiffnessCorotational (Vector &f, const Vector &x, int i = 0, Index a = 0, Index b = 1, Index c = 2, Index d = 3, SReal fact = 1.0)
```

```
void handleTopologyChange ()  
void computeVonMisesStress ()  
void handleEvent (core::objectmodel::Event *event)
```

Protected Attributes

```
VecMaterialStiffness materialsStiffnesses  
VecStrainDisplacement strainDisplacements  
helper::vector<Transformation> rotations  
helper::vector<VoigtTensor> _plasticStrains  
SReal m_potentialEnergy  
core::topology::BaseMeshTopology *_mesh  
const VecElement *_indexedElements  
bool needUpdateTopology  
HyperReducedTetrahedronFEMForceFieldInternalData<DataTypes> data  
Real m_restVolume  
Eigen::MatrixXd m_modes  
std::vector<std::vector<double>> Gie  
Eigen::VectorXd weights  
Eigen::VectorXi reducedIntegrationDomain  
unsigned int m_RIDsize  
helper::vector<helper::fixed_array<Coord, 4>> _rotatedInitialElements  
helper::vector<Transformation> _initialRotations  
helper::vector<unsigned int> _rotationIdx  
helper::vector<Transformation> _initialTransformation
```

Friends

```
friend sofa::component::forcefield::HyperReducedTetrahedronFEMForceFieldInter  
template <class DataTypes>  
class sofa::component::forcefield::HyperReducedTetrahedronFEMForceFieldInternalData
```

Public Types

```
typedef HyperReducedTetrahedronFEMForceField<DataTypes> Main
```

Public Functions

```
void initPtrData (Main *m)
```

5.1.2 HyperReducedTriangleFEMForceField

```

namespace sofa

namespace sofacomponent

namespace sofa::componentforcefield

template <class DataTypes>
class sofa::component::forcefieldHyperReducedTriangleFEMForceField
    Inherits from core::behavior::ForceField< DataTypes >

```

Public Types

```

typedef core::behavior::ForceField<DataTypes> Inherited
typedef DataTypes::VecCoord VecCoord
typedef DataTypes::VecDeriv VecDeriv
typedef DataTypes::Coord Coord
typedef DataTypes::Deriv Deriv
typedef Coord::value_type Real
typedef core::objectmodel::Data<VecCoord> DataVecCoord
typedef core::objectmodel::Data<VecDeriv> DataVecDeriv
typedef sofa::core::topology::BaseMeshTopology::index_type Index
typedef sofa::core::topology::BaseMeshTopology::Triangle Element
typedef sofa::core::topology::BaseMeshTopology::SeqTriangles VecElement

```

Public Functions

```

SOFA_CLASS (SOFA_TEMPLATE) HyperReducedTriangleFEMForceField, DataTypes
    , SOFA_TEMPLATEcore::behavior::ForceField, DataTypes

virtual void init ()
virtual void reinit ()
virtual void addForce (const core::MechanicalParams *mparams, DataVecDeriv &f,
    const DataVecCoord &x, const DataVecDeriv &v)
virtual void addDForce (const core::MechanicalParams *mparams, DataVecDeriv
    &df, const DataVecDeriv &dx)
virtual SReal getPotentialEnergy (const core::MechanicalParams *, const
    DataVecCoord&) const
void draw (const core::visual::VisualParams *vparams)
Real getPoisson ()
void setPoisson (Real val)
Real getYoung ()

```

```
void setYoung (Real val)
int getMethod ()
void setMethod (int val)
```

Public Members

```
int method
Data<std::string> f_method
Data<Real> f_poisson
Data<Real> f_young
Data<Real> f_thickness
Data<bool> f_planeStrain
Data<bool> d_prepareECSW
Data<unsigned int> d_nbModes
Data<unsigned int> d_nbTrainingSet
Data<unsigned int> d_periodSaveGIE
Data<bool> d_performECSW
Data<std::string> d_modesPath
Data<std::string> d RIDPath
Data<std::string> d_weightsPath
```

Public Static Attributes

```
const int SMALL = 1
const int LARGE = 0
```

Protected Types

```
typedef defaulttype::Vec<6, Real> Displacement
typedef defaulttype::Mat<3, 3, Real> MaterialStiffness
typedef sofa::helper::vector<MaterialStiffness> VecMaterialStiffness
typedef defaulttype::Mat<6, 3, Real> StrainDisplacement
typedef sofa::helper::vector<StrainDisplacement> VecStrainDisplacement
typedef defaulttype::Mat<3, 3, Real> Transformation
typedef defaulttype::Mat<9, 9, Real> StiffnessMatrix
```

Protected Functions

```

HyperReducedTriangleFEMForceField()
virtual ~HyperReducedTriangleFEMForceField()
virtual void applyStiffness (VecCoord &f, Real h, const VecCoord &x, const SReal &kFactor)
void computeStrainDisplacement (StrainDisplacement &J, Coord a, Coord b, Coord c)
void computeMaterialStiffnesses ()
void computeForce (Displacement &F, const Displacement &Depl, const MaterialStiffness &K, const StrainDisplacement &J)
void initSmall ()
void accumulateForceSmall (VecCoord &f, const VecCoord &p, Index elementIndex, bool implicit = false)
void applyStiffnessSmall (VecCoord &f, Real h, const VecCoord &x, const SReal &kFactor)
void initLarge ()
void computeRotationLarge (Transformation &r, const VecCoord &p, const Index &a, const Index &b, const Index &c)
void accumulateForceLarge (VecCoord &f, const VecCoord &p, Index elementIndex, bool implicit = false)
void applyStiffnessLarge (VecCoord &f, Real h, const VecCoord &x, const SReal &kFactor)
void computeElementStiffnessMatrix (StiffnessMatrix &S, StiffnessMatrix &SR, const MaterialStiffness &K, const StrainDisplacement &J, const Transformation &Rot)
void addKToMatrix (sofa::defaulttype::BaseMatrix *mat, SReal k, unsigned int &offset)

```

Protected Attributes

```

VecMaterialStiffness _materialsStiffnesses
VecStrainDisplacement _strainDisplacements
sofa::core::topology::BaseMeshTopology *_mesh
const VecElement *_indexedElements
Data<VecCoord> _initialPoints
Eigen::MatrixXd m_modes
std::vector<std::vector<double>> Gie
std::vector<double> b_ECSW
Eigen::VectorXd weights
Eigen::VectorXi reducedIntegrationDomain
unsigned int m_RIDsize

```

```
sofa::helper::vector<helper::fixed_array<Coord, 3>> _rotatedInitialElements  
sofa::helper::vector<Transformation> _rotations
```

5.1.3 HyperReducedTetrahedronHyperelasticityFEMForceField

```
namespace sofa
```

```
namespace sofacomponent
```

```
namespace sofa::componentforcefield
```

```
template <class DataTypes>  
class sofa::component::forcefieldHyperReducedTetrahedronHyperelasticityFEMForceField  
Inherits from core::behavior::ForceField< DataTypes >
```

Public Types

```
typedef core::behavior::ForceField<DataTypes> Inherited  
typedef DataTypes::VecCoord VecCoord  
typedef DataTypes::VecDeriv VecDeriv  
typedef DataTypes::Coord Coord  
typedef DataTypes::Deriv Deriv  
typedef Coord::value_type Real  
typedef core::objectmodel::Data<VecDeriv> DataVecDeriv  
typedef core::objectmodel::Data<VecCoord> DataVecCoord  
typedef Mat<3, 3, Real> Matrix3  
typedef MatSym<3, Real> MatrixSym  
typedef std::pair<MatrixSym, MatrixSym> MatrixPair  
typedef std::pair<Real, MatrixSym> MatrixCoeffPair  
typedef helper::vector<Real> SetParameterArray  
typedef helper::vector<Coord> SetAnisotropyDirectionArray  
typedef core::topology::BaseMeshTopology::index_type Index  
typedef core::topology::BaseMeshTopology::Tetra Element  
typedef core::topology::BaseMeshTopology::SeqTetrahedra VecElement  
typedef sofa::core::topology::Topology::Tetrahedron Tetrahedron  
typedef sofa::core::topology::Topology::TetraID TetraID  
typedef sofa::core::topology::Topology::Tetra Tetra  
typedef sofa::core::topology::Topology::Edge Edge  
typedef sofa::core::topology::BaseMeshTopology::EdgesInTriangle EdgesInTriangle
```

```
typedef sofa::core::topology::BaseMeshTopology::EdgesInTetrahedron EdgesInTetrahedron
typedef sofa::core::topology::BaseMeshTopology::TrianglesInTetrahedron TrianglesInTetrahedron
```

Public Functions

```
SOFA_CLASS (SOFA_TEMPLATE) HyperReducedTetrahedronHyperelasticityFEMForceField,
    DataTypes
    , SOFA_TEMPLATEcore::behavior::ForceField, DataTypes

void setMaterialName (const string name)
void setParameter (const vector<Real> param)
void setDirection (const vector<Coord> direction)
virtual void init ()

virtual void addForce (const core::MechanicalParams *mparams, DataVecDeriv
    &d_f, const DataVecCoord &d_x, const DataVecDeriv
    &d_v)
virtual void addDForce (const core::MechanicalParams *mparams, DataVecDeriv
    &d_df, const DataVecDeriv &d_dx)
virtual SReal getPotentialEnergy (const core::MechanicalParams *, const
    DataVecCoord&) const
virtual void addKToMatrix (sofa::defaulttype::BaseMatrix *mat, SReal k, unsigned int
    &offset)
void draw (const core::visual::VisualParams *vparams)
Mat<3, 3, double> getPhi (int tetrahedronIndex)
```

Public Members

```
sofa::component::fem::MaterialParameters<DataTypes> globalParameters
```

Protected Functions

```
HyperReducedTetrahedronHyperelasticityFEMForceField()
virtual ~HyperReducedTetrahedronHyperelasticityFEMForceField()
void testDerivatives ()
void saveMesh (const char *filename)
void updateTangentMatrix ()
```

Protected Attributes

```
core::topology::BaseMeshTopology *m_topology
VecCoord m_initialPoints
bool m_updateMatrix
bool m_meshSaved
```

```
Data<bool> d_stiffnessMatrixRegularizationWeight
Data<string> d_materialName
Data<SetParameterArray> d_parameterSet
Data<SetAnisotropyDirectionArray> d_anisotropySet
Data<bool> d_prepareECSW
Data<unsigned int> d_nbModes
Data<unsigned int> d_nbTrainingSet
Data<unsigned int> d_periodSaveGIE
Data<bool> d_performECSW
Data<std::string> d_modesPath
Data<std::string> d RIDPath
Data<std::string> d_weightsPath
Eigen::MatrixXd m_modes
std::vector<std::vector<double>> Gie
Eigen::VectorXd weights
Eigen::VectorXi reducedIntegrationDomain
Eigen::VectorXi reducedIntegrationDomainWithEdges
unsigned int m RIDsize
unsigned int m RIDedgeSize
TetrahedronData<sofa::helper::vector<TetrahedronRestInformation>> m_tetrahedronInfo
EdgeData<sofa::helper::vector<EdgeInformation>> m_edgeInfo
fem::HyperelasticMaterial<DataTypes> *m_myMaterial
TetrahedronHandler *m_tetrahedronHandler
class sofa::component::forcefield::HyperReducedTetrahedronHyperelasticityFEMForceFieldEdgeInformation
```

Public Functions

EdgeInformation ()

Public Members

Matrix3 **DfDx**

Friends

ostream &**operator<<** (ostream &*os*, **const EdgeInformation&**)

istream &**operator>>** (istream &*in*, **EdgeInformation&**)

```
class sofa::component::forcefield::HyperReducedTetrahedronHyperelasticityFEMForceFieldMatrixList
```

Public Members

Matrix3 **data**[6]

```
class sofa::component::forcefield::HyperReducedTetrahedronHyperelasticityFEMForceFieldTetrahedronHandler
Inherits from TopologyDataHandler< Tetrahedron, sofa::helper::vector< TetrahedronRestInformation >>
```

Public Types

```
typedef HyperReducedTetrahedronHyperelasticityFEMForceField<DataTypes>::TetrahedronRestInformation
```

Public Functions

```
TetrahedronHandler (HyperReducedTetrahedronHyperelasticityFEMForceField<DataTypes>
*ff, TetrahedronData<sofa::helper::vector<TetrahedronRestInformation>>
*data)
void applyCreateFunction (unsigned int, TetrahedronRestInformation &t, const
Tetrahedron&, const sofa::helper::vector<unsigned
int>&, const sofa::helper::vector<double>&)
```

Protected Attributes

HyperReducedTetrahedronHyperelasticityFEMForceField<DataTypes> ***ff**

```
class sofa::component::forcefield::HyperReducedTetrahedronHyperelasticityFEMForceFieldTetrahedronRestInformation
Inherits from fem::StrainInformation< DataTypes >
```

Public Functions

```
TetrahedronRestInformation()
```

Public Members

Coord **m_shapeVector**[4]

Coord **m_fiberDirection**

Real **m_restVolume**

Real **m_volScale**

Real **m_volume**

MatrixSym **m_SPKTensorGeneral**

Matrix3 **m_deformationGradient**

Real **m_strainEnergy**

Friends

```
ostream &operator<< (ostream &os, const TetrahedronRestInformation&)
istream &operator>> (istream &in, TetrahedronRestInformation&)
```

5.1.4 MappedMatrixForceFieldAndMassMOR

```
namespace sofa

namespace sofacomponent

namespace sofa::componentinteractionforcefield

template <typename TDataTypes1, typename TDataTypes2>
class sofa::component::interactionforcefieldMappedMatrixForceFieldAndMassMOR
Inherits from MappedMatrixForceFieldAndMass< TDataTypes1, TDataTypes2 >
```

Public Types

```
typedef MappedMatrixForceFieldAndMass<TDataTypes1, TDataTypes2> Inherit
```

Public Functions

```
SOFA_CLASS (SOFA_TEMPLATE2) MappedMatrixForceFieldAndMassMOR,
TDataTypes1, TDataTypes2
, SOFA_TEMPLATE2MappedMatrixForceFieldAndMass, TDataTypes1, TDataTypes2
virtual void init ()
```

Public Members

```
Data<bool> performECSW
sofa::helper::vector<unsigned int> listActiveNodes
sofa::core::objectmodel::DataFileName listActiveNodesPath
Data<bool> timeInvariantMapping1
Data<bool> timeInvariantMapping2
Eigen::SparseMatrix<double> constantJ1
Eigen::SparseMatrix<double> constantJ2
Data<bool> saveReducedMass
Data<bool> usePrecomputedMass
sofa::core::objectmodel::DataFileName precomputedMassPath
Eigen::SparseMatrix<double> JtMJ
```

Protected Functions

```

MappedMatrixForceFieldAndMassMOR ()

virtual void accumulateJacobiansOptimized (const MechanicalParams *mparams)
virtual void addMassToSystem (const MechanicalParams *mparams, const DefaultMultiMatrixAccessor *KAccessor)
virtual void addPrecomputedMassToSystem (const MechanicalParams *mparams, const unsigned int mstateSize, const Eigen::SparseMatrix<double> &Jeig, Eigen::SparseMatrix<double> &JtK-Jeig)
virtual void buildIdentityBlocksInJacobian (core::behavior::BaseMechanicalState *mstate, sofa::core::MatrixDerivId Id)
virtual void optimizeAndCopyMappingJacobianToEigenFormat1 (const typename TDataTypes1::MatrixDeriv &J, Eigen::SparseMatrix<double> &Jeig)
virtual void optimizeAndCopyMappingJacobianToEigenFormat2 (const typename TDataTypes2::MatrixDeriv &J, Eigen::SparseMatrix<double> &Jeig)

```

5.1.5 MooneyRivlin

```

namespace sofa

namespace sofacomponent

namespace sofa::componentfem

template <class DataTypes>
class sofa::component::femMooneyRivlin
    Inherits from HyperelasticMaterial< DataTypes >

```

Private Types

```

typedef DataTypes::Coord::value_type Real
typedef defaulttype::Mat<3, 3, Real> Matrix3
typedef defaulttype::Mat<6, 6, Real> Matrix6
typedef defaulttype::MatSym<3, Real> MatrixSym

```

Private Functions

```
virtual Real getStrainEnergy (StrainInformation<DataTypes> *sinfo, const MaterialParameters<DataTypes> &param)
virtual void deriveSPKTensor (StrainInformation<DataTypes> *sinfo, const MaterialParameters<DataTypes> &param, MatrixSym &SPKTensorGeneral)
virtual void applyElasticityTensor (StrainInformation<DataTypes> *sinfo, const MaterialParameters<DataTypes> &param, const MatrixSym &inputTensor, MatrixSym &outputTensor)
virtual void ElasticityTensor (StrainInformation<DataTypes> *sinfo, const MaterialParameters<DataTypes> &param, Matrix6 &outputTensor)
```

5.1.6 NeoHookean

```
namespace sofa

namespace sofacomponent

namespace sofa::componentfem

template <class DataTypes>
class sofa::component::femNeoHookean
Inherits from HyperelasticMaterial<DataTypes>
```

Private Types

```
typedef DataTypes::Coord::value_type Real
typedef defaulttype::Mat<3, 3, Real> Matrix3
typedef defaulttype::Mat<6, 6, Real> Matrix6
typedef defaulttype::MatSym<3, Real> MatrixSym
```

Private Functions

```
virtual Real getStrainEnergy (StrainInformation<DataTypes> *sinfo, const MaterialParameters<DataTypes> &param)
virtual void deriveSPKTensor (StrainInformation<DataTypes> *sinfo, const MaterialParameters<DataTypes> &param, MatrixSym &SPKTensorGeneral)
virtual void applyElasticityTensor (StrainInformation<DataTypes> *sinfo, const MaterialParameters<DataTypes> &param, const MatrixSym &inputTensor, MatrixSym &outputTensor)
```

```
virtual void ElasticityTensor(StrainInformation<DataTypes> *sinfo, const
                               MaterialParameters<DataTypes> &param, Matrix6
                               &outputTensor)
```

5.2 Loaders

```
namespace sofa

namespace sofacomponent

namespace sofa::componentloader

template <class EigenMatrixType>
class sofa::component::loaderMatrixLoader
```

Public Functions

```
MatrixLoader()
virtual ~MatrixLoader()
void load()
void getMatrix(EigenMatrixType &matrix)
void setFileName(std::string fileName)
```

Protected Attributes

```
unsigned int m_nbRows
unsigned int m_nbColumns
std::string m_fileName
EigenMatrixType m_matrix
```

5.3 Mappings

```
namespace sofa

namespace sofacomponent

namespace sofa::componentmapping

template <class TIn, class TOut>
class sofa::component::mappingModelOrderReductionMapping
#include <ModelOrderReductionMapping.h> Component mapping state variables onto precomputed global basis vectors (aka: modes).
```

This maps vec1d MechanicalObjects to vec3d MechanicalObjects. Inputs: modesPath: Path to the text file containing the precomputed modes RIDpath: In case perfromECSW is true, path to the text file containing the reduced integration domain indices. The size of the “position” attribute will fix the number of modes used for the mapping

Inherits from core::Mapping< TIn, TOut >

Public Types

```
enum [anonymous]:component::mapping__anonymous0
Values:
sofa::component::mappingN = OutDataTypes::spatial_dimensions

enum [anonymous]:component::mapping__anonymous1
Values:
sofa::component::mappingNIn = sofa::defaulttype::DataTypeInfo<InDeriv>::Size

enum [anonymous]:component::mapping__anonymous2
Values:
sofa::component::mappingNOut = sofa::defaulttype::DataTypeInfo<Deriv>::Size

typedef core::Mapping<TIn, TOut> Inherit
typedef TIn In
typedef TOut Out
typedef In::Real InReal
typedef In::VecCoord InVecCoord
typedef In::VecDeriv InVecDeriv
typedef In::Coord InCoord
typedef In::Deriv InDeriv
typedef In::MatrixDeriv InMatrixDeriv
typedef Out::VecCoord VecCoord
typedef Out::VecDeriv VecDeriv
typedef Out::Coord Coord
typedef Out::Deriv Deriv
typedef Out::MatrixDeriv MatrixDeriv
typedef Out OutDataTypes
typedef OutDataTypes::Real OutReal
typedef OutDataTypes::VecCoord OutVecCoord
typedef OutDataTypes::VecDeriv OutVecDeriv
typedef Inherit::ForceMask ForceMask
typedef core::topology::BaseMeshTopology::index_type Index
typedef core::topology::BaseMeshTopology::Tetra Element
typedef core::topology::BaseMeshTopology::SeqTetrahedra VecElement
```

```
typedef linear solver::EigenSparseMatrix<TIn, TOut> eigen_type
typedef helper::vector<default type::BaseMatrix *> js_type
typedef default type::Mat<N, N, InReal> Mat
```

Public Functions

```
SOFA_CLASS (SOFA_TEMPLATE2) ModelOrderReductionMapping, TIn, TOut
, SOFA_TEMPLATE2core::Mapping, TIn, TOut

virtual bool sameTopology () const
    Return true if the destination model has the same topology as the source model.

    This is the case for mapping keeping a one-to-one correspondance between input and output
    DOFs (mostly identity or data-conversion mappings).

void init ()

void apply (const core::MechanicalParams *mparams, Data<VecCoord> &out, const
    Data<InVecCoord> &in)
void applyJ (const core::MechanicalParams *mparams, Data<VecDeriv> &out, const
    Data<InVecDeriv> &in)
void applyJT (const core::MechanicalParams *mparams, Data<InVecDeriv> &out,
    const Data<VecDeriv> &in)
void applyJTT (const core::ConstraintParams *cparams, Data<InMatrixDeriv> &out,
    const Data<MatrixDeriv> &in)
const sofa::default type::BaseMatrix *getJ ()
const js_type *getJs ()
```

Public Members

sofa::core::objectmodel::DataFileName **d_modesPath**

Protected Functions

```
ModelOrderReductionMapping ()
virtual ~ModelOrderReductionMapping ()
```

Protected Attributes

```
Eigen::MatrixXd m_modesEigen
Eigen::MatrixXi m_listActiveNodes
eigen_type m_J
js_type m_js
```

**CHAPTER
SIX**

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

m

`mor`, 21
`mor.animation`, 21
`mor.script`, 22
`mor.script.morUtilityFunctions`, 28
`mor.wrapper`, 29

INDEX

A

addParamWrapper() (mor.script.reduceModel.ReductionParamor (module), 21
method), 25
addToLib() (mor.script.reduceModel.PackageBuilder
method), 25

B

buildArgStr() (in module mor.wrapper.writeScene), 31

C

checkExistance() (mor.script.reduceModel.PackageBuilder
method), 24
checkNodeNbr() (mor.script.reduceModel.PackageBuilder
method), 25
cleanStateFile() (mor.script.reduceModel.PackageBuilder
method), 25
convertRIDInActiveNodes() (in module
mor.script.morUtilityFunctions), 29
copy() (in module mor.script.morUtilityFunctions), 28
copy() (mor.script.reduceModel.PackageBuilder
method), 24
copyAndCleanState() (mor.script.reduceModel.PackageBuilder
method), 25
copyFileIntoAnother() (mor.script.reduceModel.PackageBuilder
method), 25

D

defaultShaking() (in module mor.animation), 22
distance3D() (in module mor.script.morUtilityFunctions),
28

E

errDif() (in module mor.script.morUtilityFunctions), 28
etaTild() (in module mor.script.morUtilityFunctions), 28

F

finalizePackage() (mor.script.reduceModel.PackageBuilder
method), 25

G

generateListOfPhase() (mor.script.reduceModel.ReductionParam (class in mor.script.reduceModel), 25
method), 24

M

mor (module), 21
mor.animation (module), 21
mor.script (module), 22
mor.script.morUtilityFunctions (module), 28
mor.wrapper (module), 29
MORWrapper (in module mor.wrapper), 29

O

ObjToAnimate (class in mor.script.reduceModel), 23
openDirName() (in module
mor.script.morUtilityFunctions), 28
openFileName() (in module
mor.script.morUtilityFunctions), 28

P

PackageBuilder (class in mor.script.reduceModel), 24
performReduction() (mor.script.reduceModel.ReduceModel
method), 26
phase1() (mor.script.reduceModel.ReduceModel
method), 27
phase2() (mor.script.reduceModel.ReduceModel
method), 27
phase3() (mor.script.reduceModel.ReduceModel
method), 27
phase4() (mor.script.reduceModel.ReduceModel
method), 28

R

readGieFileAndComputeRIDandWeights() (in module
mor.script.morUtilityFunctions), 29
readStateFilesAndComputeModes() (in module
mor.script.morUtilityFunctions), 28
ReduceModel (class in mor.script.reduceModel), 25
ReductionAnimations (class in mor.script.reduceModel),
24
ReductionParam (class in mor.script.reduceModel), 25

S

selectECSW() (in module
mor.script.morUtilityFunctions), 28

setFileName() (mor.script.reduceModel.ReductionParam
method), [25](#)
setListSofaScene() (mor.script.reduceModel.ReduceModel
method), [26](#)
setNbIteration() (mor.script.reduceModel.ReductionAnimations
method), [24](#)
setNbTrainingSet() (mor.script.reduceModel.ReductionParam
method), [25](#)
shakingInverse() (in module mor.animation), [22](#)
shakingSofia() (in module mor.animation), [22](#)

U

update_progress() (in module
mor.script.morUtilityFunctions), [28](#)

W

writeFooter() (in module mor.wrapper.writeScene), [30](#)
writeGraphScene() (in module mor.wrapper.writeScene),
[30](#)
writeHeader() (in module mor.wrapper.writeScene), [30](#)