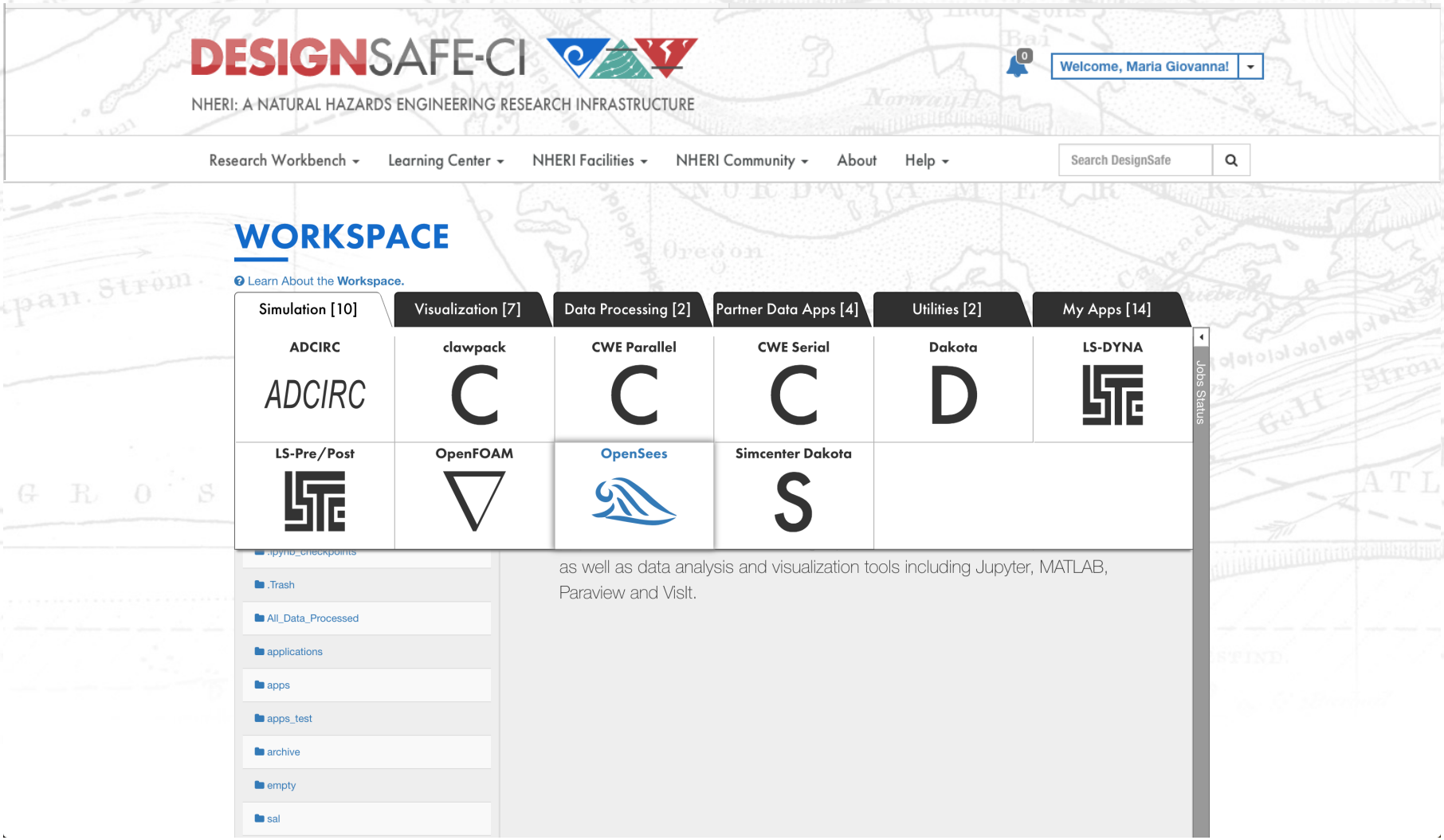
# *OpenSees & DesignSafe:* OpenSeesMP

November 2018

Maria Giovanna Durante, Ph.D.
Postdoctoral Research Fellow
University of Texas at Austin
mgdurante@utexas.edu

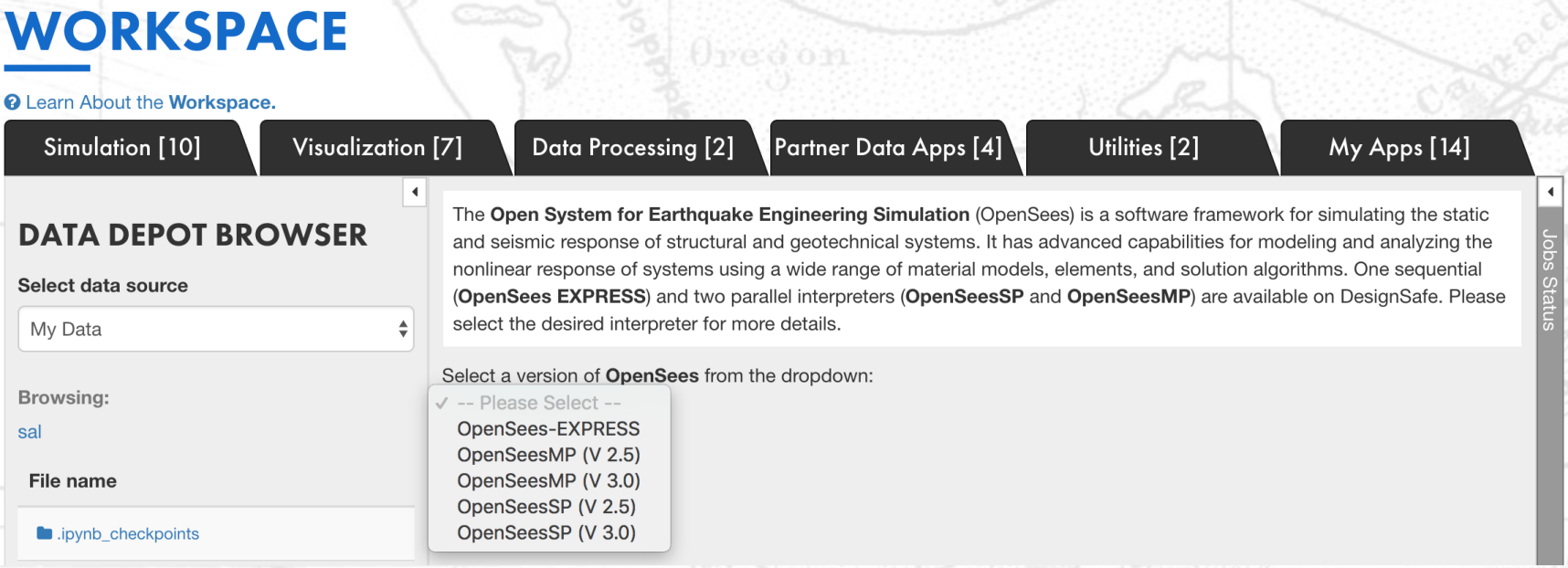
DESIGNSAFE-CI A NATURAL HAZARDS ENGINEERING COMMUNITY
TEXAS The University of Texas at Austin
TACC RICE Florida Tech

# OpenSees applications on DesignSafe

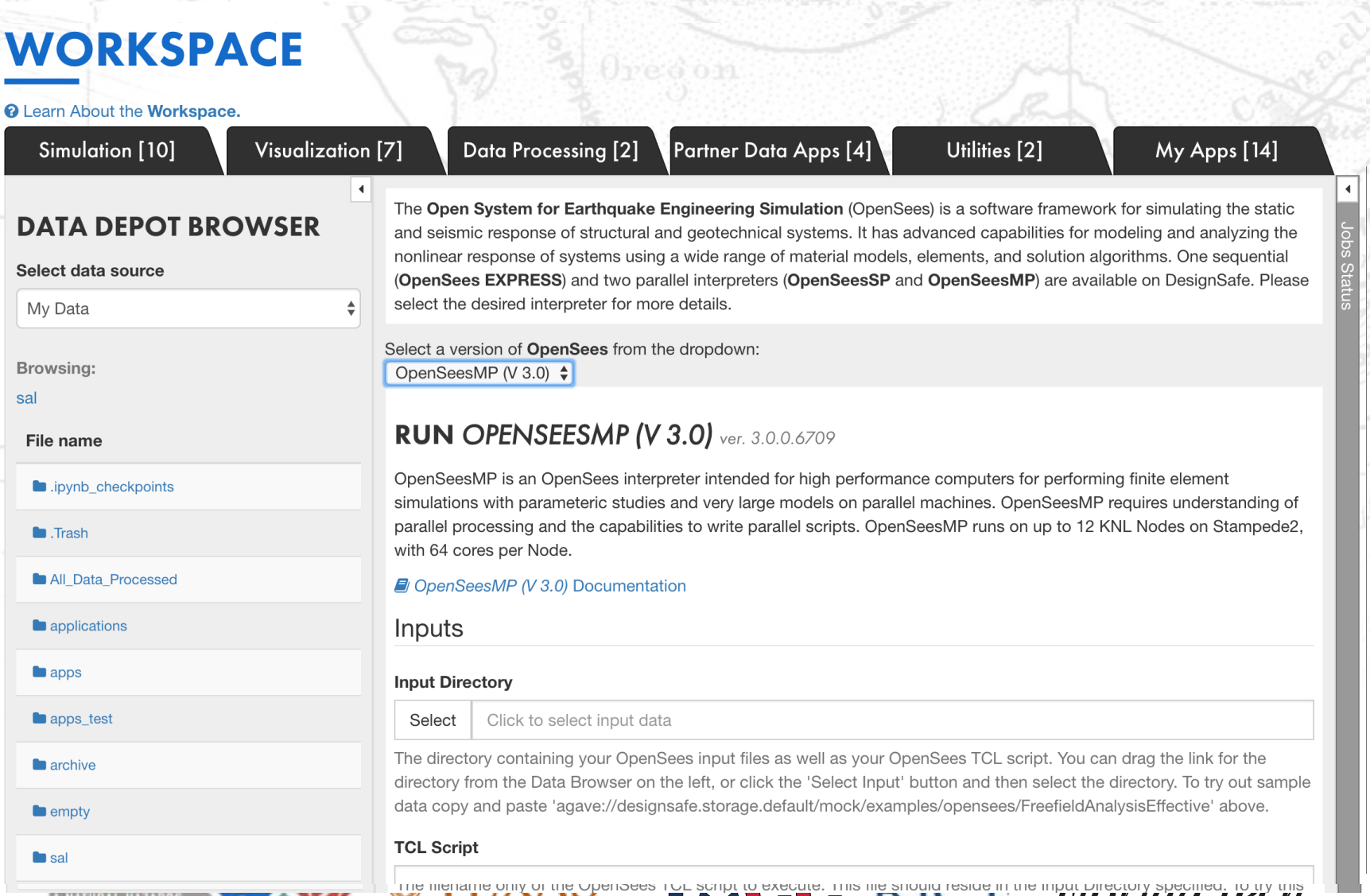



DESIGNSAFE-CI A NATURAL HAZARDS ENGINEERING COMMUNITY · TEXAS The University of Texas at Austin · TACC · RICE · Florida Tech

# OpenSees applications on DesignSafe



# OpenSees applications on DesignSafe




A NATURAL HAZARDS ENGINEERING COMMUNITY
The University of Texas at Austin
Florida Tech

# *Why OpenSeesMP?*

OpenSeesMP (V 2.5)

OpenSeesMP (V 3.0)

OpenSeesMP is specifically developed for HPC.

**Pros:**

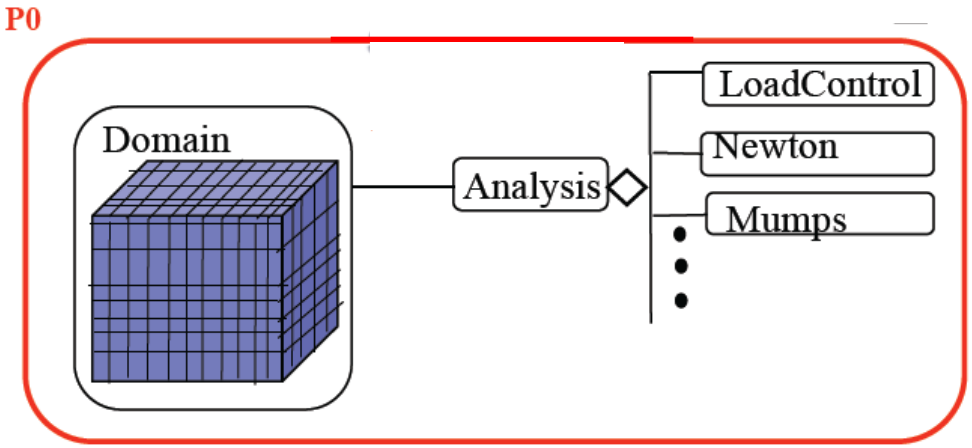
- Ideal for very large models and parametric studies;
- More control on the parallelization process;

**Cons:**

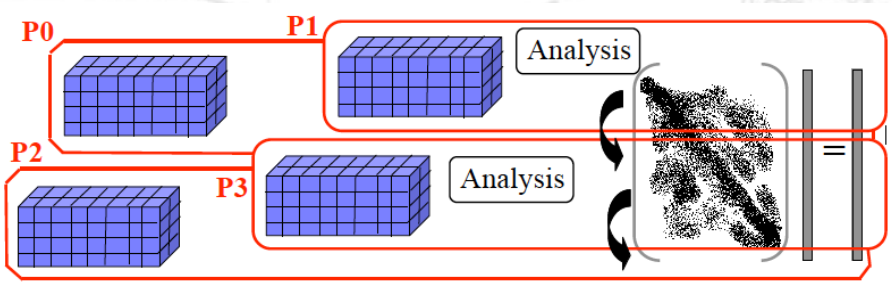
- Additional script needed for parametric analysis;
- It goes into the queue.

# OpenSeesMP: The **M**ultiple **P**arallel OpenSees Interpreter

In case of large models, it works like OpenSeesSP.




The main Processor (P0) interprets the script to build the model and to construct the analysis.



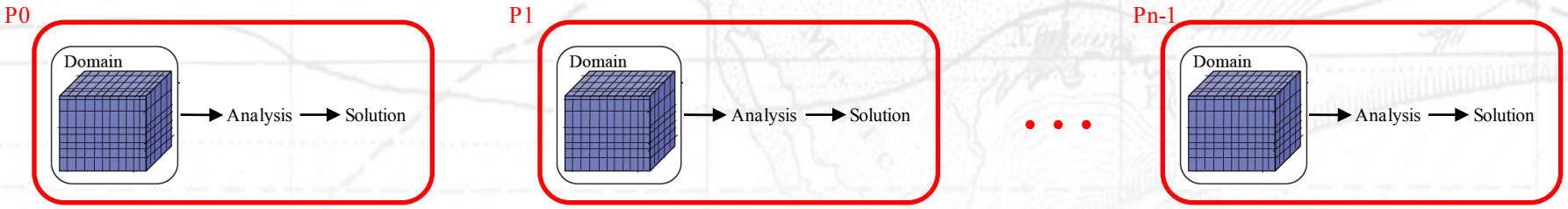

Other processors (P1, P2, P3) are running sub-domains of the model.

Graphics by McKenna

# OpenSeesMP: The **M**ultiple **P**arallel OpenSees Interpreter

In case of parametric analysis, this interpreter runs in parallel several slightly modified version of the basic OpenSees interpreter. Each of them runs an independent analysis and provides:

- the total Number of Processors available (***np***);
- Its unique Processor ID number (***pid***).



# OpenSeesMP: How to modify the script

For large model, changes to the scripts include:

- Change how degrees-of-freedom are numbered (***Numberer Command***) to one of the following:
  - numberer ParallelPlain;
  - numberer ParallelRCM.

- Change the System of Equation and the Solver (***System Command***) to one of the following:
  - system ParallelProfileSPD;
  - system Mumps;
  - system Petsc.

# OpenSeesMP: New Commands

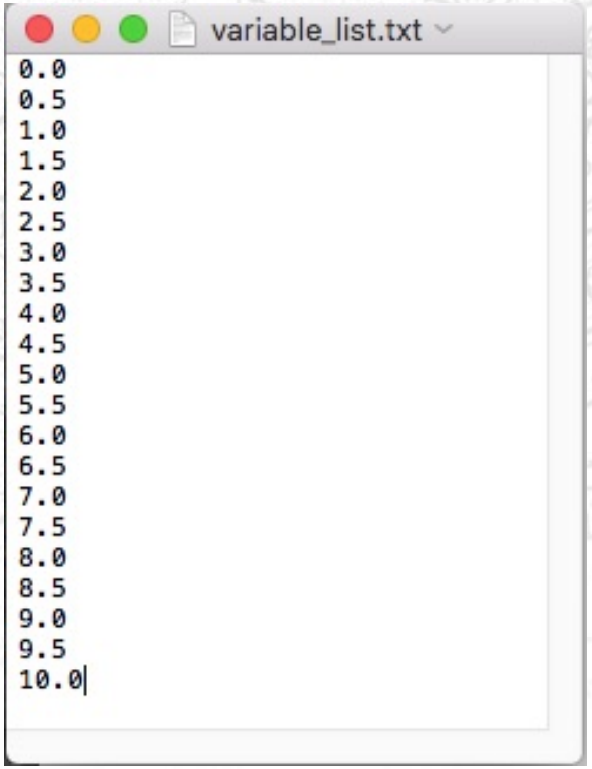
In order to allow each running process to determine the processor it is running on, the number of processors that the user started and to allow the inter-process communications, the following additional commands are provided:

- **getNP**: returns the total number of processors assigned to the user for the job;
- **getPID**: returns a unique processor number ranging between 0 and ($getNP-1);
- **send –pid $pid $data**: to send the data from a local process to a process whose process id is given by the variable pid. Pid must be in the range 0 to [expr[getNP]-1];
- **recv –pid $pid variableName**: to receive data from a remote process and set the variable named variableName to be equal to that data. Pid must be set {0,..[expr [getNP] -1, ANY}. If the value of $pid is ANY, the process can receive data from any process;
- **barrier**: causes all processes to wait until all process reach this point in the code.

Using these commands it is possible for the user to perform their own domain decomposition analysis. The getNP and getPID commands allow the user to specify which nodes and elements are created on which processor.

# OpenSeesMP: Additional tcl script for parametric studies

```
set pid [getPID]                                  # obtains the Processor ID
set np [getNP]                                    # obtains the number of processors
set par_list [open ”variable_list.txt" r]         # defines the list of variables
```



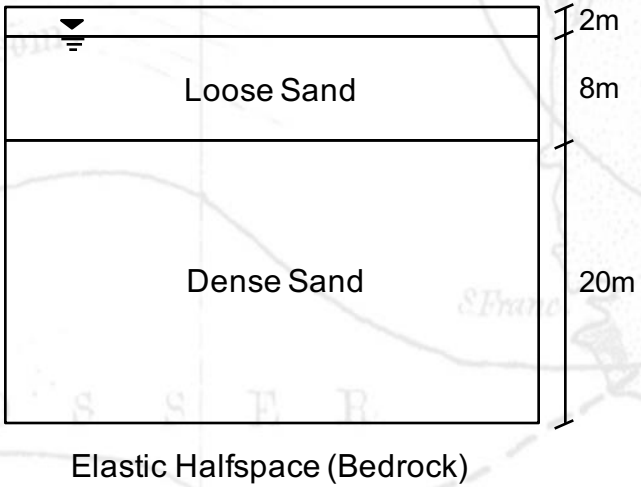
# OpenSeesMP: Additional tcl script for parametric studies

```
set pid [getPID]                                  # obtains the Processor ID
set np [getNP]                                    # obtains the number of processors
set par_list [open ”variable_list.txt" r]         # defines the list of variables
set countP 0                                      # initiates a count to split the analyses

foreach val [split [read $par_list] \n] {         # loops through the variables

if {[expr $countP % $np] == $pid} {               # assigns each analysis to one
                                                    processor
        set valuesList [split $val "/"]           # reads one value of the variable
        set grade [lindex $valuesList end]        # assigns the value to the variable
        source Sequential_script.tcl              # calls the sequential script
        wipe                                      # cleans the workspace
        }

        incr countP 1                             # increases the count to move to the
                                                    next analysis
        }
```

DESIGNSAFE-CI A NATURAL HAZARDS ENGINEERING COMMUNITY · TEXAS The University of Texas at Austin · TACC · RICE · Florida Tech

# Effective Site Response Analysis

(http://opensees.berkeley.edu/wiki/index.php/Effective_Stress_Site_Response_Analysis_of_a_Layered_Soil_Column)



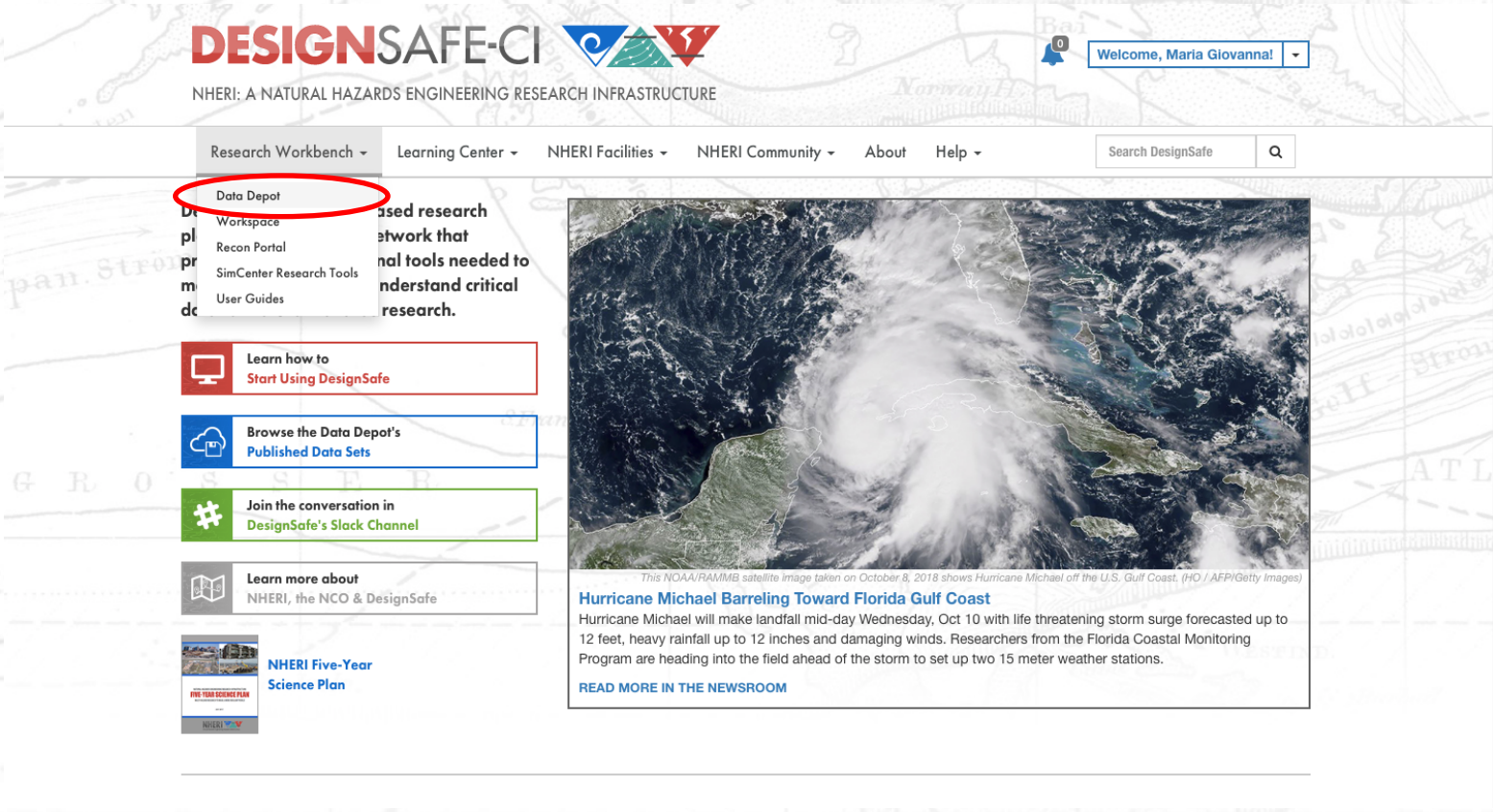

Elastic Halfspace (Bedrock)

Problem: effective stress site response analysis of a layered deposit of cohesionless soil underlain by an elastic half-space.

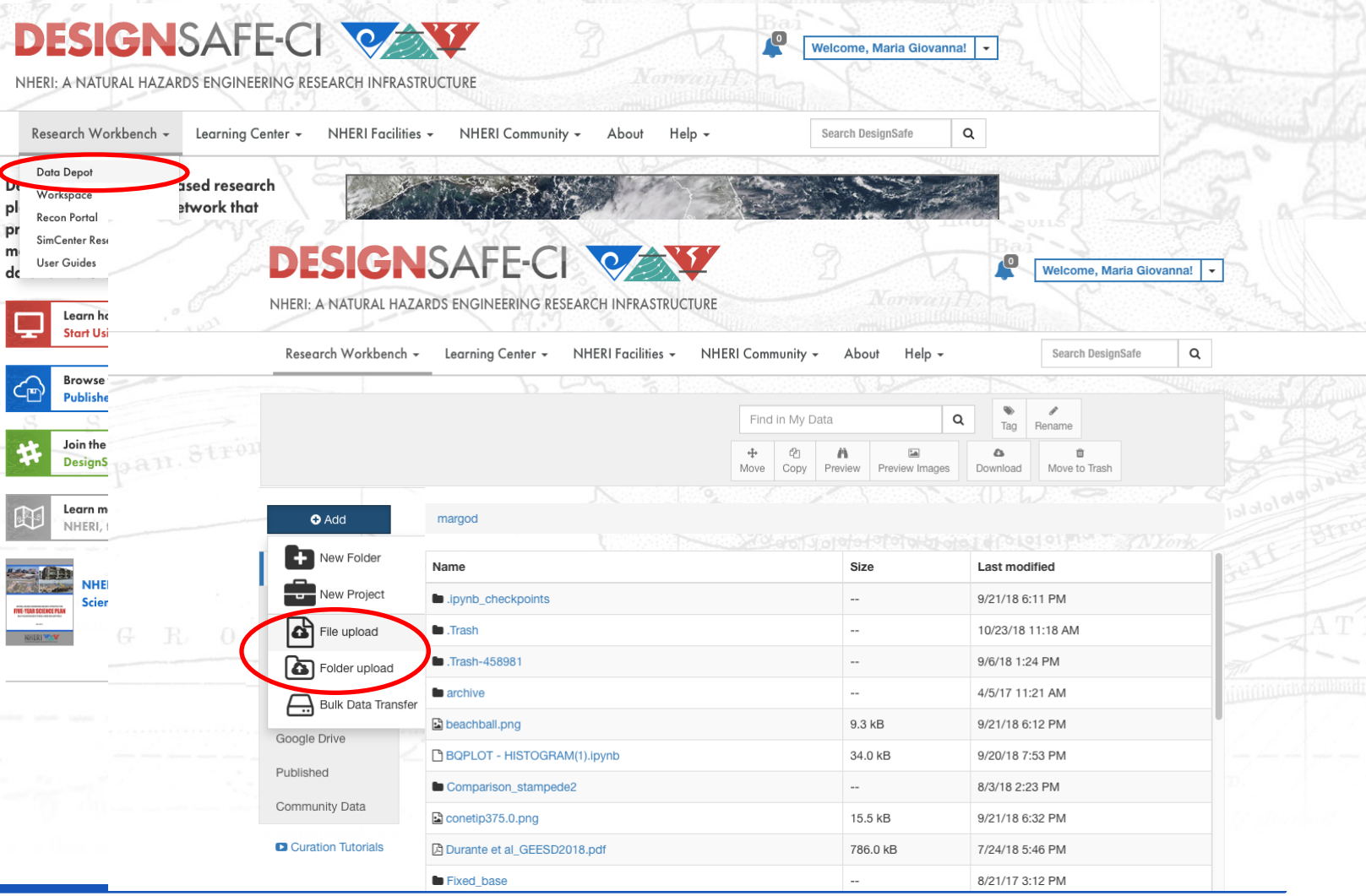
Model: A single column of soil is modeled in 2D (with periodic boundary conditions to emulate a 1D analysis) and is subject to an earthquake excitation. Nine node quadrilateral elements with both displacement and pore pressure degrees of freedom enable the model to track changes in pore pressure and effective stress during the earthquake excitation. A Lysmer-Kuhlemeyer (1969) dashpot is utilized to account for the finite rigidity of the underlying elastic medium.

Parametric analysis with OpenSeesMP using three different input motions

# Upload files in My Data

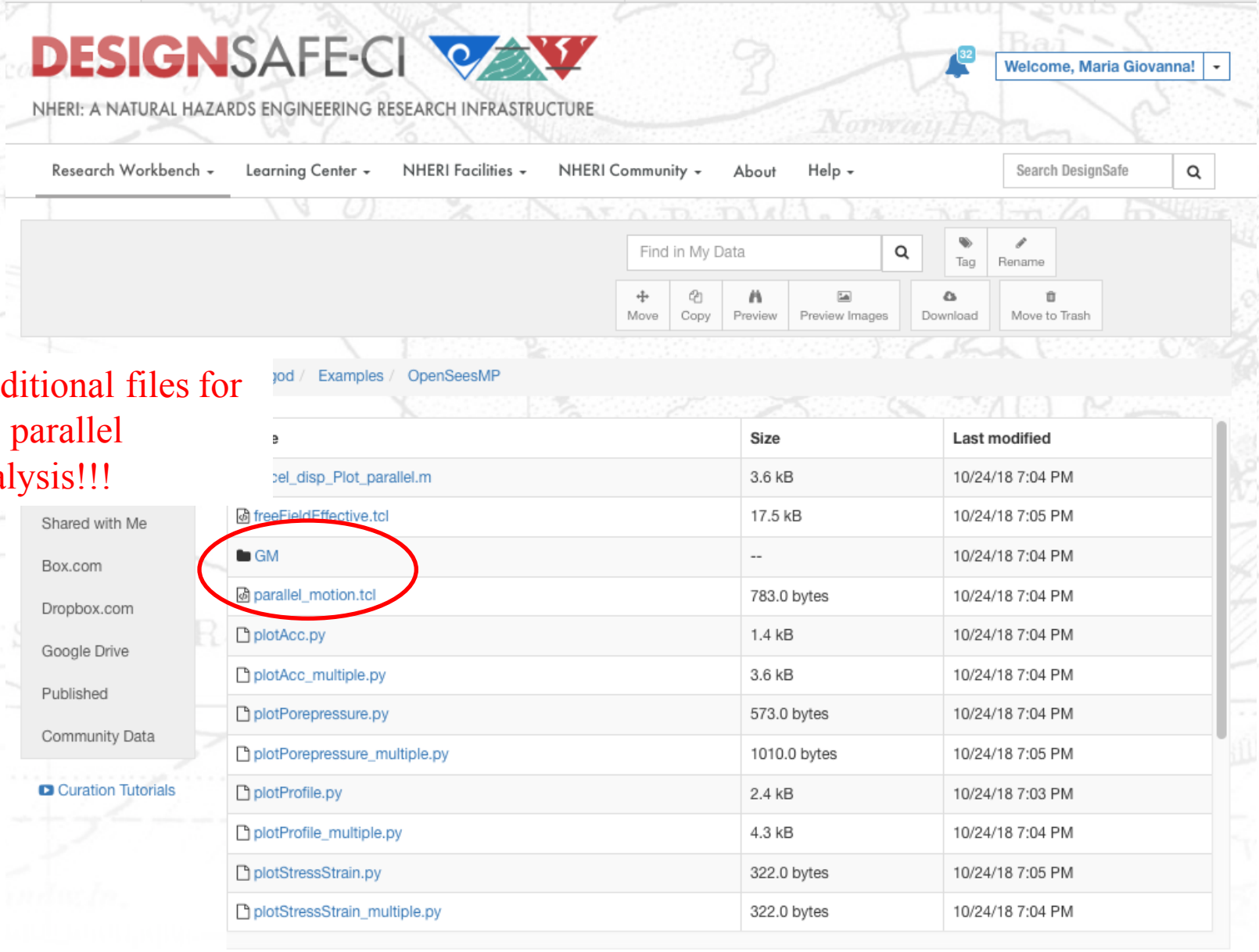


# Upload files in My Data



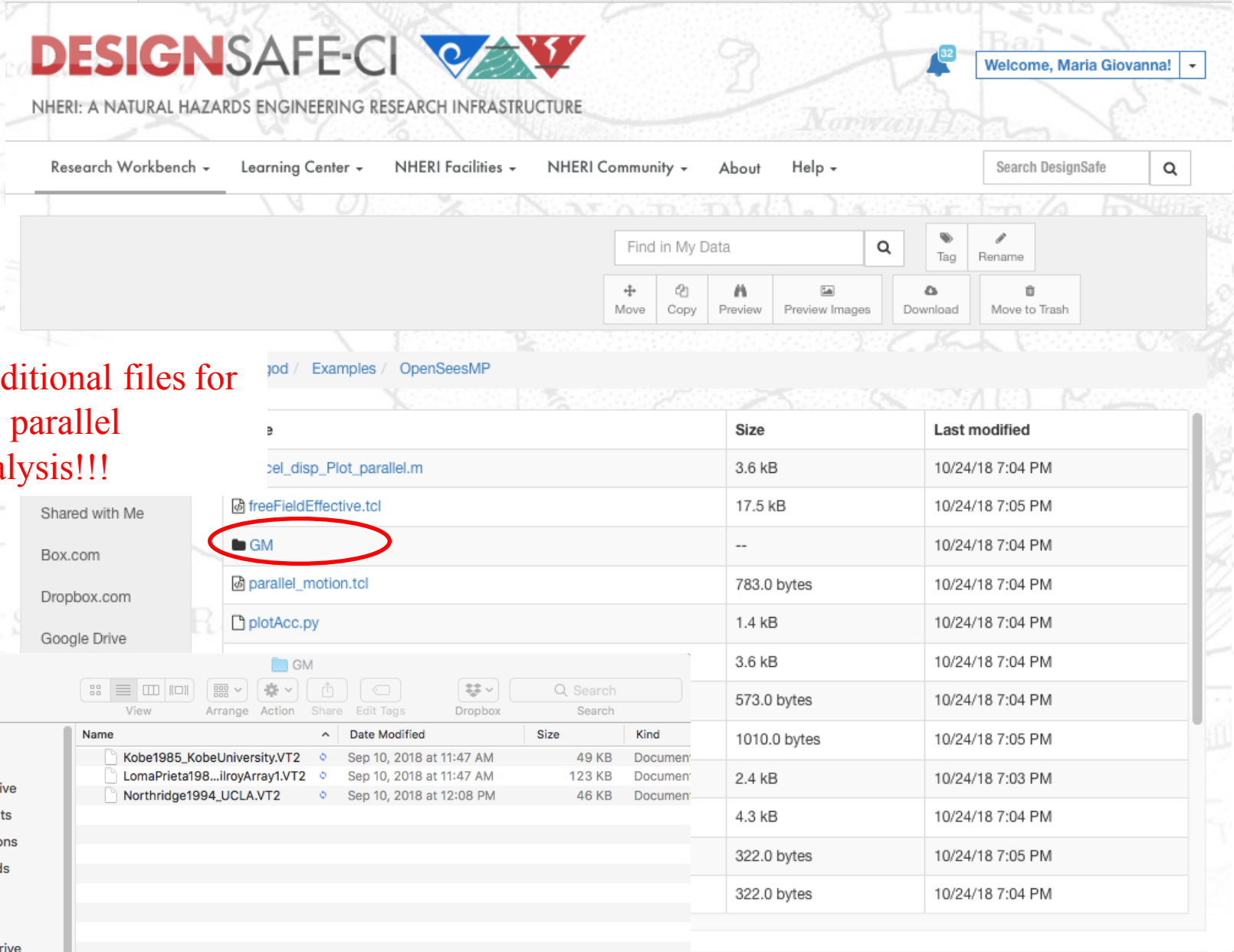

DESIGNSAFE-CI A NATURAL HAZARDS ENGINEERING COMMUNITY · TEXAS The University of Texas at Austin · TACC · RICE · Florida Tech

# OpenSeesMP analysis: *additional files needed*




Additional files for the parallel analysis!!!

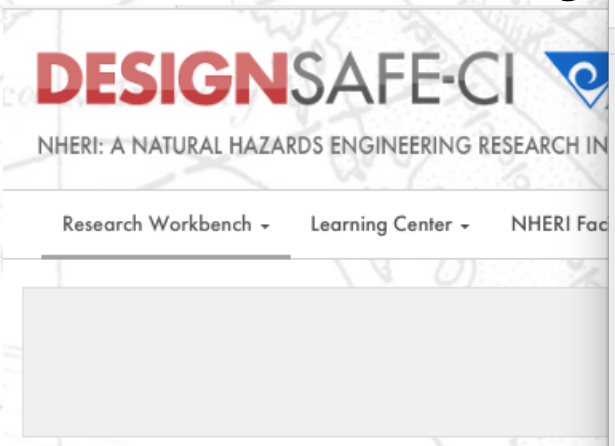
# OpenSeesMP analysis: *additional files needed*




Additional files for the parallel analysis!!!

# OpenSeesMP analysis: *additional files needed*

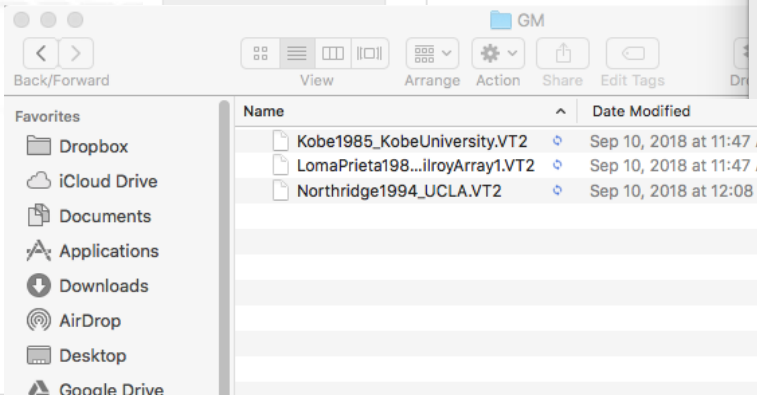
Additional files for the parallel analysis!!!



parallel_motion.tcl

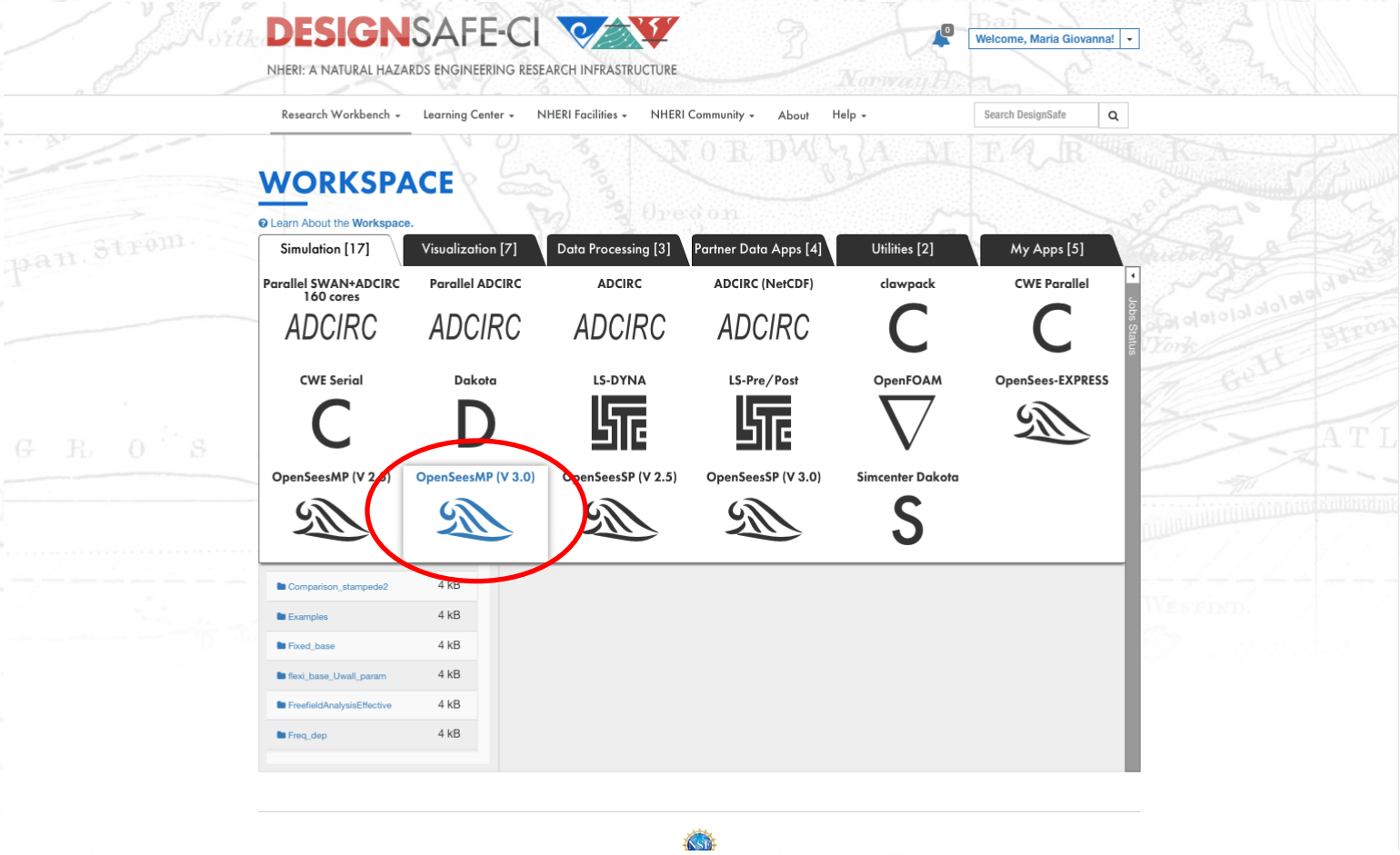
~/Desktop/Research/UserGuide/Webinar/OpenSeesMP/parallel_motion.tcl

```
set pid [getPID]
set np [getNP]
#set par_list [open "slope_values.txt" r]
set countP 0;

source ReadRecord.tcl

set startT  [clock seconds]

foreach gMotion [glob -nocomplain -directory GM *.VT2] {
if {[expr $countP % $np] == $pid} {

set gMotionName [string range $gMotion 0 end-4]

set MotionID [string range $gMotion 3 end-4]

set name1 Motion$MotionID

ReadRecord ./$gMotionName.VT2 ./$gMotionName.dat dT nPts

set velocityFile ./$gMotionName.dat

puts "set velocityFile ./$gMotionName.dat"

# time step in ground motion record
set motionDT     $dT
# number of steps in ground motion record
set motionSteps  $nPts

source freeFieldEffective.tcl
 wipe
}
incr countP 1;
}
set endT    [clock seconds]

puts "Total duration for processor $pid [expr $endT-$startT] seconds!"
```

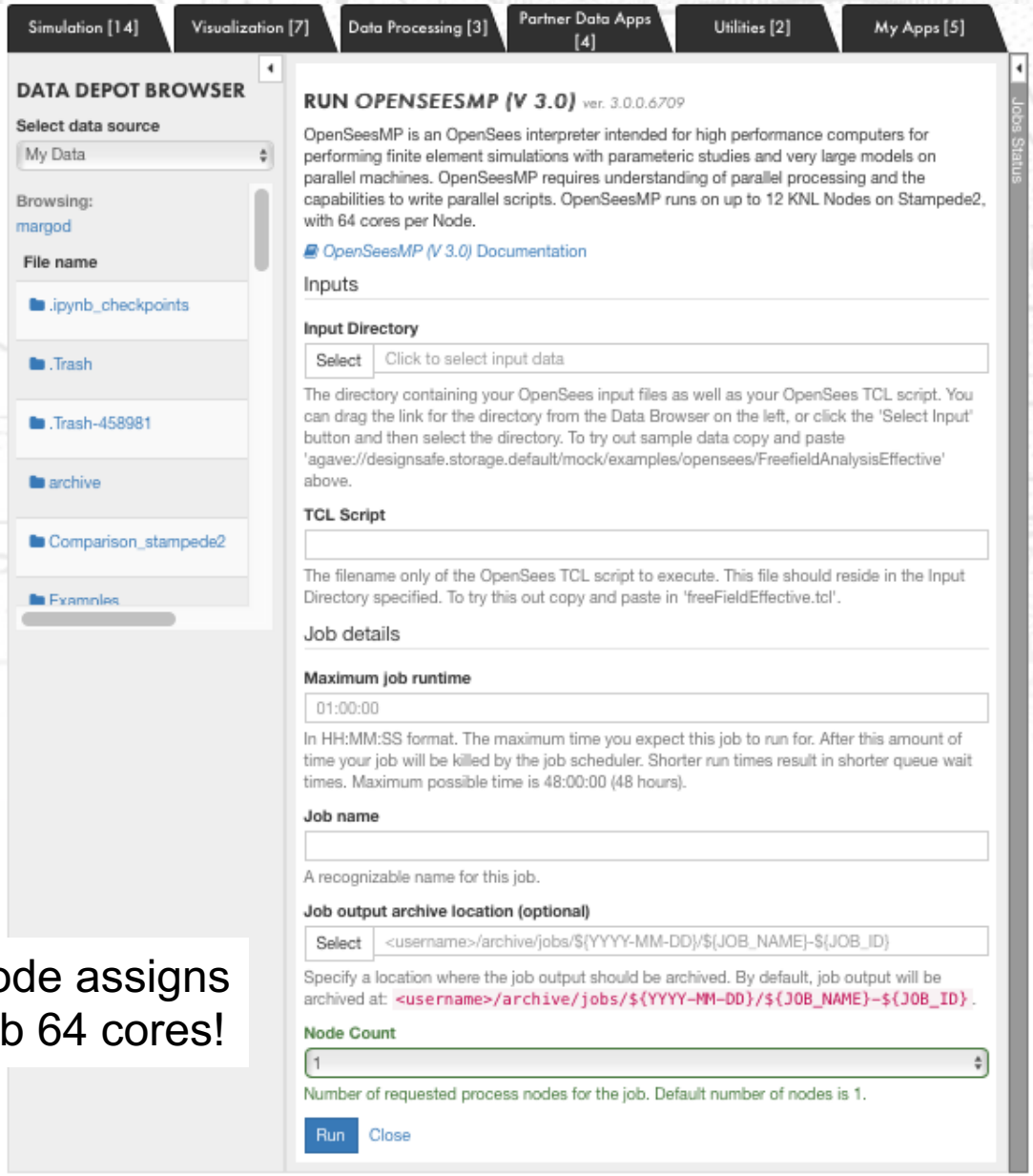



| | |
|---|---|
| 1010.0 bytes | 10/24/18 7:05 PM |
| 2.4 kB | 10/24/18 7:03 PM |
| 4.3 kB | 10/24/18 7:04 PM |
| 322.0 bytes | 10/24/18 7:05 PM |
| 322.0 bytes | 10/24/18 7:04 PM |

# OpenSeesMP analysis: *Run the analysis*



# OpenSeesMP analysis: *Run the analysis*

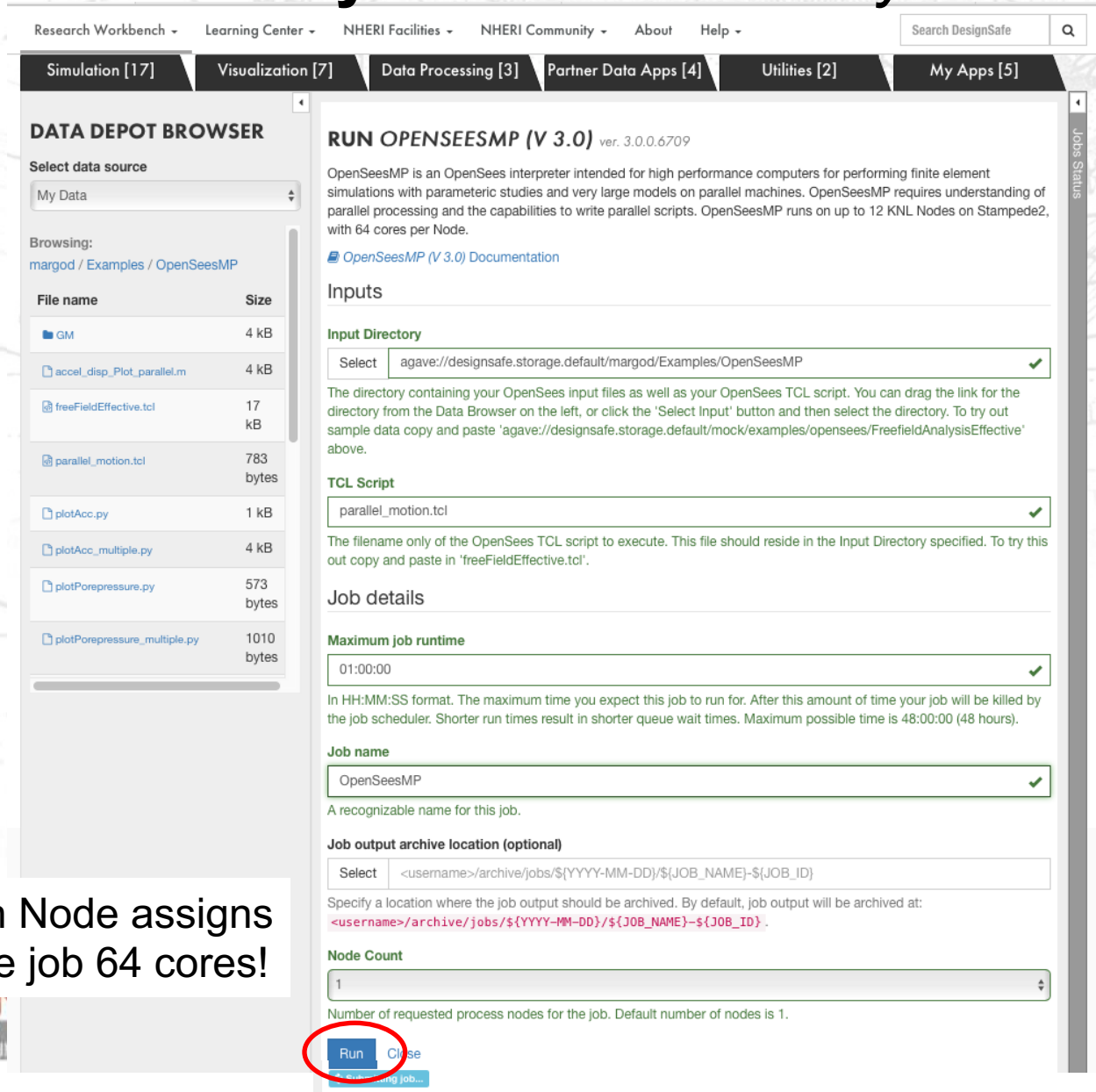



Each Node assigns to the job 64 cores!

DESIGN SAFE
A NATURAL HAZARDS ENGINEERING COMMUNITY

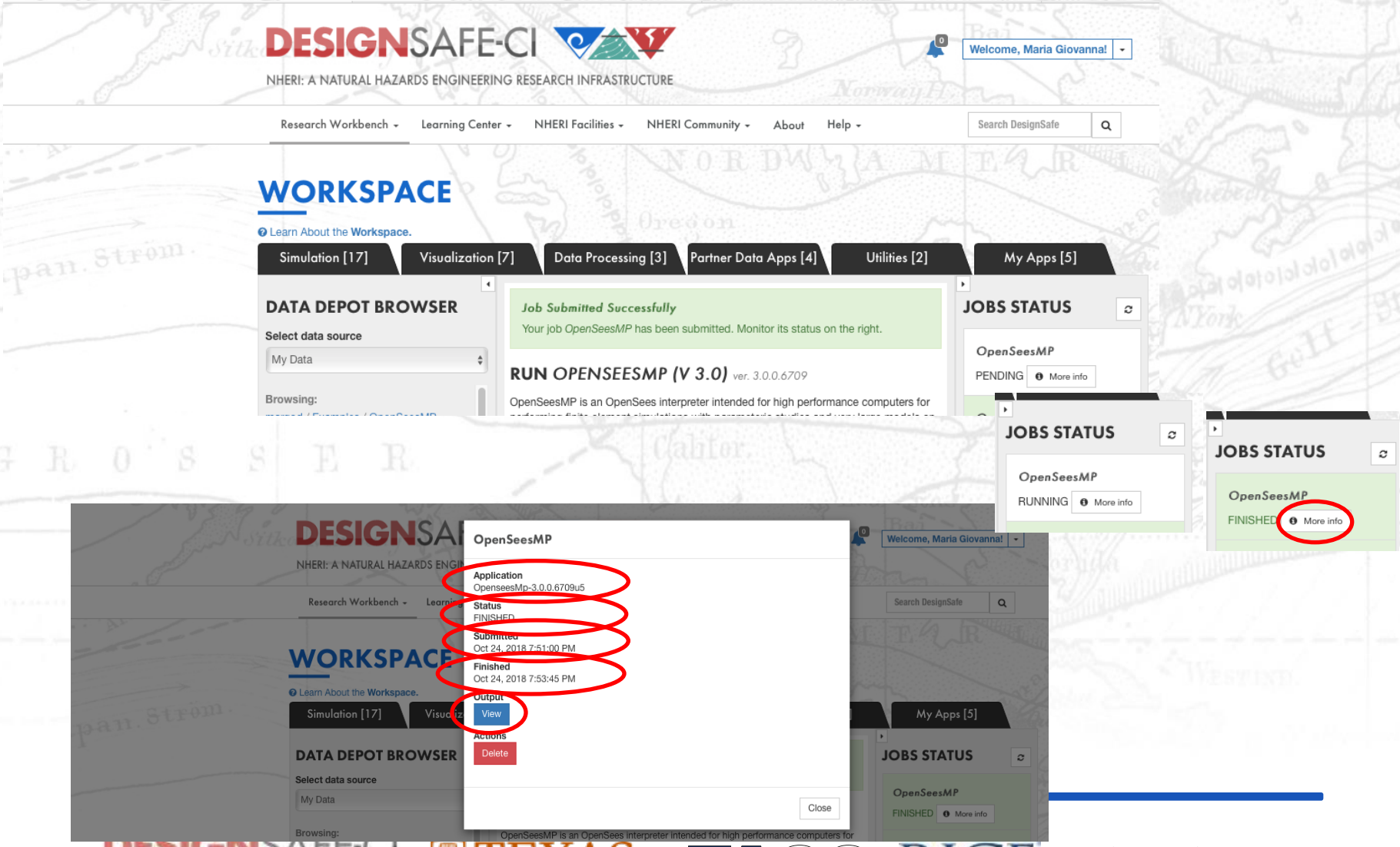
'ida Tech

# OpenSeesMP analysis: *Run the analysis*




Each Node assigns to the job 64 cores!

# OpenSeesMP analysis: *Check job status*




DESIGNSAFE-CI A NATURAL HAZARDS ENGINEERING COMMUNITY

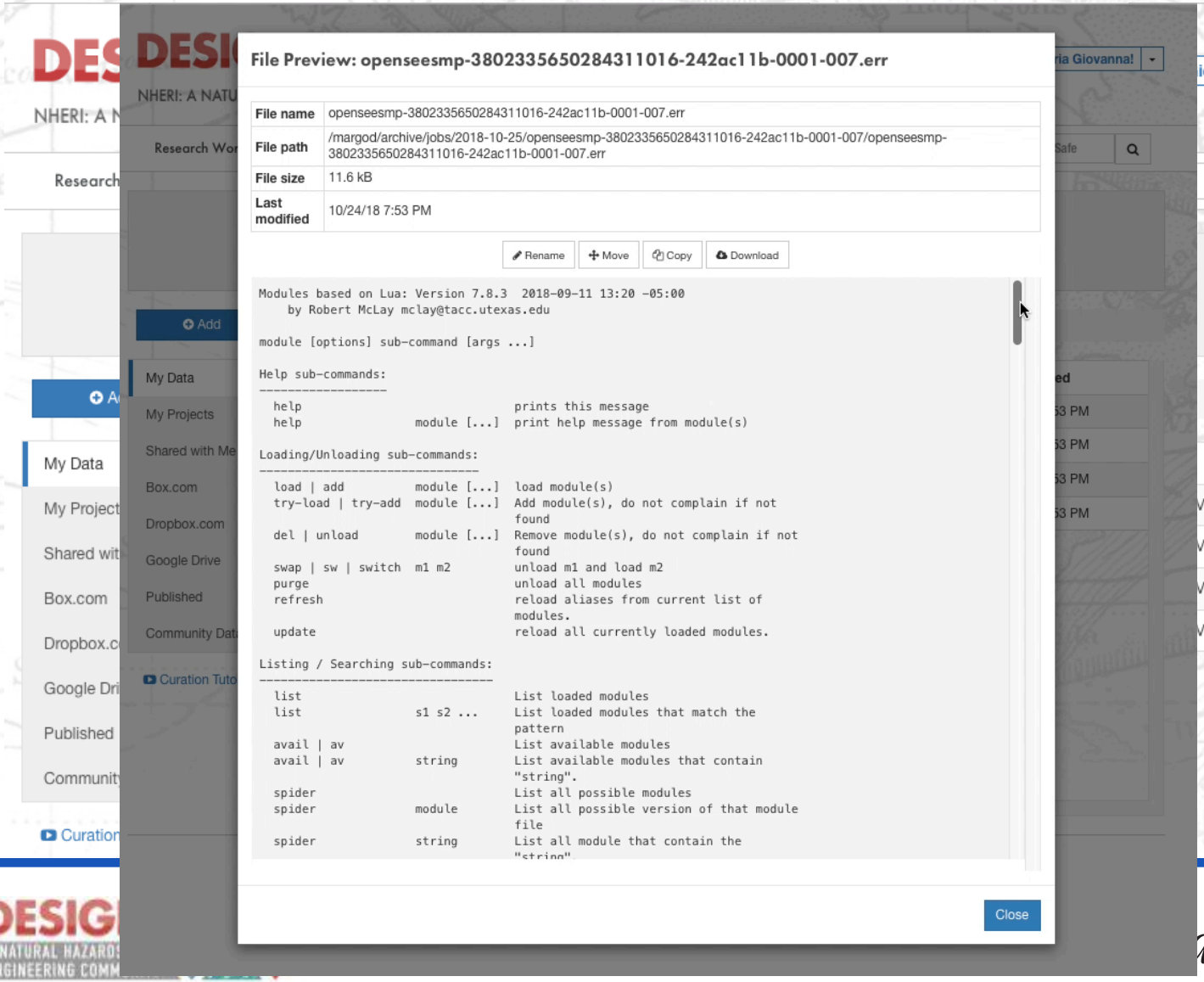
TEXAS The University of Texas at Austin

TACC

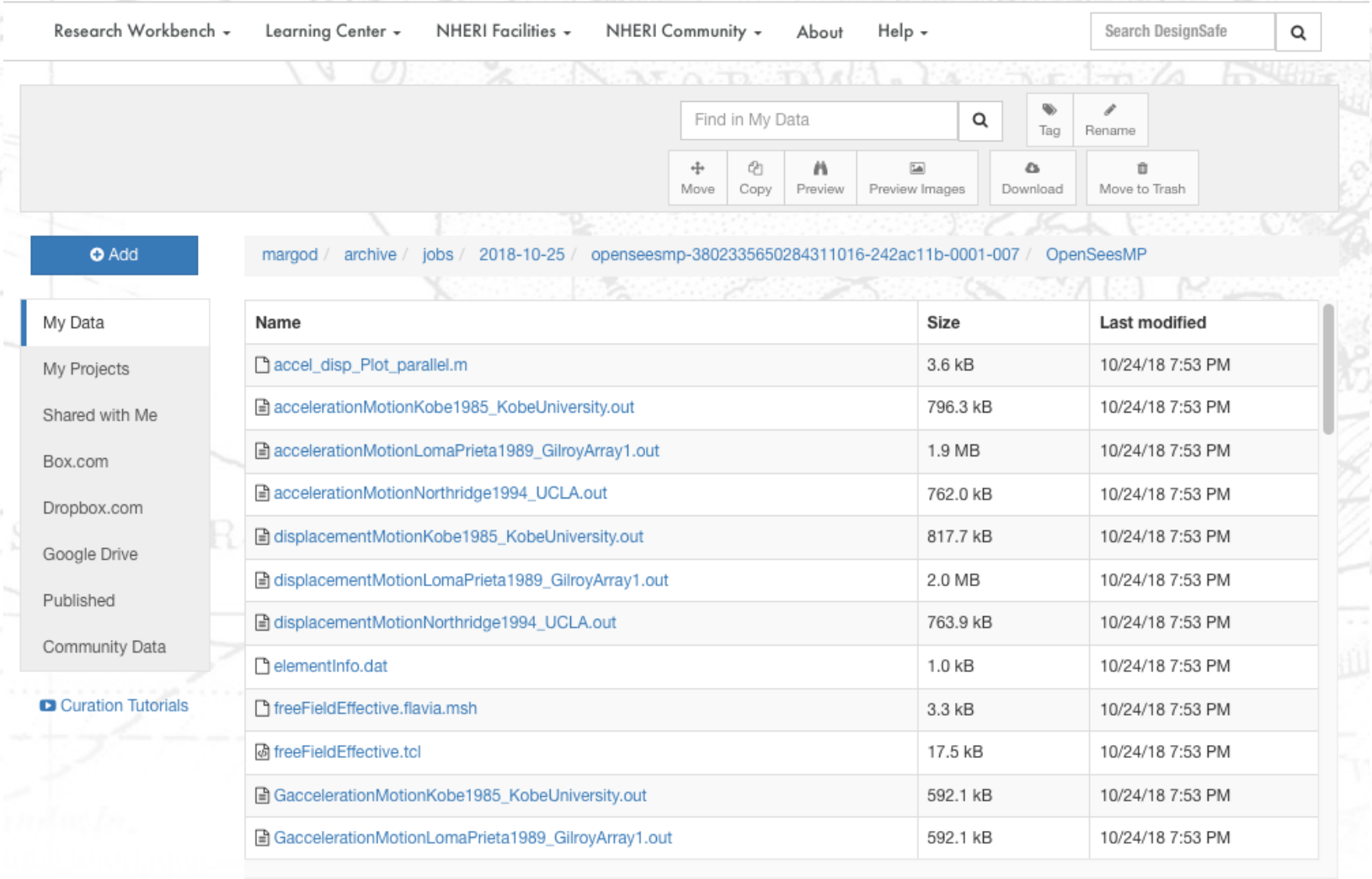
RICE

Florida Tech

# OpenSeesMP analysis: *Check job result*



# OpenSeesMP analysis: *Check job result*



# Post-processing alternatives

- Work on the Cloud using the tools available on DesignSafe (Data Processing Tab):
    - Jupyter Notebook;
    - Matlab.

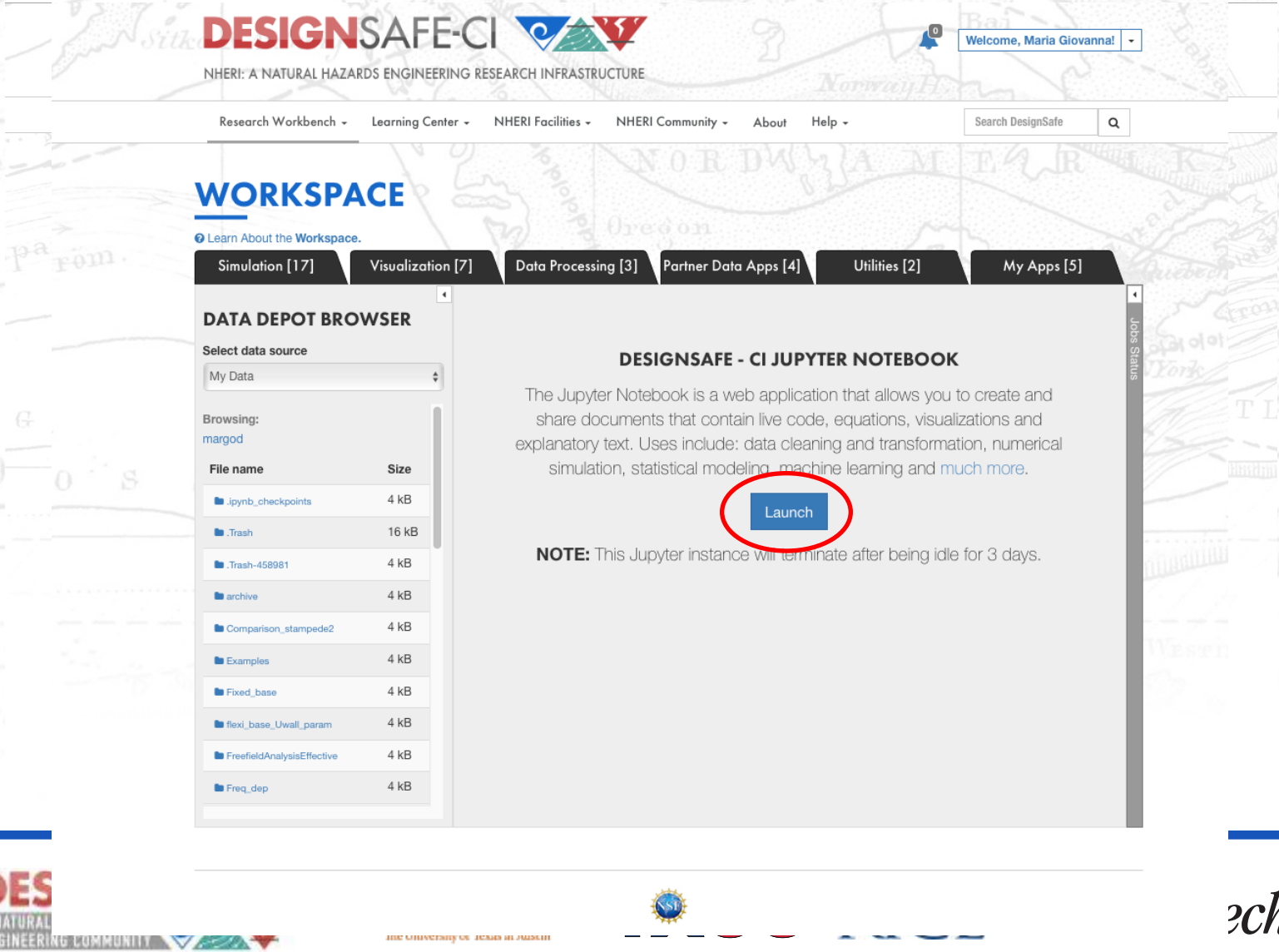
- Download all the output and post-process data locally.

# Post-processing alternatives

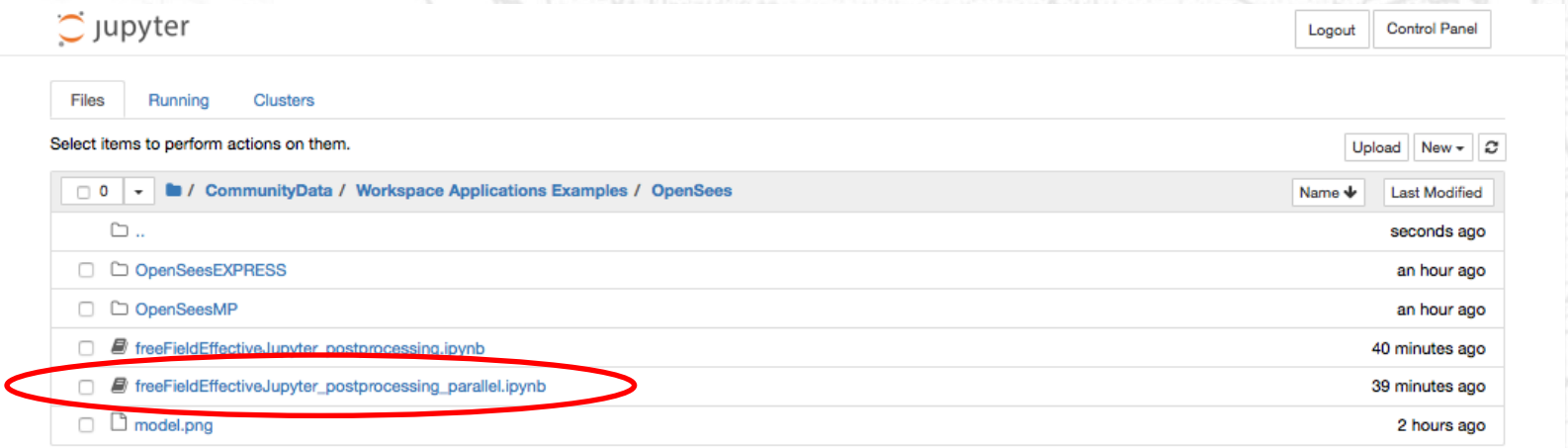
- Work on the Cloud using the tools available on DesignSafe (Data Processing Tab):
    - Jupyter Notebook;
    - Matlab.

- Download all the output and post-process data locally.

# Post-processing alternatives: *Jupyter Notebook*



# **Post-processing alternatives:** *Jupyter Notebook – OpenSeesMP* *(Community Data/Workspace Applications Examples / OpenSees/ freeFieldEffectiveJupyter_postprocessing_parallel.ipynb)*



# **Post-processing alternatives:** *Jupyter Notebook – OpenSeesMP* *(Community Data/Workspace Applications Examples / OpenSees/ freeFieldEffectiveJupyter_postprocessing_parallel.ipynb)*

### Free Field Analysis Example

This example shows how to postprocess the output results of OpenSeesMP analysis using python scripts.

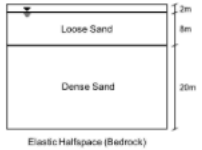
The site response analysis discussed in this article is for a soil profile consisting of a 10 m thick layer of loose sand (Dr = 40%) above a 20 m thick layer of more dense sand (Dr = 75%). A schematic representation of the analyzed soil profile is shown in Fig. 1. The entire soil profile is underlain by an elastic half-space which represents the finite rigidity of an underlying medium such as bedrock. The groundwater table is located at a depth of 2 m, therefore, saturated unit weights are used for the soil below this point and effective stress analysis is considered through the use of nine-node quadrilateral elements which are able to simulate fluid-solid coupling.

The results are presented for each ground motion in terms of:

a) Time history of acceleration at the surface and corresponding response spectra;

b) Profiles of maximum displacement, PGA, maximum shear strain, stress ratio, and stress strain plots for a point near the center of the liquefiable zone;

c) Evolution of pore water pressure for a point near the center of the liquefiable zone.




### Postprocess Results

Out[1]: The raw code for this IPython notebook is by default hidden for easier reading. To toggle on/off the raw code, click here.

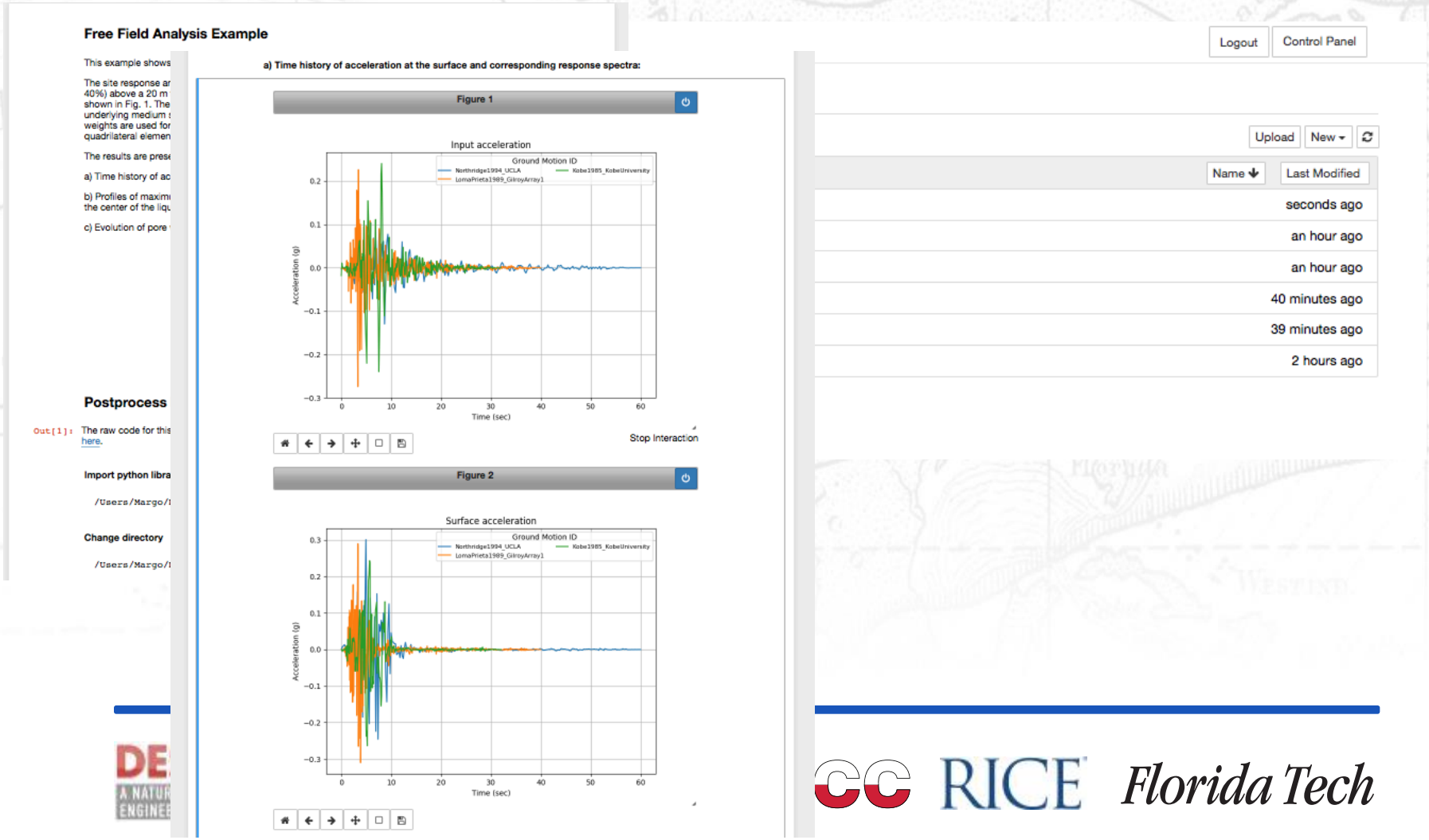
**Import python libraries**

```
/Users/Margo/Desktop/Research/UserGuide/Webinar
```

**Change directory**

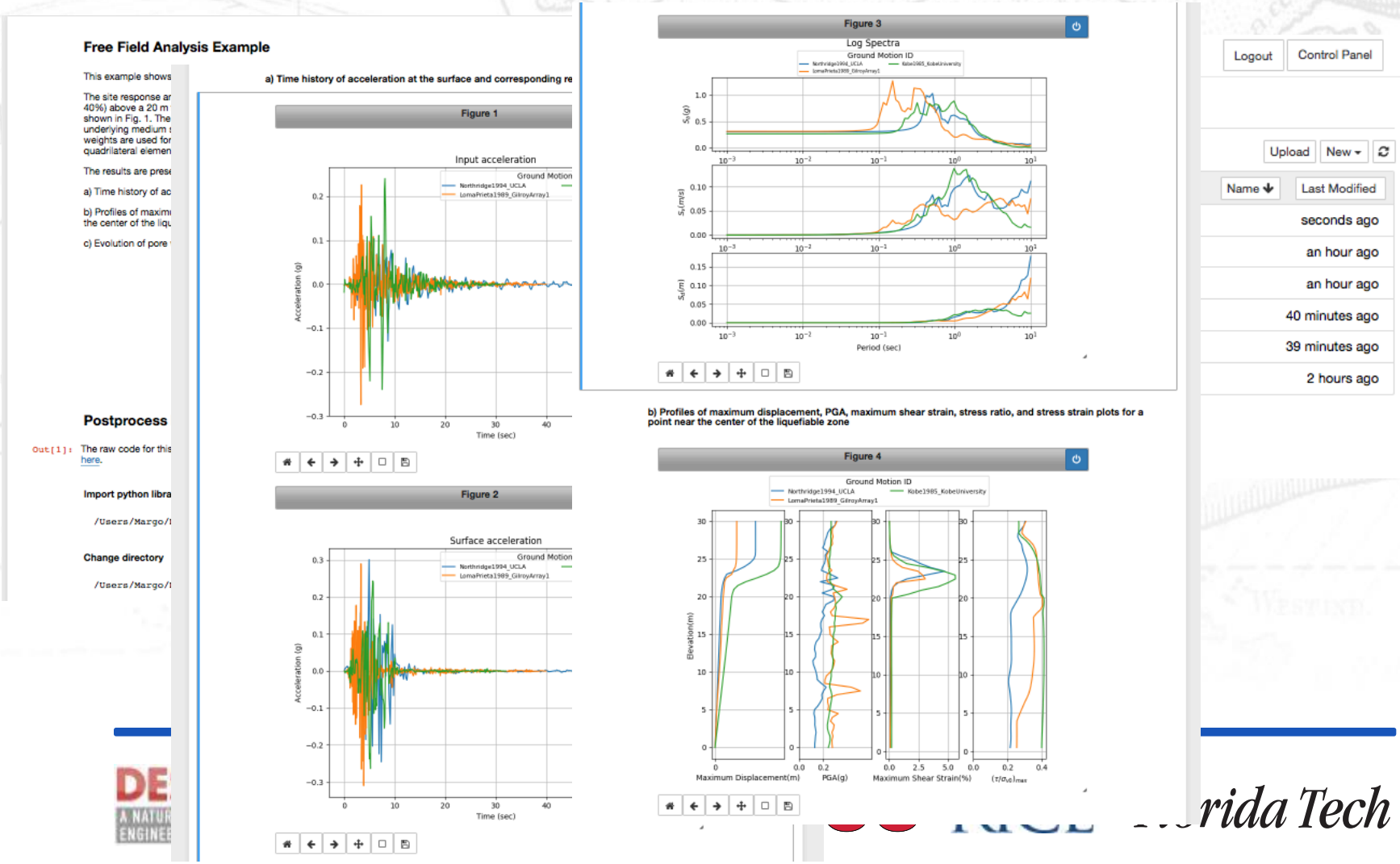
```
/Users/Margo/Desktop/Research/UserGuide/Webinar/OpenSeesMP_motion
```

# **Post-processing alternatives:** *Jupyter Notebook – OpenSeesMP* *(Community Data/Workspace Applications Examples / OpenSees/ freeFieldEffectiveJupyter_postprocessing_parallel.ipynb)*

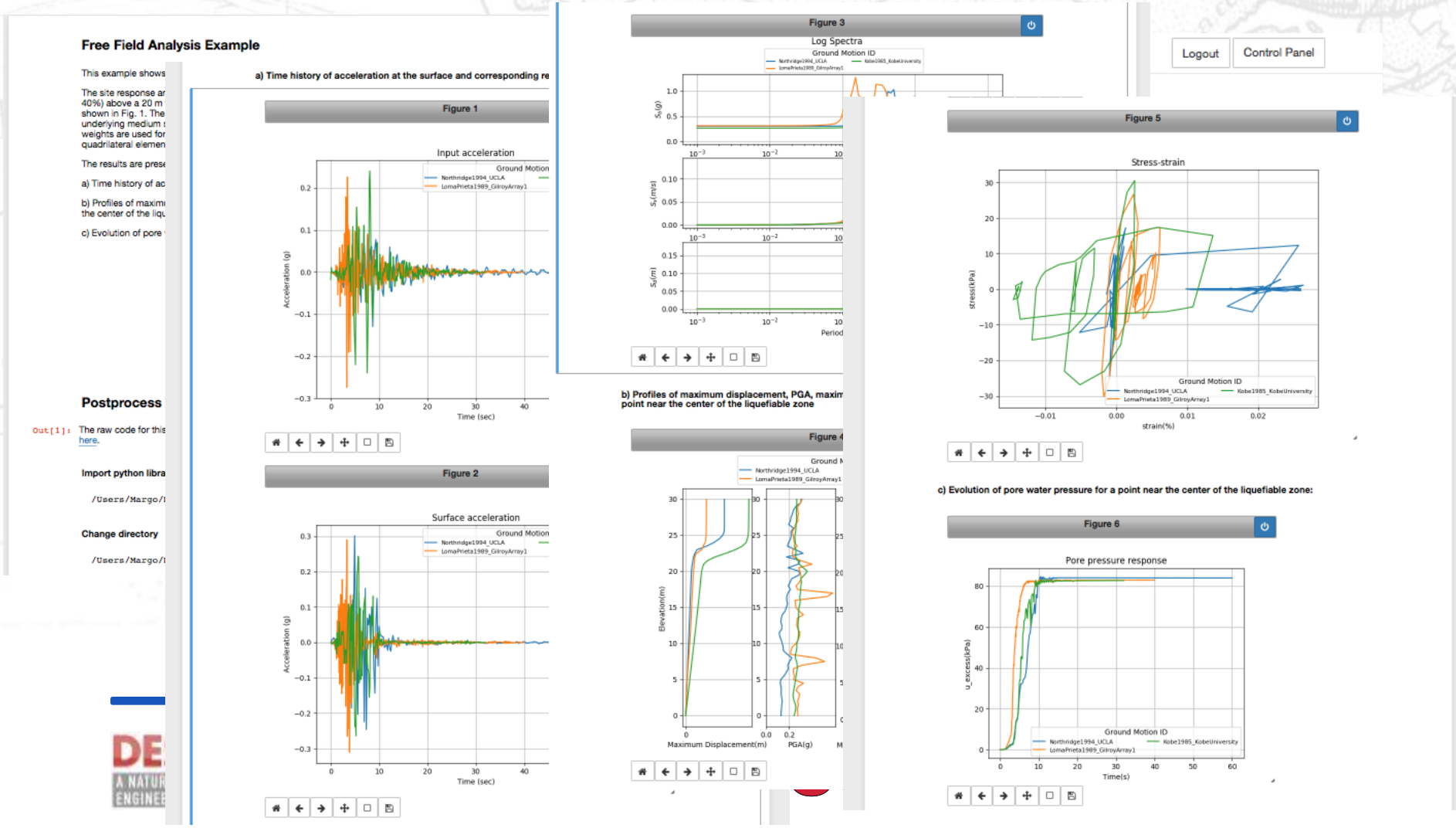



RICE Florida Tech

# **Post-processing alternatives:** *Jupyter Notebook – OpenSeesMP* *(Community Data/Workspace Applications Examples / OpenSees/ freeFieldEffectiveJupyter_postprocessing_parallel.ipynb)*



# **Post-processing alternatives:** *Jupyter Notebook – OpenSeesMP* *(Community Data/Workspace Applications Examples / OpenSees/ freeFieldEffectiveJupyter_postprocessing_parallel.ipynb)*



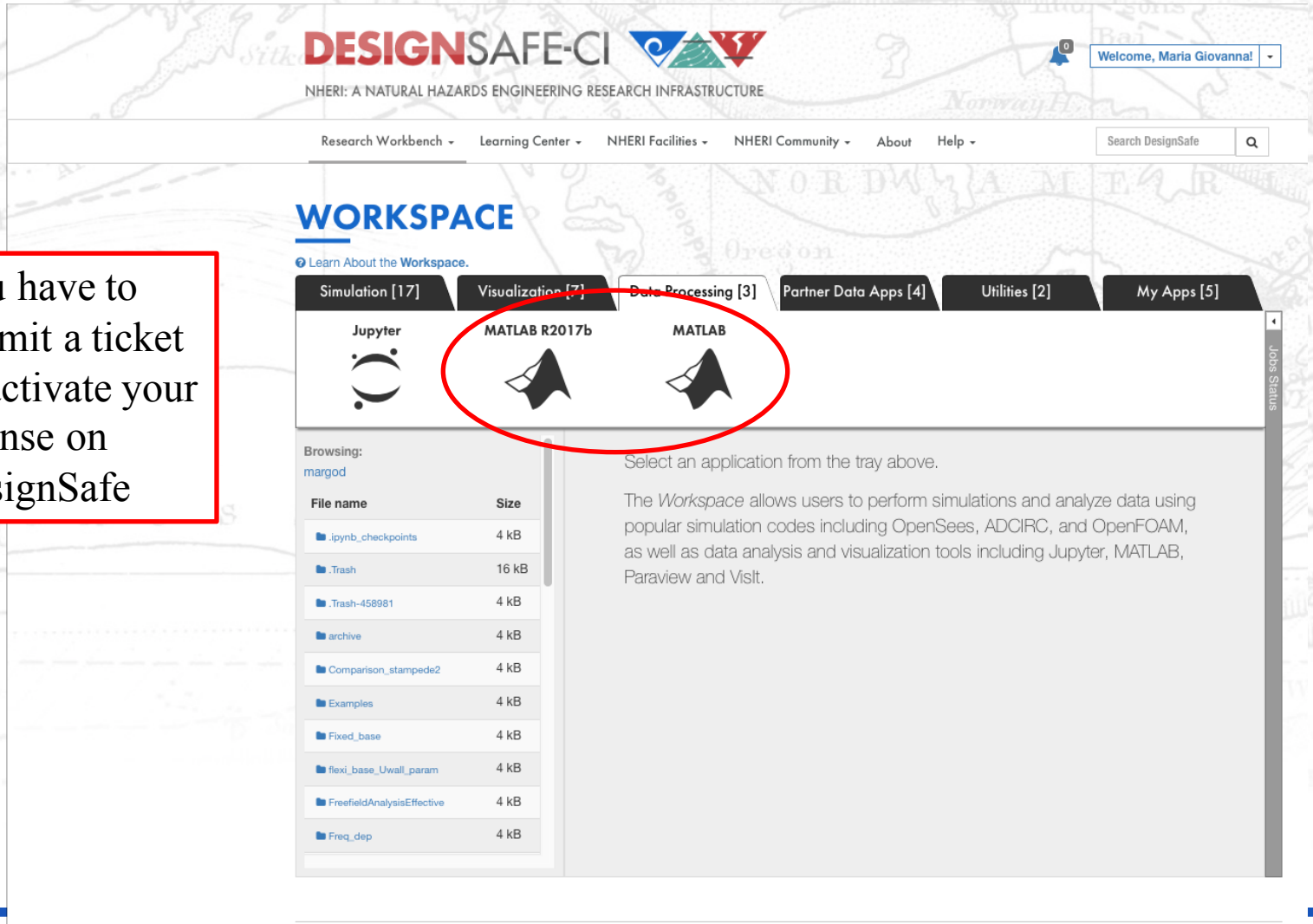
# Post-processing alternatives

- Work on the Cloud using the tools available on DesignSafe (Data Processing Tab):
    - Jupyter Notebook;
    - Matlab.

- Download all the output and post-process data locally.

# Post-processing alternatives: *Matlab*

You have to submit a ticket to activate your license on DesignSafe



# Post-processing alternatives: *Matlab*

MATLAB

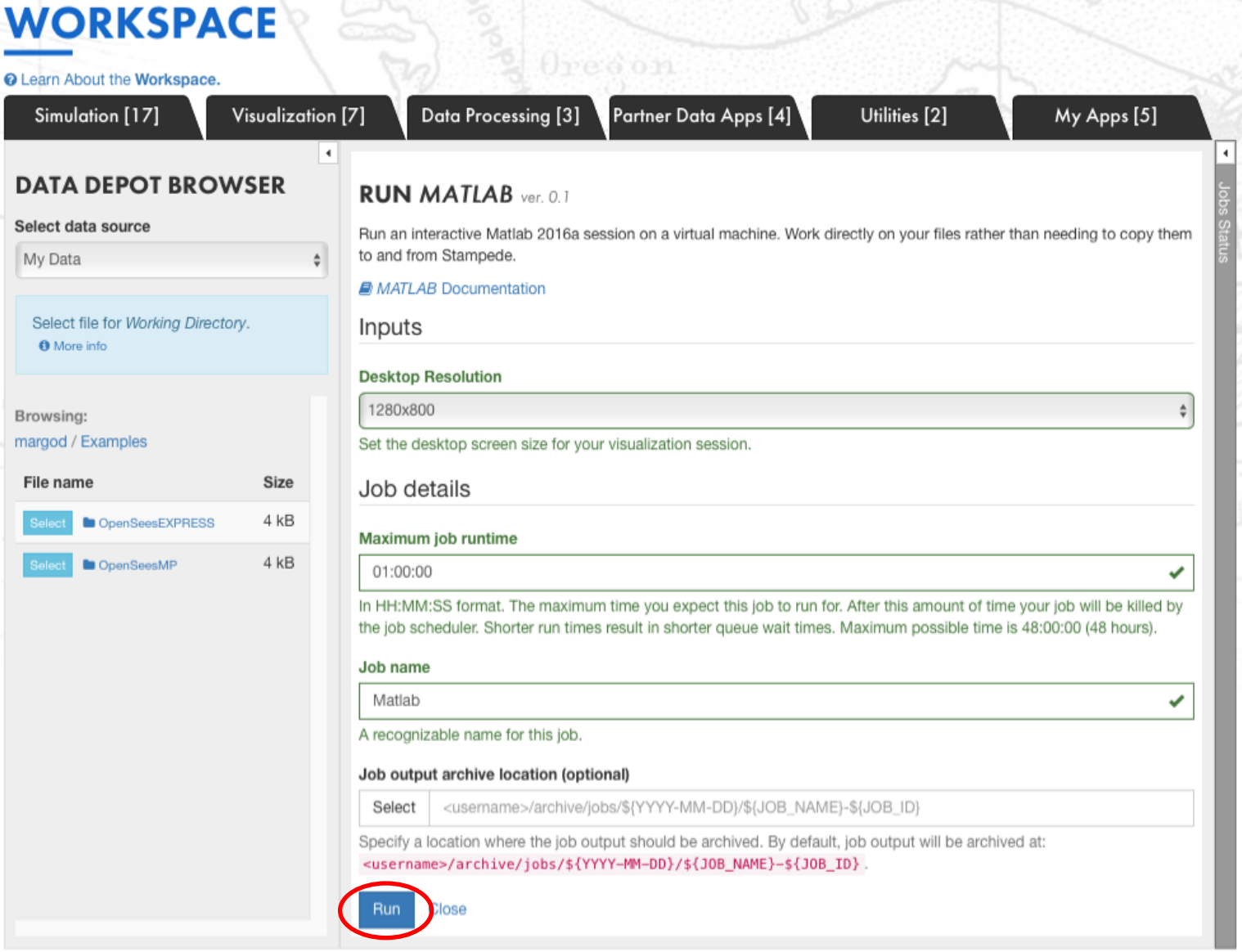
Run an interactive Matlab 2016a session on a virtual machine. Work directly on your files rather than needing to copy them to and from Stampede.

Recommended for standard post-processing analyses.

MATLAB R2017b

Run an interactive MATLAB 2017b session on Stampede2.

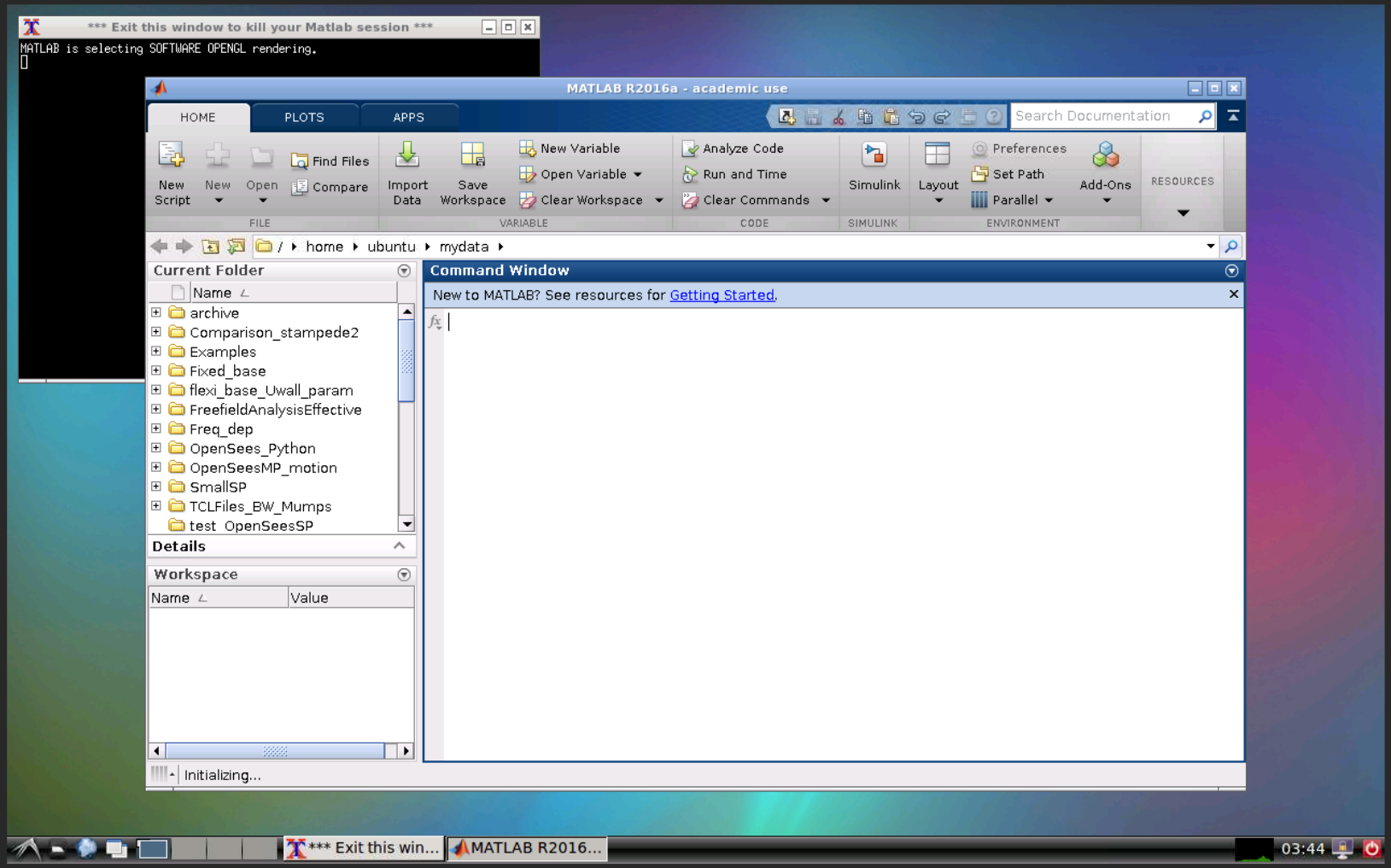
# Post-processing alternatives: *Matlab*



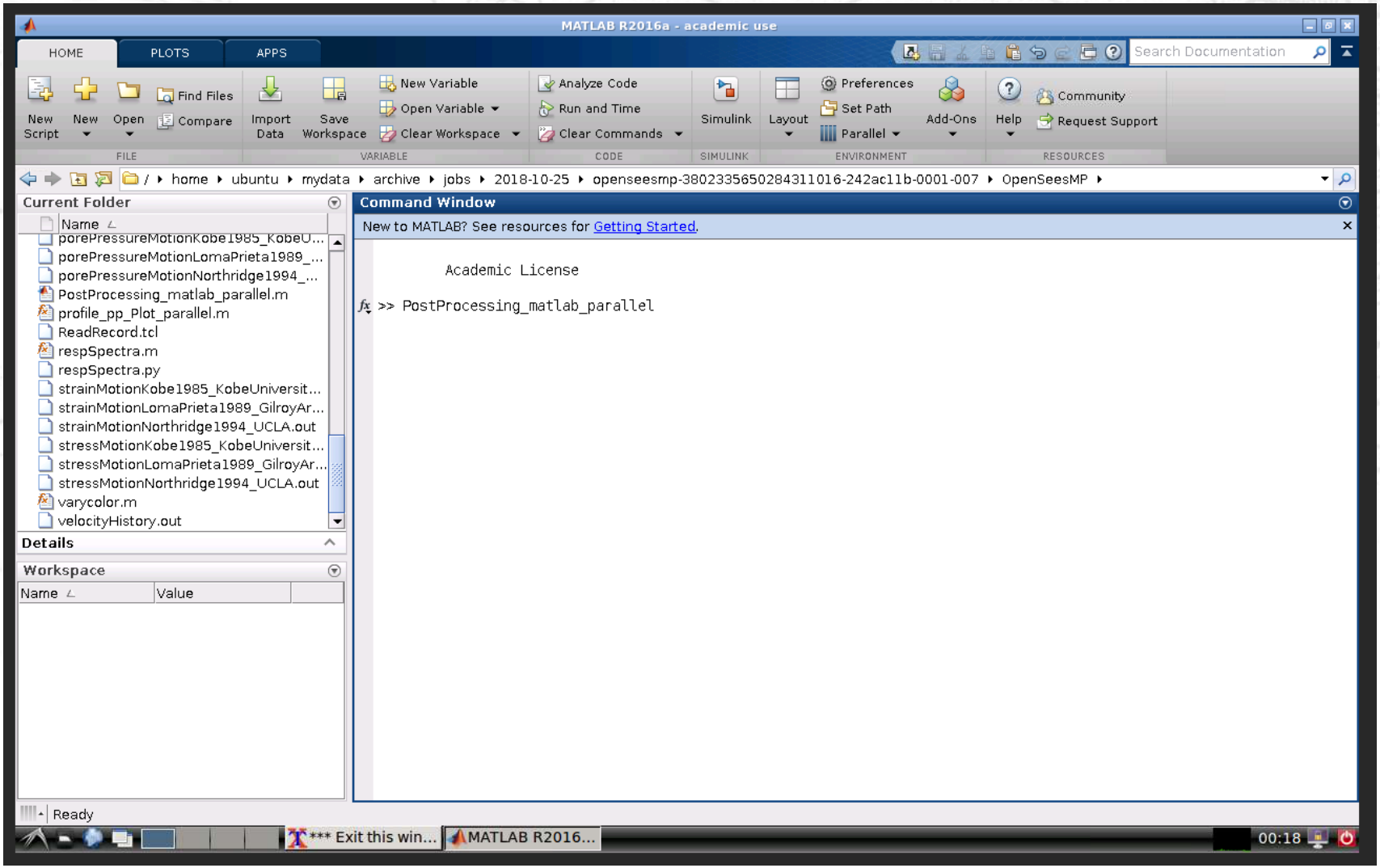

d for

alyses.

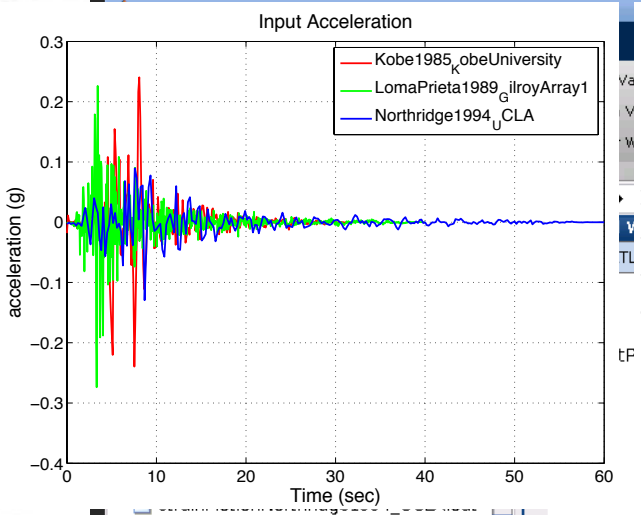
# Post-processing alternatives: *Matlab*

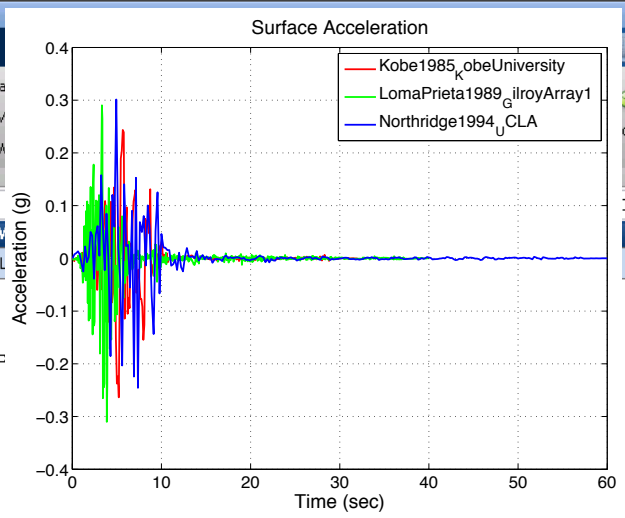


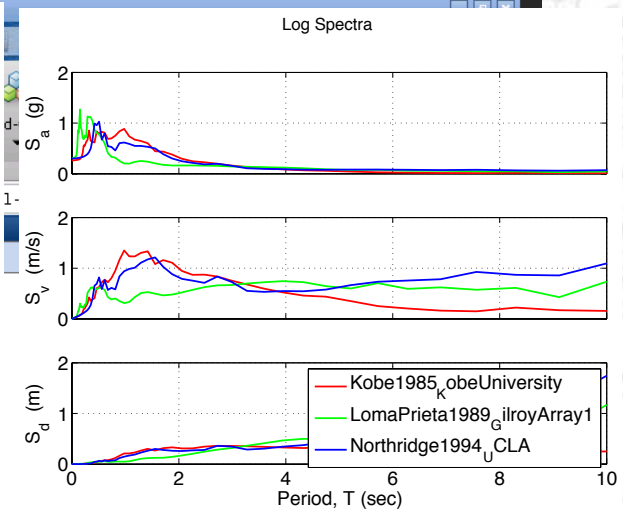
# Post-processing alternatives: *Matlab – OpenSeesMP* *(Community Data/Workspace Applications Examples / OpenSees/ OpenSeesMP /PostProcessing_matlab_parallel.m )*

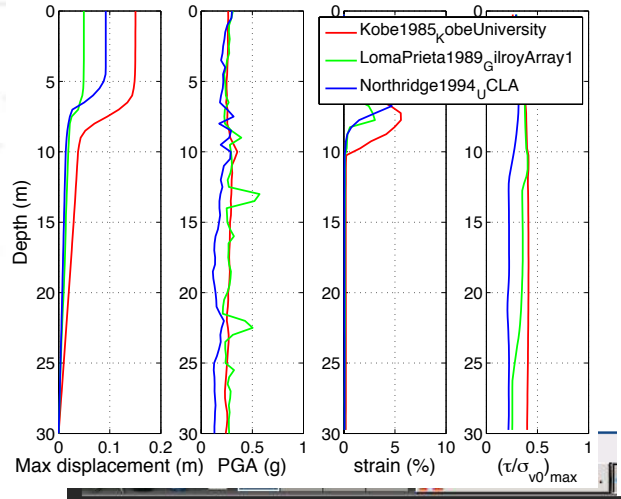


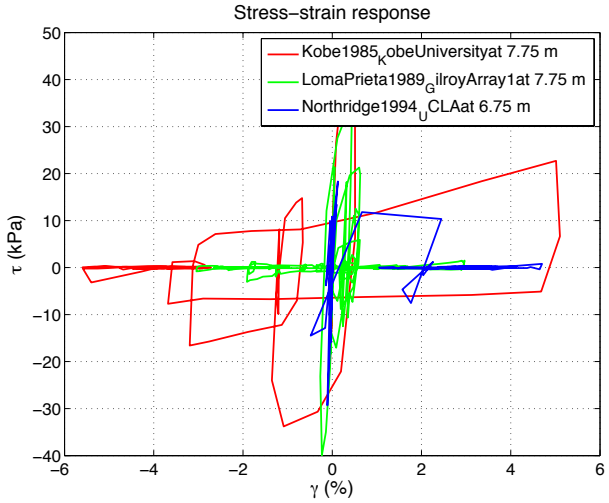
# Post-processing alternatives: *Matlab – OpenSeesMP* *(Community Data/Workspace Applications Examples / OpenSees/ OpenSeesMP /PostProcessing_matlab_parallel.m )*

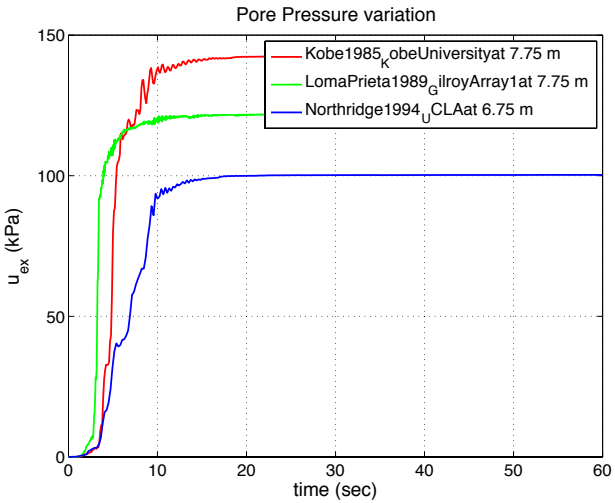


# *OpenSees & DesignSafe:* OpenSeesMP

# …Questions?

Maria Giovanna Durante, Ph.D.
Postdoctoral Research Fellow
University of Texas at Austin
mgdurante@utexas.edu

DESIGNSAFE-CI A NATURAL HAZARDS ENGINEERING COMMUNITY · TEXAS The University of Texas at Austin · TACC · RICE · Florida Tech