Autofakequakes Description/Instructions:

Scripts/Files:

autofakequakes_v3.sh (the main file that runs the FakeQuakes steps)

autofakequakes_v3_wrap.sh (a wrapper file that that you edit to set parameters. This calls the main file to run FakeQuakes)

autofakequakes_v3_to_txt.sh (The file you call to run things. You call this script with no arguments. This creates a unique name for the run so that the other scripts have access to it.

get_dataset_fromOSG.py (A python script that uses Paramiko to retrieve the dataset containing input files from OSG)

madair_mudpy_mudsing_image_complete_v1.sif (Singularity image which has the MudPy software and all its dependencies in it)

How to Run:

First you must have an environment with Singularity installed in it (for running FakeQuakes in my Singularity container) as well as Paramiko (for retrieving the dataset from OSG). I installed conda, and then used that to create a conda environment with Singularity and Paramiko in it.

You must also have the 'madair_mudpy_mudsing_image_complete_v1.sif' Singularity image sitting in your home directory – this is the same directory as where the 3 bash scripts and 1 python script must reside.

To run, you'll first need to have the 3 bash scripts, 1 python script, and the Singularity image sitting in the home directory of the AWS virtual machine. You'll also need to have the environment with Singularity and Paramiko active.

You'll also need to then edit the 'autofakequakes_v3_wrap.sh' script to have the correct parameters as you desire. You have the options to choose a 'dataset', set the number of 'nrealizations', and customize 5 different important parameters: Here is a description for each of these:

dataset: So, there is a variable set in the wrapper file called "dataset". This is what determines what input files will be retrieved from OSG. So rather than needing to upload files and input their names yourself (for the 5 required files for running FakeQuakes), you simply need to choose from 1 out of 4 dataset options. The 4 options for 'dataset' are:

small_chile - This is just the 5 Chile files using a small station list, so things run faster. When using this dataset option, distance and G matrices will be made.

small_chile_recycle - This is the 5 Chile files and it uses a small station list. This dataset also contains the distance and G matrices for these Chile files so that they're recycled which make things run much faster.

full_chile - This is just the 5 Chile files and it uses a full-size station list, so things take longer to run. When using this dataset option, distance and G matrices will be made.

full_chile_recycle - This is the 5 Chile files and it uses a full-size station list. This dataset also contains the distance and G matrices for these Chile files so that they're recycled which make things run much faster than when they're not recycled.

Next, you can set the 'nrealizations' variable. This is what determines how many ruptures will be generated, and thus, the number of waveforms that will be generated for those ruptures. To run FakeQuakes successfully, nrealizations % ncpus = 0. Here ncpus is set to 4 by default. This means that 'nrealizations' must be 4 at a minimum and it also must be a multiple of 4. The number of ruptures that are generated will be 4x the number that 'nrealizations' is set to. So, at the minimum when 'nrealizations' is 4, then 16 ruptures and their waveforms will be generated.

Lastly, you can optionally change 5 important parameters to customize the FakeQuakes run. These 5 parameters are:

utmzone: UTM zone that defines (roughly) the center of the fault model to help project geographic coordinates to cartesian coordinates. (e.g. "19J")

timeepi: The UTC date and time at which the synthetic events will originate

targetmw: Input parameters of numpy.arrange() which returns an array of values within a given interval. Decides of what approximate magnitudes FakeQuakes are generated for. Enter in 3 numbers (start, stop, step) that are separated with command and have no space in between. For Fakequakes we are trying to get data for high magnitude earthquakes. Because they don't happen often, we need to simulate them. (For example: "8.5,9.2,0.2")

maxslip: The maximum slip (meters) allowed in the model. A reasonable integer. (e.g "100")

hypocenter: Defines the specific hypocenter location (if force_hypocenter=True which the fault is not). The hypocenter is the starting location of a FakeQuake simulation. (e.g “0.8301,0.01,27.67”)

So, to run first choose a dataset option; the input files reside in OSG, so just set the name of dataset variable in the wrapper file. It will call the python script to retrieve the dataset and place the input files where they need to be for ‘autofakequakes_v3.sh’. Also choose how many ruptures/waveforms you want to generate by setting ‘nrealizations’. Also, you can optionally change the 5 parameters.

Once you have activated the environment with Singularity and Paramiko and set the variables as you want in autofakequakes_v3_wrap.sh, to run FakeQuakes you can then simply run the command:

‘nohup bash autofakequakes_v3_to_txt.sh’

or also

‘sh autofakequakes_v3_to_txt.sh’

This ‘to_txt’ file will output the unique name of the run so if you use ‘nohup’ to launch it, that unique run name will be appended to nohup.out as the latest run. If you use ‘sh’ then the unique run name will be output to the console.

What launching this does is launch the ‘autofakequakes_v3_to_txt.sh’ script which creates the required folders if they don’t exist, creates a unique run-name based off of the date and time (and it echoes that), and it runs the wrapper file using nohup. It also writes all the nohup and FakeQuake output to a uniquely named text file.

The wrapper file then retrieves the input files from OSG, starts writing the status of the run to an output folder, and then executes the autofakequakes_v3.sh script in the Singularity container with the correct arguments.

The autofakequakes_v3.sh script will run all the FakeQuakes steps while creating timestamps at the different steps during the process and writing them to a text file.

When things are finished, the output of the FakeQuakes run, as well as text-files containing its console output and status/timestamps, will be located in a unique folder:

~/fakequakesoutput/fakequakes_run<date>_<time>

In there will be a file named ‘fakequakes_run<date>_<time>_output.txt’ which contains the console output from the FakeQuakes run.

There will also be a file named ‘fakequakes_run<date>_<time>_status.txt’ which contains status updates and timestamps from the run.

There will also be the actual FakeQuakes output (ruptures, waveforms, etc.) in that unique directory. It will have been copied there after the run is complete and is located in a folder named 'fakequakes_run<date>_<time>'.

Again, you can find the unique name of each fakequakes run (with the unique date and time) after launching the autofakequakes_v3_to_txt.sh script; it is "echoed' in that script.